Consultant-as-a-Service: An Interactive and Context-driven Approach to Mobile Decision Support Services

Ahad Yarazavi and Kamran Sartipi
Department of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology
Oshawa, ON, L1H 7K4, Canada
{*Ahad.Yarazavi, Kamran.Sartipi*}*@uoit.ca*

Abstract

This paper presents a new generation of decision support services where techniques from software agents, semantic analysis, and data mining are orchestrated through a new cloud-based concept namely "Consultant-as-a-Service". The proposed concept will be offered as a set of new cloud application program interfaces that coach the user, who is not familiar with an organization, to effectively 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 consultant services can be provided for a variety of strategic business domains such as: banking, insurance, government, healthcare, and on-line shopping. A prototype mobile service has been developed using iOS for iPhone.

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 investments and efforts have been allocated to develop a long list of sophisticated business services, the effective and user-friendly offering of such services to the clients is still a challenge for the organizations, requiring them to expand their marketing departments and hire more information desk employees.

On the other hand, most customers of such large organizations are not familiar with these services and their long lists of features, which makes it extremely difficult for them to choose relevant services based on their needs. For instance, typically a customer who intends to invest his assets in a bank does not know the most suitable financial services that the bank can offer him based on his status; or he may not be familiar with features and advantages (or disadvantages) of different services to decide on their appropriateness. As a common practice, customers either call the information desk or visit the human consultants to get such strategic information. Moreover, to make an optimal decision, they should be able to compare the pros and cons of similar services in competing organizations.

In the proposed approach, the user installs an application (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 consultant service to serve the user, which collects the context of the user locally to select a specific business service (e.g., stock invest). This preserves the user's data privacy and reduces the communication traffic. The consultant service also uses the user's context to provide a list of third-party applications for the user to select. Once selected, the consultant service connects rele-

vant third-party applications with the business service to provide a customized environment for the user to effectively use the service.

As a case study, we modelled the system in a banking environment. The consultant service performs a series of interactions with the user to identify a proper attribute-value for each individual attribute of the service-context. The consultant service: i) prompts the text of a general question to the user, such as "*Which bank do you need service from*?" ii) collects the user's input, parses and tokenizes the input and performs a semantic search against the stored attributed-value synonyms within the WordNet database. The resulting standard synonyms of attribute-values trigger the next question for the user. iii) The above process will be repeated until the attribute-values for all attributes in the service-context are identified which means the specific service is selected. Then the consultant service invokes the selected service.

As the main contribution, this paper proposes a novel, sophisticated, easy-to-use, and cloud-based service selection technique that: i) coaches the user step by step to the desired service; ii) uses semantic analysis and natural language processing in the search process; and iii) uses data mining operations to explore hidden information in the organization's data assets to better guide the new users.

The remaining of this paper is organized as follows. Related work is discussed in Section 2. In Section 3 our proposed approach to Consultant Service is explained. Section 4 is allocated to a case study in the banking system. Finally, discussion and conclusion are presented in Sections 5 and 6.

2 Related Work

In this section, we discuss the approaches that are related to our work.

Semantic Matching

Semantic matching, as defined in [3], is obtained by computing the semantic relations between the concepts of the corresponding nodes of the graphs. In [4] Guinchiglia and Shvaiko describe an algorithm which takes two graph-like structures and produces mappings between the nodes that are semantically related. Possible semantic relations that are used in [4] are: *equivalence* (=), *more general* ($\sqsupseteq$), *less general* ($\sqsubseteq$), *mismatch* ($\perp$), and *overlap* ($\sqcap$). For example, "picture" and "image" are equivalent; "Europe" is more general than "Spain"; and "Asia" and "Europe" are mismatch. Overlapping does not provide any useful information because we cannot understand the whole concept based on overlapping. However, the other relations mentioned above provide useful information to understand the user's desire.

Decision Support System

Decision Support System (DSS) as defined in [7] and [5] 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 used for maintaining and retrieving knowledge from related domains in order to use in inference engine [5]. The inference engine 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 systems (ODSS), Group decision support system (GDSS), etc.

Concept Lattice

Concept lattice was founded by G. Birkhoff [1] in 1940. Concept lattice analysis provides a means to identify aggregation of *objects* that have common *attributes*. In [6, 9, 10] concept lattice analysis is used to identify the relation between program functions and their attribute values. More recently concept lattice analysis has been used in implementation of certain features of the software system [2]. The data mining and discovery nature of concept lattice analysis has been exploited in [13, 11] to gain knowledge from large databases. In the healthcare domain, the authors in [13] produced a database in which "diseases" and "symptoms" represent objects and attributes, respectively. Then they used concept lattice analysis to gather frequent itemsets (concepts) to assist the physicians to diagnose patients. Similarly, we use concept lattice analysis to recommend services and promotions which best match with the user's circumstances. To reduce the time complexity of the searching algorithm we built a concept database offline and then used it in the application at run-time.

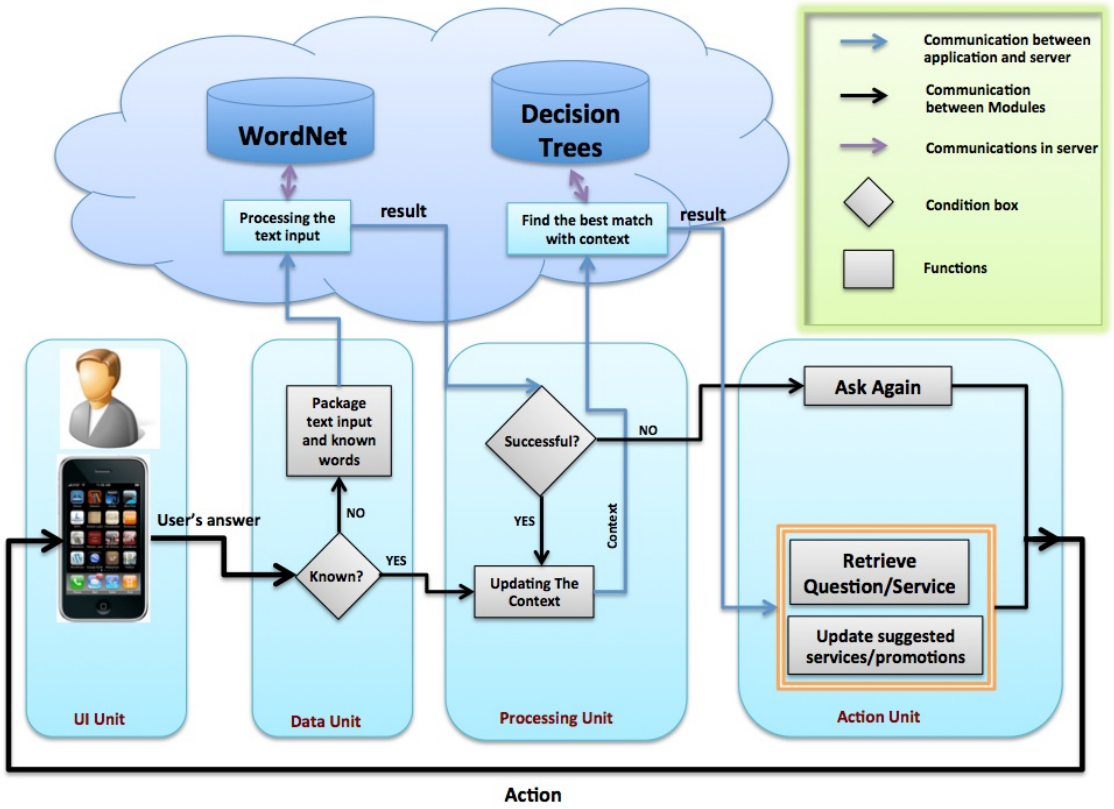

Figure 1: The architecture of the proposed Consultant-as-a-Service.

3 Proposed approach: Consultant-as-a-Services

In the proposed approach, a mobile consultant service provider, e.g., "mConsultant-Firm", provides registration facility for the organizations that will participate in the "Consultant-as-a-Service" facility. To offer this service, the mConsultant-Firm provides the following facilities for two groups of clients: i) *Registered organizations*: web-enabled applications for registering, billing, and customer service purposes; and ii) *Mobile users*: an application to be installed on their mobile devices (i.e., smart phones and tablets) which allows the users to select their "target organization" among the registered organizations, and to invoke the consultant service of the target organization to perform the consultation task for the users. Each registered organization has its own cloud-based server that provides: i) a specific *Consultant Service* to be installed on the user's mobile device, and ii) a set of *Decision Support Trees* at server side 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.

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

Step 2 (*Consultant service*). The application invokes that the specific consultant service of the target organization to be installed on the user's mobile device (as a local agent application). The consultant service starts interacting with the user by asking questions to fill out the attributes of the user context. At each iteration, the updated user's context will be sent to the organization's server asking for the next question available in the decision trees of that organization. At some point,

Algorithm 1 : User Interface Mechanism

```
1: while (True) do
2:    wait for Action Unit to send data;
3:    action = Receive the action;
4:    if (action contains a question) then
5:       Show the question;
6:       Wait for user to respond;
7:       Send the answer to Data Unit;
8:       update recommended promotions and services;
9:    else if (action contains a service) then
10:      show the link to the service;
11:      show users the map to nearest branch of the
         company;
12:      update recommended promotions and services;
13:   else
14:      show "Not Recognizable Answer";
15:      ask the question again;
16:   end if
17: end while
```

the user context will be detailed enough to allow the consultant service to identify the service(s) which are similar enough to the user's desired service. The consultant service will then prompt with a list of service names along with a short description for each service (service information). Finally, the user will select one or more services from the list to be performed.

Step 3 (*Recommended services/promotions*). The consultant service prompts to the user recommendations (in the form of services and promotions) that have been adopted by the current or previous customers of the organization with similar set of attributes to the user.

Step 4 (*Utility applications*). After the user selects the desired service, the consultant service provides additional utility applications that are required to effectively perform the service for the user. For example, a map will show the closest bank to the user.

Figure 1 illustrates the architecture of the proposed Consultant-as-a-Service approach. In the followings, the main four units of the architecture are discussed. These are: *User Interface Unit, Data Unit, Processing Unit* and *Action Unit*.

User Interface Unit. The UI Unit is responsible for communicating and collecting data from the user as well as presenting the results of the Action Unit to the user. At the end, the UI will display the service(s) that best match with the user's context. The pseudo-code in Algorithm 1 presents the high level behaviour of the system.

Data Unit. The Data Unit collects data from the user interface based on the asked questions. There are two types of input data as "known" and "unknown". For example, if the target attribute in the user context that we are aiming to provide a value for it is "occupation", the question that we ask from the user would be: "*what is your occupation?*". Then, we should also define different types of occupations that we expect our system to recognize. In this case, the known answers that the system recognizes would be: *employee, employer, self-employed, student, unemployed*. Such known answers are attached to the above question that the system will receive from the server. However, if the user's answer is something else, such as "waitress", it would be tagged as an unknown answer.

Processing Unit. This unit processes the input data (user's answer) and fills the targeted context attribute accordingly. As mentioned earlier, we have two types of input data (known and unknown). Handling known input data is straightforward as we know exactly what the user means, and hence simply fill the targeted context attribute with the same input data. However, the process is more complicated when the input data is unknown for the system. In this case, we use WordNet database to map the input data into a known answer for the system. For each targeted context attribute the following information will be sent to the server to be processed: "unknown input data", and the set of "known answers". In the server, using NLTK (Natural Language Toolkit), the system first tokenizes the input data to extract the important components of the unknown input, which can be a word (as noun or verb) or a sentence. Then, the WordNet databases and WUP (Wu and Palmer [12]) similarity metric are used to find the best match between the extracted component of the unknown input and the known answers. The identified closest answers to the unknown input data will be sent back to the mobile application. Otherwise, the server notifies the application with an error message stating that an acceptable similarity value was not found.

Action Unit. The Action Unit acts based on the current user context. If the system can not update the context in the Processing Unit it will ask the question in a different way so that the user's answer would be recognizable. Otherwise, the known an-

swer will be used to update the targeted context attribute. In this case, the system will be able to proceed by searching the decision-tree database to find the next decision-tree to use. If the instantiated user context so far is not informative enough, the system continues asking more questions to fill more context attributes. Finally, after a few more questions the user context will be informative enough that can be matched against the contexts of one or more services from the service registry, to be listed for the user.

Knowledge extraction. 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 consultant service to assist new users in their service selections. At each step during interaction with the user for collecting his/her attributes, the consultant 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 consultant 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 7. 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 generate a relation between the user attributes and the selected services and used them as described above. More detailed discussion

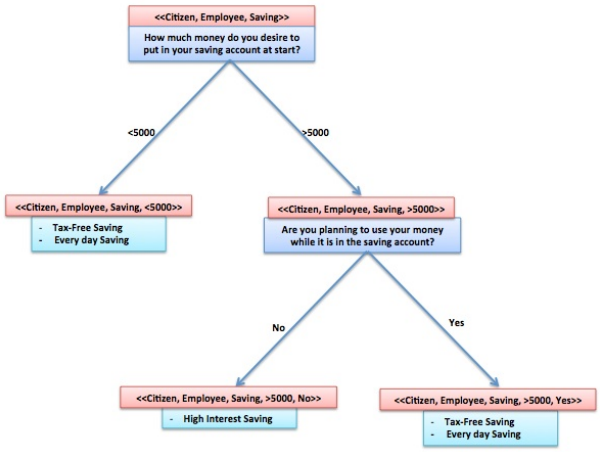

Figure 2: An example of a decision tree for TD Bank.

is provided in the next section.

4 Case Study

For implementation of the proposed system we have used a MacBook Pro computer with 2.3 GHz Intel Core i5 processor and 4 GB memory. We used XCode 4.6 to write the client-side of the application for iPhone and PHP for server-side services. Also we have used Python 2.7.2 for text processing part of the application.

To demonstrate the functionality of the system we implemented a banking application based on the services in TD Bank of Canada. 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 aim to gradually fill out, and accordingly suggest proper services to the user. Each attribute in the context tuple has an attached question along with a set of possible answers that the system recognizes (i.e., known answers). The decision trees are designed with respect to the attribute-tuple and the next proper question to ask.

Figure 2 illustrates a simple example of decision tree that we 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

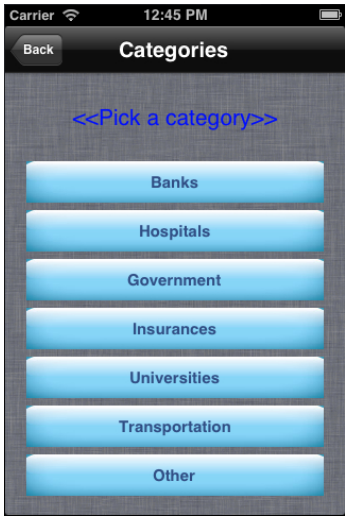

Figure 3: Example of different organizations that may provide consultant services.

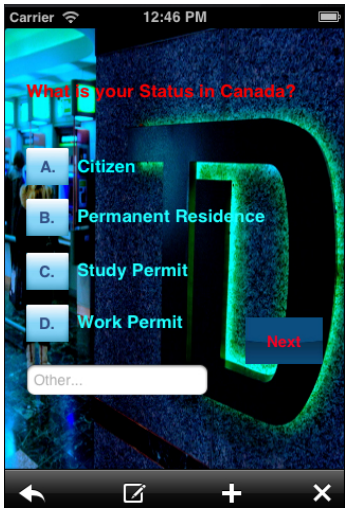

Figure 4: TD Bank consultant.

to suggest. The context of the root node for each decision tree has three common attributes; "Status in Canada", "Occupation", and "Type of Services" that the user desires. In our research with TD Bank services we identified these three attributes as the most important ones. Hence the consultant service would ask these questions in the beginning of the interaction with the user, and based on the user's answers it will match a decision tree which best suits with the user's context.

The proposed mobile decision support service is applicable for different application domains (e.g., banking, insurance, government, healthcare, transportation, and on-line shopping). Figure 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 "Banks" the application will show the names of all banks in Canada and the user can search for specific bank with which he intends to do business. In this case, when the user selects TD Bank from the list, the application searches the database of consultants in the server and retrieves the TD Bank consultant to run on the mobile device. This consultant only holds the first few questions and their known answers for that specific organization. Also the application would receive the link to the table of the decision trees specified for that organization. After receiving these information from the server, the application starts asking more questions from the user and filling out the context attributes required for the organization. Figure 4 displays the consultant for TD Bank.

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 user context will be filled with the answer. Otherwise, the user's answer will be tagged as unknown and the application will send the user's answer and the set of known answers for that specific question to the python script at the server side for text processing. At this point the system will attempt to map the user's answer with one of the known answers. In the Python script: i) user's answer will be tokenized and spell checked; ii) set of tokens will be reduced by trimming all parts except the "nouns"; iii) using NLTK (Natural Language Tool Kit) and WordNet, the system searches to find a semantic match between the resulting answer-array (tokenized, spell checked and reduced) with each known answer; and iv) resulting best matches will be returned to the mobile application as the "matched answers", provided that the calculated similarity values are greater than 50%. If the Python script could find the matched answer, the user's context will be filed out and updated context will be sent to server to proceed to 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 from the user's answer the user will be informed and the question will be asked again.

Finding a matching percentage between two

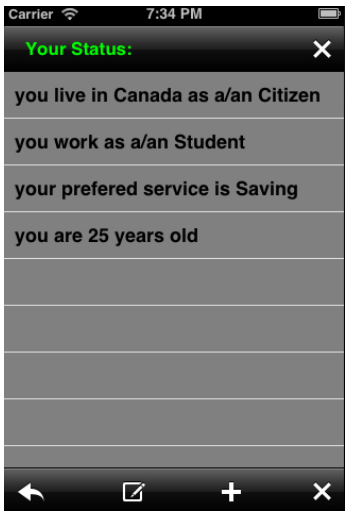

Figure 5: Accumulated user context during the consultation process.

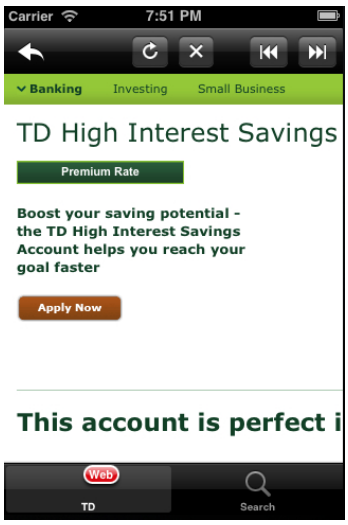

Figure 6: Consultant service redirects the user to the TD Bank website to run the selected service.

concepts (or synsets) is the main task of the Python script. To achieve this, the *wup-similarity* from WordNet is used. *wup* finds the path lengths of two concepts to the closest common root (if both concepts are in the same graph in WordNet database) and returns a value between 0 and 1. If two concepts are from different graphs the similarity function returns -1 as they have no common root [8]. For instance, *wup* returns 0.9 when it compares two words "employee" and "waiter". When the user answers the question: "what is your occupation" as: "I am a waiter in MacDonald's", the user's answer will be tokenize and only the nouns will be kept. In this case, the words "waiter" and "MacDonald's" will be retained. Then the system tries to find out if there is any similarity between these two words and the known answers . For this particular question the known answers could be "employee", "employer", "self-employed" and "student" and the highest similarity value will be 90% between the words "waiter" and "employee". Therefore, the application will update the user's attribute "occupation" with the value "employee". As shown in Figure 5, at each step in the questionnaire the user can observe his/her context to verify whether the application could recognize his/her circumstances correctly or not.

After asking several questions, the user's context would be specified enough to identify a service or a set of services which adequately match with the user's circumstances. In this case, the mobile application receives the name of the services which are appropriate for the user. Now, the user can select a preferred service to use and the consultant service would redirect the user to the website for that service to utilize the service. The Map application has also been integrated with the mobile application to show the nearby TD Banks to the user (Figure 6).

At each step the user can also view the recommended services, promotions, etc. which are most probably suitable for the user. The user receives these recommendations based on mined-information of the current customers, who have similar attributes to this user, and have applied for those recommended services.

In Figure 4, the "+" button at the bottom of the application is intended to provide "recommendations" for the user during each step of the consultation. The consultant service uses two concept databases to identify the proper recommendations. One database contains concepts from the "costumers-attributes" lattice (or relation) and the other contains concepts from the "services/promotions-customers" lattice. These two databases are used to provide appropriate recommendations (services/promotions) for the current user.

Consider the lattice in Figure 7 that is built based on the relation customers-attributes with 35 customers and 28 attributes. The generated concept lattice consists of 504 concepts (highly related groups of customers and their attributes) which are

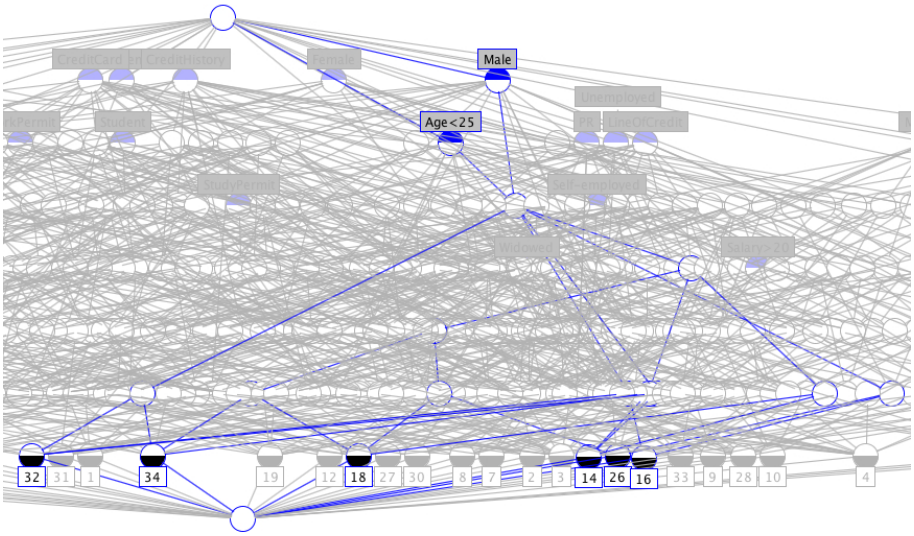

Figure 7: Lattice that represents the relation (customers, attributes).

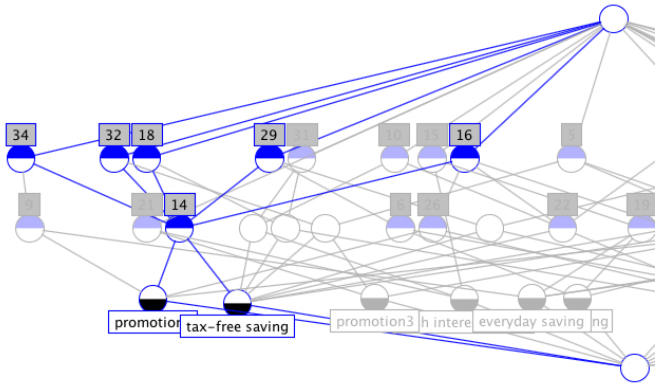

Figure 8: Lattice that represents the relation (services, customers).

stored in a database to be used in the application. As an example, for a male user with the age below 25 we find the concept *"<<male, age<25 >, <14, 16, 18, 26, 32, 34>>"*. This concept is highlighted in the lattice of Figure 7 which indicates that the customers with id numbers: 14, 16, 18, 26, 32, and 34, all share the same attributes (i.e., male and age<25). We also generate another lattice (Figure 8) based on the relations among all 35 customers and their selected services and promotions. By comparison of the concepts in these two lattices (Figure 7 and Figure 8) we obtain the concepts that have highest overlaps among their sets of customers. Note that the entity "customer" exists in both lattices. This property leads us to generate a new set of relations between "groups of attributes" and "selected service and promotions". Such relations will be used by the consultant service to provide recommendations to the current user based on the current user's accumulated attributes during the consultant-guided decision making process.

Based on the above example, Figure 8 highlights the best concept we can find (indicated as node 14 in the figure) which includes customers 14, 16, 18, 32 and 34 that all use the "tax-free saving account" and "promotion-2" services. Hence, these are the best options to display for the user based on what we know about his/her attributes until now (i.e., male user with the age below 25).

5 Discussion

During implementation of the application we encountered several technical problems. First, there is no practical way to directly connect an iPhone application (or generally xCode) to a Python script (text processing part of the project). We solved this problem by using PHP as a bridge to connect them. Hence we wrote a PHP program to wrap and execute the Python script and send the result to the iPhone application. Moreover, since in the Python script we search through the WordNet database for several times, each time we execute the Python script it takes about 5 seconds (considering our execution environment). We believe that the execution time will be highly reduced if the database and Python script will execute on a cloud environment which provides better processing power. To keep 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.

6 Conclusion

In this paper, we presented a novel service selection model for business enterprises 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 methods from software agents, decision support systems (DSS), and semantic analysis (WordNet) to develop a new concept namely Consultant-as-a-Service which has been implemented as an iPhone application. Moreover, we used concept lattice analysis from data mining domain to match the user's attributes with those of current and previous customers of the organization to provide new user with the best recommendation of services at each step of the decision making process. As the next step, we will continue working on expanding the Consultant Service concept to make it more sophisticated and support more features. Also by exporting the server side of the application to the cloud environment, we will enhance the usability of the application in real world.

References

[1] G. Birkhoff. Lattice theory. *American Mathematical Society, 1st edition*, 1940.

[2] Thomas Eisenbarth, Rainer Koschke, and Daniel Simon. Derivation of feature com- ponent maps by means of concept analysis. *Fifth European CSMR*, 2001.

[3] F.Giunchiglia and P.Shvaiko. Semantic matching. *Knowledge engineering review journal*, pages 265–280, 2003.

[4] F.Giunchiglia, P.Shvaiko, and M.Yatskevich. S-MATCH: An algorithm and an implementation of semantic matching. *European Semantic Web Symposium, LNCS 3053*, pages 61–75, 2004.

[5] K.Sartipi, N.Archer, and M.Yarmand. Challenges in developing effective clinical decision support systems. *Efficient Decision Support Systems: Practice and Challenges From Current to Future. In-Tech Open Access Publishing. ISBN 978-953-307-258-6*, pages 1–20.

[6] Christian Lindig and Gregor Snelting. Assessing modular structure of legacy code based on mathematical concept analysis. *In Proceedings of the 19th ICSE*, pages 349–359, 1997.

[7] M.Druzdzel and R.Flynn. Decision support systems. *Encyclopedia of Library and Information Sciences, Third Edition DOI: 10.1081/E-ELIS3-120043887*, 2004.

[8] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

[9] Michael Siff and Thomas Reps. Identifying modules via concept analysis. *IEEE Transactions on Software Engineering*, pages 749–768, 1999.

[10] Arie van Deursen and Tobias Kuipers. Identifying objects using cluster and concept analysis. *In Proceedings of the ICSE 1999*, pages 246–255, 1999.

[11] Kitsana Waiyamai and Lotfi Lakhal. Knowledge discovery from very large databases using frequent concept lattices. *Machine Learning: ECML 2000. Springer Berlin Heidelberg*, pages 437–445, 2000.

[12] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[13] Anis Yousefi, Negin Mastouri, and Kamran Sartipi. Scenario-oriented information extraction from electronic health records. In *IEEE CBMS'09*, pages 1–5, 2009.