Context Aware Web Service Discovery using Conceptual Clustering Based UDDI

Renuka Sruthi Thorati1,∗, M. S. Saleem Basha2 and P. Dhavachelvan3

1Department of Computer Science, PG Student, Pondicherry University, Pondicherry 605 014, India.
2Department of Computer Science, Assistant Professor, Pondicherry University, Pondicherry 605 014, India.
3Department of Computer Science, Professor and Head, Pondicherry University, Pondicherry 605 014, India.
e-mail: sruthithorati@gmail.com

Abstract. Current research in Web Service discovery is towards building Context Aware Pervasive Systems that give appropriate results based on users interests, but it is not completely discussed how far the retrieved services are actually relevant to the user. Also, the current context aware systems confine themselves to giving best results for user’s queries based on their profiles, which assume user to query based only on the interests which are listed or acquired by the context sensor. But this system can’t be validated in the cases where users search for services beyond their interests. In this paper we discuss context based on two aspects namely service context and user’s context. Also we discuss how the capabilities of UDDI can be extended to organize Web Services based on Conceptual Clustering so as to provide user with better results. Also we discuss how we achieve relevant services using clustering, context and user chosen QoS. In this paper we introduce an improvised model of COBWEB Algorithm for the incorporation QoS and Clustering Module for Conceptual Clustering.

Keywords: Web service discovery, Context aware, Conceptual clustering, QoS, UDDI.

1. Introduction

Context Aware is a term derived from Ubiquitous computing which means “A system can detect the change in an entity and act/respond accordingly to these changes”. Presently, the parameters which are considered to describe context are location, date, time and activity. Unfortunately all these parameters are not sufficiently enough to derive results which are relevant to user’s query. So we consider the definition of Context by Anind [1] which states

“Context is any information that can be used to characterize the situation of any entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”

Thus in our model we make use of Context Tool Kit as described by Anind [1] and also clustering so as to achieve context.

Our main focus in this paper is to get more appropriate results which are relevant to user’s context and user chosen QoS. Also we try to provide user with relevant recommendation of services based on user’s context. Traditional UDDI uses keyword matching for retrieval of web services. Later many methods have evolved which extended the services of UDDI through the implementation of semantics, clustering and ontology concepts based on user’s query. So, in this paper we also discuss how we organize Web Services in UDDI when provider publishes web services so as to reduce the time of query execution. We also discuss how we apply the techniques of conceptual clustering for organizing services in UDDI provided. For conceptual clustering we introduce an improvised COBWEB algorithm called iCOBWEB and a Clustering Module for non redundant organization of data.

2. Background

Conceptual clustering is a machine learning technique which deals with unsupervised learning and in which each cluster comes along with description. COBWEB is one such algorithm which makes use of conceptual clustering of

∗Corresponding author
© Elsevier Publications 2013.
data and builds hierarchy trees consisting of nodes as described in [2,3]. Here each node can be distinguished by a concept and represents a particular class and each class has objects which are associated with object descriptions based on events, facts, observations etc. Figure 1 shows how knowledge representation is done using COBWEB Algorithm.

Suppose that $C_0$ represents a concept Hotel and $C_1$ represents a concept Resort. We considered Object Set $O_1$ having 3 objects, each object having three properties namely [has−boatride, is−wifi enabled, has−tenniscourt]. If an object belongs to class $C_1$ then the probability that it has boat ride is 2/3. Thus concept description of node $C_1$ is described as category-conditional probability of the properties at $C_1$ which is $p(x/C_1) = [2/3, 1/3, 3/3]$.

3. Proposed System

Proposed Framework for Web Service Discovery encompasses Service Providers, Service Consumers, Interface Module along with Context Sensors, Clustering Module and a UDDI registry as shown in figure 2.

Service consumer

Service Consumer sends a Service Request (SR) using Interface Module which can be expressed as set of Inputs ($I$), set of outputs ($O$) and set of constraints ($C$) as in (1)

$$SR = \{I, O, C\}$$ (1)
We also obtain QoS parameters manually from the user which can be expressed as in (2)

\[ QoS = \{Av, Ac, As, Th, Re, Pt\} \]  

where,
- Av: Availability
- Ac: Accuracy
- As: Accessibility
- Th: Throughput
- Re: Reliability
- Pt: Processing Time

**WSDL**

QoS plays a major role in web service discovery as discussed in [4,5] as most of the retrieved service are not functional. So our proposed system insists on QoS integrated WSDL so as to provide users with services of desired QoS and for easy publishing of services. The Web Service Provider can either define QoS parameters while publishing services using Interface Module or can integrate QoS parameters into WSDL. QoS parameters can also be passed through WSDL file as described in paper [4].

**Service provider**

Service provider publishes Web Services using WSDL files along with QoS parameters through the Interface Module instead of using traditional WSDL files.

**Parser**

Parser will extract information from WSDL files using XML parser and passes them to UDDI for storage. Our XML parser is designed to extract name of the service, operations offered by the Web Service, description of the service and also QoS parameters. Our proposed model discusses only six important QoS parameters as shown in figure 3.

**UDDI**

Universal Description, Discovery and Integration assists in storing and retrieving information. Unlike traditional UDDI as discussed in [6,7], it stores information of web services along with QoS parameter values and context information of service which is extracted by context sensor and also optionally given by the provider in the form of tags.

**Interface module**

Interface Module helps in the flow of information, display of results, extracting information from Consumers and Providers. It encompasses Parser and passes the extracted information to UDDI for storage.

**Context sensor**

Context Sensor senses context information \((CI)\) from both consumer \((CI_u)\) and service \((CI_s)\). Context Toolkit discussed by Anind [1] can be used as a Context Sensor as it has the ability to capture, access and store context. The obtained CI is used for matching retrieved web services to filter non-relevant services as described in [8,9].

**Similarity checker**

We extract similar web services i.e., web services offering similar operations by applying the methods as described by Aparna Konduri [10].
Similarity of two services is calculated by a pair-wise comparison as given by (3).

\[
F_{\text{similarity}}(S_1, S_2) = \max_{i=1}^{m} \sum_{j=1}^{n} F_{\text{Operations}}(O_{1i}, O_{2j}) \times x_{ij}
\]

\[
x_{ij} = \begin{cases} 
1 & \text{combine } S_1 \text{ with } S_2 \\
0 & \text{else} \end{cases}
\]

\[
\sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, \ldots, m \\
\sum_{i=1}^{m} x_{ij} = 1, j = 1, 2, \ldots, n
\]

where,
- \(S_1, S_2\): Web services
- \(O_{1i}, O_{2j}\): Operations from Web Services \(S_1\) and \(S_2\)
- \(m, n\): Number of operations in \(S_1\) and \(S_2\)
- \(x_{ij}\): Weight which is set to 1 while matching operations \(O_{1i}\) and \(O_{2j}\).

Similarity between web services with operations \(O_1, O_2\) is also calculated via input and output parameters using the formula given by (4).

\[
F_{\text{similarity}}(O_1, O_2) = \max_{i=1}^{m} \sum_{j=1}^{n} F_{\text{Operations}}(I_{1i}, I_{2j}) \times x_{ij} + \max_{i=1}^{u} \sum_{j=1}^{v} F_{\text{Outputs}}(P_{1i}, P_{2j}) \times y_{ij}
\]

\[
x_{ij} = \begin{cases} 
1 & \text{combine } I_{1i} \text{ with } I_{2j} \\
0 & \text{else} \end{cases}
\]

\[
y_{ij} = \begin{cases} 
1 & \text{combine } P_{1i} \text{ with } P_{2j} \\
0 & \text{else} \end{cases}
\]

\[
\sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, \ldots, m \\
\sum_{i=1}^{m} x_{ij} = 1, j = 1, 2, \ldots, n \\
\sum_{i=1}^{u} y_{ij} = 1, j = 1, 2, \ldots, u \\
\sum_{j=1}^{v} y_{ij} = 1, i = 1, 2, \ldots, v
\]

where,
- \(I_{1i}, I_{2j}\): Input parameters of Web Service operations \(O_{1i}\) and \(O_{2j}\) respectively.
- \(P_{1i}, P_{2j}\): Output parameters of Web Service operations \(O_{1i}\) and \(O_{2j}\) respectively.
- \(m, n\): Number of inputs for Web Service operations \(O_1\) and \(O_2\) respectively.
- \(x_{ij}, y_{ij}\): Weights which are set to 1 while matching input parameters with output parameters.
- \(u, v\): Number of outputs of Web Service operations \(O_1\) and \(O_2\) respectively.

Semantic Similarity in the names of parameters and operations of the services are detected using WordNet tool. Through Similarity Checker all the similar services are thus grouped together using conceptual clustering. Through similarity checker we can also identify services which have few matching operations so that they can be branched under the same class.

**iCOBWEB**

Our proposed iCOBWEB is an extension of COBWEB algorithm which includes QoS parameters into the object properties. Our clustering model will take into consideration the user defined tags, context information, semantic and similarity conditions to group objects so as to facilitate context aware, QoS aware and semantic discovery. So now our clusters will not only define functional parameters of web services but will also define non-functional parameters. iCOBWEB organizes these non-functional parameters into another property list as shown in figure 3. Unlike the property lists of COBWEB which are binary-valued, our iCOBWEB allows rational values for QoS property lists \((Q_i)\). Like in COBWEB we define concept description of QoS for node \(C_1\) as \(p(x|C_1) = [0.91, 0.81, 0.50, 10.6, 0.63, 4]\) and also calculate likelihood for all QoS parameters.
We inherit the properties of COBWEB as discussed in paper [3] namely Category Utility (CU) and operators. These are used for building COBWEB and searching through the space hierarchy. CU serves as an evaluation measure and operators help in classifying objects. All the services are classified so as to form a hierarchical tree structure as described in COBWEB model. Services thus resulted from iCOBWEB give information ranging from more generalized to more specialized classes. Our model also includes “cutoff” parameter to regulate the growth as explained in [14].

Clustering module

Clustering module is a combination of Similarity Checker and iCOBWEB. When a new service is added, it checks for similarity conditions using Similarity Checker, and calculates CU only if similarity exists as described in figure 4.

Since CU cannot be qualified as a perfect measure for evaluation, we use Similarity scores to identify all similar services and then calculate CU to identify the best host. This method will serve as a better evaluation criterion when compared to CU as described in [3]. When a SR is recognized by Clustering Module it matches SR to existing classes by traversing through the cluster and search and identifying all the candidate classes.

Ranking

Ranking is given to the obtained services so as to provide user with more relevant services options for selecting services. The retrieved services are ranked based on the amount of match of consumers query and QoS to the classes formed as a result of clustering. Thus Ranking module provides user with relevant recommendations.

Steps involved in Proposed Model for Web Service Discovery are as follows:

Step 1: \( SR = \{I, O, C\} \) is obtained along with \( QoS = \{Av, Ac, As, Th, Re, Pt\} \) by Interface Module.
Step 2: \( CI_u \) is obtained using Context Sensor.
Step 3: Semantic data for \( SR \) is calculated in Similarity Checker using WordNet.
Step 4: SR along with Semantic data is matched with concept tree in Clustering Module resulting in Desired Services \((DS) = \{DS_1, DS_2, \ldots DS_n\}\).

Step 5: Consumer QoS and CI\(u\) are matched with QoS and CI\(s\) of \(DS\). Based on the amount of match, \(DS\) are given Ranking(R).

Step 6: Thus Interface Module provides consumer with \(DS = \{DS_1, DS_2, \ldots DS_n\}\) where \(R(DS_1) < R(DS_2) < \ldots R(DS_n)\).

4. Simulation and Analysis

For the proposed Clustering algorithm we used an Open Source Machine Learning Tool called WEKA. When we take a dataset of 20 web services our clustering algorithm yielded the following tree hierarchy is as shown in figure 5.

20 Instances resulted in 8 classes with 6 merges and 3 splits. The time taken to build the tree hierarchy is 0.01 seconds. Figure 6 shows time taken to build models for various datasets using iCOBWEB. For this we have collected 400 web services, of which 165 web services are developed on NetBeans 7.2.1 IDE using Java. Rest of the services are collected from online sources. As we use cutoff parameter, our algorithm can handle huge datasets.

Table 1 show the time variations when different test libraries are evaluated by existing COBWEB model and proposed Clustering Module (iCOBWEB + Similarity Checker).

From the above results we can see that our proposed Clustering Module saves time to build the model, as we calculate CU only with respect to few classes. Also our model gives more relevant results to users and avoids non-relevant recommendations.
Highlights of proposed model

✓ QoS satisfaction
✓ User relevant services
✓ Context based
✓ Semantic & Similarity checking
✓ Ranking
✓ Service Recommendations

Acknowledgement

This work is a part of the Research Project sponsored under the Major Project Scheme, UGC, India, Reference No: F. No. 41-619/2012(SR), dated 17 July 2012. The authors would like thank the Sponsored Agency for the financial support.

References