Abstract

This paper proposes a context-aware semantic planning graph technique for Web services composition. We first use an ontology based context model for extending Web services descriptions with information about the most suitable context for its use. Then, we transform the composition problem into a semantic context aware graph planning problem to build a set of best composed Web services based on user's context. The construction of the planning graph is based on semantic context-aware Web service discovery. This allow, for each step of the construction, to add most suitable Web services in terms of semantic compatibility between the services parameters, and their context similarity with the user's context. In the backward search step, semantic and contextual similarity scores are used to find composed Web services list. Finally, in the ranking step, a score is calculated for each candidate solution and a set of ranked solutions is returned to the user.

© 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of KES International

Keywords: Semantic Web Service, Web Service Composition, Adaptation, Context, Graph Planning;

1. Introduction

The Web service composition refers to the process of creating a composite service offering a new functionality, from simple Web services, through dynamic discovery process, integration and implementation of these services in a specific order to meet a specific need. To implement Web services and their compositions, many languages have been proposed. These languages include WSDL, OWL-S, XLANG, BPEL or WSCI.

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: benlaminerihab@yahoo.fr
Because of the diversity of users, which generally are not interested in the same features and do not have the same profiles (context, preferences,...), Web services must develop adaptation mechanisms to provide the user with relevant and adaptable services. According to Dey et al., "context covers all information that can be used to characterize the situation of an entity. The latter can be a person, place, or object relevant to the interaction between the user and the application, including the user and the application themselves". To ensure this need, it is necessary to integrate the context throughout the whole Web services life cycle. Many researchers propose their composition approach based on AI planning techniques, by describing a Web service as an action which is specified by its preconditions and effects. Moreover, planning is a costly computational approach and the size of the data involved in the planning process over the Web, will be much bigger than ones used in classical planning problems. Context information, which is the important element to be considered during selecting and combining services, can increase the acceptance and the effectiveness of composition. A service composition model which can integrate, and make use of context information to derive the optimal component services of the composite services is still an ongoing research problem.

In this perspective, using a means of context aware AI planning method to Web services composition is the central foundation of our work. For thus, we propose to exploit context information throughout the whole Web services lifecycle: in the description, discovery, composition (planning) steps. Furthermore, if the selected composed Web service is no longer functional, user can select another composed Web service from the list returned by our composition algorithm.

As main contributions of this paper, we mention: (i) the enhancing of the AI planning graph with the concepts of semantic similarity link and context similarity, (ii) a Web service composition algorithm based on the semantic context-aware graph, (iii) a method for extracting the set of ranked solutions based on the semantic similarity links matrix which stores the semantic links and their associated scores, and the context similarity table which stores the scores of context compatibility of each service with the user context.

The remainder of this paper is organized as follows. Section 2 highlights the related work about Web service composition based on AI planning and context-awareness. In Section 3, the context model and the adaptable (context-aware) Web service description are defined. In Section 4, we present our composition algorithm. A case study illustrating our composition method is presented in Section 5. In Section 6, evaluation of results is presented. Finally, Section 7 concludes this paper.

2. Related Work

In this section, we give an overview of some research works in the field of context-aware Web service composition.

In Tari et al., a layered design framework for service composition is proposed. Using a rule-based planning technique, it adopts an abstract way for generating plans, in order to adapt to the context's changes. However, user's context is reduced to its preferences in terms of QoS. This approach needs the setting up of abstract services directory and to grouping together services which can perform the same task. Yu et al. described a goal-driven approach for context-aware composition of Web services. A Goal Description Language and Context Condition/Effect are designed to describe the dynamic semantics of goal requirements and service capability. A planner is designed to dynamically compose services based on the current contexts, and a service runner is designed to invoke proper services based on the contexts and interactions with users. Nevertheless, the context is considered only in the composition step. It's neglected in the description, the discovery and the execution steps. Moreover, they used their own Web service description language.

The aspect-oriented paradigm is exploited in Li et al. to support context-aware semantic service composition. These compositions are performed by weaving context aspects within plain compositions to handle dynamic contexts. Contexts are defined as ontology concepts and context configurations are used to associate service messages to context concepts. However, the authors do not focus on automatic composition. Plain compositions already exist and are augmented by adding context service to become context-aware. In Baidouri et al., a context-aware composition specification is proposed as a base for the context-aware composition metamodel. They presented a Context-Aware Composition Builder tool in order to provide dynamic composition of services depending to the current user's context. Nevertheless, the semantic dimension remains ignored.
Mccheick et al.\textsuperscript{6}, proposed a semantic Web services composition model by orchestration which aims to face different problems of heterogeneity and adaptation to the context. The composition is adapted to context change during execution by applying aspect weaving on BPEL process. However, user's context is not considered in the planning step. Moreover, authors used a classical planning technique. Furno et al.\textsuperscript{7} described a design approach based on a semantic model for context representation by an extension of the OWL-S ontology by means of context conditions and adaption rules. Their planner is implemented using PDDL\textsuperscript{4} which is based on GraphPlan algorithm Blum, et al.\textsuperscript{8}. Even so, semantic relation between Web services' inputs and outputs are not considered. The context is also neglected both in the discovery and the execution steps. In Wang et al.\textsuperscript{9}, a heuristic method of Web services automatic composition based on a planning graph model is proposed. The service composition consists of two modules: the planning graph's construction and the search for composition scheme. The rich relationship information of use is not considered. A comparison of the studied works is given in Table 1. (+) and (X) mean the presence and the absence of the adaptation to context respectively.

<table>
<thead>
<tr>
<th>Work</th>
<th>Method/Technique</th>
<th>Atomic Web Services</th>
<th>Composed Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tari et al.\textsuperscript{1}</td>
<td>Rule-based Planning</td>
<td>Extension with QoS attributes</td>
<td>X</td>
</tr>
<tr>
<td>Yu et al.\textsuperscript{3}</td>
<td>Goal-driven planning</td>
<td>Extension with context concepts</td>
<td>X</td>
</tr>
<tr>
<td>Li et al.\textsuperscript{4}</td>
<td>Aspect Oriented Programming</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Baidouri et al.\textsuperscript{5}</td>
<td>MDA/NDC(Notify, Decide, Configure)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Mccheick et al.\textsuperscript{6}</td>
<td>Planning/Semantic matching</td>
<td>OWL-S extension</td>
<td>+</td>
</tr>
<tr>
<td>Furno et al.\textsuperscript{7}</td>
<td>Planning/PDDL(GraphPlan)</td>
<td>OWL-S extension</td>
<td>X</td>
</tr>
<tr>
<td>Wang et al.\textsuperscript{9}</td>
<td>Graph-based Planning</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Yan et al.\textsuperscript{10}</td>
<td>Graph-based Planning</td>
<td>QoS attributes</td>
<td>X</td>
</tr>
<tr>
<td>Our work</td>
<td>Context-aware semantic graph planning</td>
<td>OWL-S extension with context information</td>
<td>+</td>
</tr>
</tbody>
</table>

From this table we conclude that the context is not exploited in each step of the life cycle of Web services. Most of the studied works use context information in the description and the composition steps.

Our Web service composition method is also based on planning graphs, but in contrast of the works mentioned above, ours is context-aware semantic graph. Moreover, we provide a ranked set of best composition solutions as response to the user's request. Besides, we use the semantic similarity between concepts of Web services and the context similarity to enhance the planning graph. In addition, the user's context will be considered in each step of the life cycle of Web services i.e. from the description of atomic services to their composition.
3. Context Model and Adaptable Web Service Description

3.1. Context Model

Context is defined by Dey as "any information that can be used to characterize the situation of an entity". Several context models are proposed, including key-value models, object oriented models, markup schema models, ontology models, and logic models. We choose ontologies for context representation. Our choice is justified by the fact that ontologies are regarded as the most expressive model. Ontologies can model existing semantic between context parameters and their relations. Besides they are based on semantic Web languages that enable sharing, reasoning and reuse. Like in Najar et al.\textsuperscript{11}, we rely on multi-level ontology based context model. Using such ontology, a vocabulary is provided for describing context information. This ontology is composed of an upper level, that describes more specific context information. In this level, basic concepts are defined such that "User", "Environment" and "Device". This ontology is extensible to support domain-specific concepts. The detail of the basic context information, represented in the upper level of this ontology, is defined in the lower level which is domain-specific and varies from one domain to another.

3.2. Adaptable Web Service Description

Most recent research in service oriented computing, are oriented to the use of OWL-S to describe semantically Web services. OWL-S Martin et al.\textsuperscript{12} describes web service capabilities in three parts representing interrelated sub-ontologies named service profile, process model and grounding. The service profile expresses what the service does. It gives a high-level description of a service, for purposes of advertising, constructing service requests and matchmaking. The process model answers to the question: how is it used? It represents the service’s behaviours as a process and describes how it works. Finally, the constructs of the process model are mapped, by the grounding, onto detailed specification of message formats and protocols.

As demonstrated by works such Kirsch-Pinheiro et al.\textsuperscript{13}, Suraci et al.\textsuperscript{14}, the OWL-S ontology is a flexible and extensible language. Similar to these works, we extend service description in OWL-S by including context information. This extension concerns the service profile in order to allow the service provider to define context information describing the execution context for which the service is best suited. According to \textsuperscript{13,14}, due to its dynamic nature, context information cannot be statically stored on the service profile. In fact, context properties associated to service execution can evolve. Thus, in order to handle dynamic context information on static service description, we opt to store context information in an external file. Thus, service provider can easily update contextual non-functional information related to the service itself. The URL of this external file is stored in the service profile with the attribute "adaptedTo". To store the extended Web service description, we extended an implementation of UDDI register using JUDDIv3 and MySQL database.

4. Adaptable Web Service Composition Method

Because of the big number of Web services, which increase dramatically, generating the composed service becomes more difficult. Thus, it is important to automate the Web services composition in order to accelerate the development of Web applications. Automatic Web services composition problem can be easily mapped to AI planning problem where services correspond to operators, user request contains the planning goal and the composition corresponds to the plan\textsuperscript{15}.

Many types of AI planning techniques were used to solve Web services composition. These includes state-space search (Wu et al.\textsuperscript{16}), situation calculus (Zhao et al.\textsuperscript{17}), automatic theorem proving (Li et al.\textsuperscript{18}), problem deduction (Chen et al.\textsuperscript{19}) and planning-graph (Li et al.\textsuperscript{20}).

Graph planning based approaches filter active services into layers (planning graph) to prune the search space. Thus, a planning graph is a powerful and useful search space and more performing than classic planning based techniques. Generally, an AI planning graph is described as a sequence of layers, each layer containing a set of propositions $P_i$ and a set of actions $A_i$. The set of actions on the layer $i$ is defined as $A_i = \{ a_t | \text{ the preconditions of } a_t \in A_i \}$. The set of propositions on the layer $i$ is defined as $P_i = P_{i-1} \cup \text{eff}_{i-1}$, where $\text{eff}_{i-1}$ represents the effects of the actions $a_t \in A_{i-1}$. A solution of a planning graph is a layered plan consisting of a sequence of actions $\{A_1', A_2', ..., A_n'\}$ where $A_i'$ is a subset of
actions in $A_i$ that can be executed simultaneously. The preconditions of $A_i'$ represent a subset of $P_r$. Thus, approaches based on planning graphs are considered the most efficient approaches to solve large-scale automatic service composition.

4.1. Problem Formalization

In this section, we give clear definitions to the key concepts in context-aware automatic semantic Web service composition problem.

- Adaptable Web service: An adaptable Web service is denoted by a triple $w=(I,O, Cx)$ where $I$ is a set of input parameters, $O$ is a set of output parameters. $I$ and $O$ are both represented by domain ontology's concepts. $Cx$ is a set of context parameters represented by context ontology's concepts.

- Composition request: A composition request is defined as $ReqComp=(InReq, OutReq, CReq)$ where $InReq$ is the set of requested input parameters, $OutReq$ is the set of requested output parameters and $CReq$ is the set of the user context parameters.

- Semantic links Matrix: Semantic links established between the services on different layers of the context-aware planning graph are stored in the semantic links matrix. Two services $s_1$ and $s_2$ are semantically linked if there is a degree of match (DoM) (Paolucci et al.\textsuperscript{21}) between the set of outputs of service $s_1$ and the set of inputs of service $s_2$. In this matrix, columns and rows are both labeled with services from the planning graph. Formally, the semantic links matrix can be defined by $SLM = [slm_{ij}]_{i=1..n, j=1..m}$ where each element $slm_{ij}$ is defined by:

$$slm_{ij} = \begin{cases} 0, & \text{if } SimS(s_1.o_k, s_j.i_k) = 0 \\ slm_{ij}(s_1, s_j), & \text{otherwise} \end{cases}$$  \hspace{1cm} (1)

In (1), the semantic link between the service in the row $i$ and the service in column $j$ is represented by the element $sl_{ij}$. Each semantic link $sl_{ij}$ is a tuple: $sl_{ij}=(V, SimS)$ where $V = \{ s_i.o_k, s_j.i_k \mid s_i.o_k \in s_i.o, s_j.i_k \in s_j.i \}$ is a set of pairs of output parameters $s_i.out_k$ of service $s_i$ and input parameters $s_j.in_k$ of service $s_j$ for which $DoM>0$. $SimS$ is the semantic similarity score between the subset of the service $s_i$ output parameters and the subset of input parameters of service $s_j$. For each degree of Match (DoM) is attributed a score: $Exact=1$, $Plug-in=0.9$, $Subsumes=0.5$, $Fail=0$. $SimS$ is the average of DoM scores between the two parameters subsets mentioned above.

4.2. Composition Algorithm

This sub-section describes our adaptable semantic Web Services composition algorithm. Our composition algorithm takes $O$ (a domain ontology), $R$ (a set of adaptable semantic Web services), $CO$ (a context ontology) and $ReqComp$ (a user composition request) as inputs and returns a set of best solutions ranked based on their global score. A web service composition global score is composed of two parts: a semantic score which is the sum of semantic similarity scores between services pairs belonging to successive layers of the solution. The second part is the score of context compatibility of the solution with the user context and represents the sum of context similarity scores of each service of the solution with the user context. This context similarity is computed using the method proposed in (Najar et al.\textsuperscript{11}).

Our algorithm includes three stages: a Forward Search Stage, a Backward Search Stage and a Solutions Ranking Stage. In the Forward Search Stage, a context-aware planning graph is generated. In the Backward Search Stage, candidates solutions are calculated with their corresponding scores. Finally, the ranked set of solutions is returned to the user by the Solutions Ranking Stage.
4.2.1. Forward Search

The forward search algorithm (Fig. 1) is used to construct the context-aware semantic planning graph. In this step, semantic relation between inputs and outputs parameters of Web services is exploited to enhance the planning graph with semantic information. This is a realistic approach since exact matches are not always possible. This algorithm takes $\text{ReqComp}$, $R$, $O$ and $\text{CO}$ as its inputs and gives a context-aware semantic planning graph $G$, a semantic link matrix $\text{SLM}$ and a context similarity table $\text{CS}$ as outputs. Initially the context-aware planning graph $G$ contains the layer 0 where the set of actions $A_0$ is empty and the propositions set $P_0$ represents the initial state of the planning problem and contains the request input parameters. By mapping the AI planning graph problem to semantic Web service composition, each action in $G$ represents an atomic Web service. The input parameters of a Web service are mapped to the preconditions of an action in the graph $G$ and the output parameters are mapped to positive effects of an action. Both the output and input parameters are ontology concepts represented in the model as propositions.

The graph $G$ is iteratively expanded until one of the following stopping conditions is satisfied: the planning graph reaches a level for which the set of propositions contains all user required output parameters or when the planning graph reaches a fixed point level where $A_{i+1}=A_i$ and $P_{i+1}=P_i$. For each layer $i>0$, new services are discovered from the repository of web services based on semantic matching between the services' input parameters and the set of propositions on layer $i-1$, and the context compatibility of the discovered services with the user's context ($\text{Select\_SimSC\_Services}$). For each new discovered service and if it has not been previously added to the graph $G$, a new entry is added to the semantic link matrix ($\text{Update\_SLM}$) and a new entry in the context table is created ($\text{Build\_CS}$) and the output parameters of the service are added to the proposition layer $P_i$.

![Fig. 1. Forward search algorithm.](image)

4.2.2. Backward Search

After the construction of the context-aware semantic planning graph, the semantic link matrix and the context table, we use the backward search algorithm (Fig. 2) to find best solutions for the composition problem, $\text{ACSWS\_List}$. The backward algorithm begins from the last proposition layer of the planning graph $G$. Let $g$ be the set of goal propositions that need to be achieved at a given proposition layer $P_i$. The algorithm, using the semantic link matrix, will find a set of actions from $A_i$ such that these actions together achieve $g$, and then select those with the highest semantic link value. The union of preconditions of these actions represents the new goal set to be
achieved in proposition layer $P_{\ell-1}$. This process (Fig. 3) will be recursively executed until the first proposition layer of the graph $G, P_0$.

4.2.3. Solutions Ranking

The ranking step (Fig. 4) takes the set of best composition solutions returned by the backward stage as input and gives an ordered set of composed Web services, $Ordered\_Sols$, ranked by their scores using the function $Sort$.

This score is calculated for each composed Web service with the following formula:

$$Score_{sol} = \left( w_{sem} \times Score_{sem}(Sol) + w_{context} \times Score_{context}(Sol) \right) / (w_{sem} + w_{context})$$

(2)

Where $Score_{sem}$ is the semantic score of a solution which is the sum of semantic similarity scores between pairs of services belonging to successive layers of the solution, $Score_{context}$ is the score of context compatibility of the solution with the user context and represents the sum of context similarity scores of each service of the solution with the user context. $w_{sem}$ and $w_{context}$ represent the weights accorded to the user preferences related to the semantic and context relevance of a composition solution.

5. Case Study

In this section, we illustrate our composition method by a case study. We consider a simple scenario where the user is interested in making the travel arrangements for the holidays. The user’s request and an extract from the service advertisements are illustrated in Table 2 and Table 3 respectively. Both of them are expressed in terms of ontological concepts and are enriched with a description of the user's context for the request and the context the
more suitable for the execution of the service for each service advertisement. For simplification, the context of each service \( S_i \) is denoted by \( C_i \).

### Algorithm 4: \( \text{sols-ranking} \)

**Input:** \( \text{ACSWS\_List} \) : List of valid adaptable composed Semantic Web Services  
**Output:** \( \text{Ordered\_Sols} \): ranked composition solutions  

\[
\text{for each sol in ACSWS\_List do} \\
\text{Score}_\text{sol} = \frac{w_{\text{sem}} \cdot \text{score}_\text{sem}(\text{sol}) + w_{\text{context}} \cdot \text{score}_\text{context}(\text{sol})}{w_{\text{sem}} + w_{\text{context}}} \\
\text{end} \\
\text{Ordered\_Sols} = \text{Sort(ACSWS\_List, Score}_\text{sol})
\]

Fig. 4. solutions ranking algorithm.

Based on the user request and the set of available services, the composition planning graph \( G \) (see Fig. 5), the matrix of semantic links \( SLM \) and the table of context similarity \( CS \), are iteratively built by the composition algorithm.

<table>
<thead>
<tr>
<th>User's Inputs</th>
<th>User's Outputs</th>
<th>User's context</th>
</tr>
</thead>
<tbody>
<tr>
<td>EuropeanSourceCity, EuropeanExoticDestinationCity, Check_InDate, Check_OutDate, PersonsNbr, DoubleRoom, LuxuryMediterraneanHotelType, HotelRoomsNbr</td>
<td>FlightConfirmation, AccommodationConfirmation</td>
<td>Device.Type=Intel Core2 Duo, Device.Software.OS=W7, Device.Memory=1024, ContPref.Texte=True, ContPref.Image=True, AffPref.CharacterSize=12, AffPref.Font=Times New Roman</td>
</tr>
</tbody>
</table>

**Table 3. An extract from the set of Web services.**

<table>
<thead>
<tr>
<th>Code</th>
<th>Service</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>DestinationCity, SourceCity, Check_InDate, Check_OutDate</td>
<td>FlightIdentifier</td>
<td>C1</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>ExoticDestinationCity, SourceCity, SummerCheck_InDate, SummerCheck_OutDate</td>
<td>FlightIdentifier</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>EuropeanExoticDestinationCity, EuropeanSourceCity, Check_InDate, Check_OutDate</td>
<td>EuropeanFlightIdentifier</td>
<td>C3</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>ExoticDestinationCity, HotelType</td>
<td>HotelName</td>
<td>C4</td>
<td></td>
</tr>
</tbody>
</table>

Applying the backward search algorithm on the graph \( G \), the list of best solutions for our composition problem is: \( \text{ACSWS\_List\_List} = \{\text{Sol1} = \{(S3, S8)\}/(S4, S11)\}, \text{sol2} = \{(S3, S8)\}/(S6, S10)\}, \text{sol3} = \{(S2, S9)\}/(S4, S11)\}, \text{Sol4} = \{(S2, S9)\}/(S6, S10)\} \). For each solution of the previous set, a score is calculated by Algorithm 4 and the set of ranked solutions with their respective scores is illustrated in Table 4.

### Table 4. The ranked solutions set.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Semantic Score</th>
<th>Context Score</th>
<th>Final Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sol1</td>
<td>1</td>
<td>0.807</td>
<td>0.903</td>
</tr>
<tr>
<td>Sol2</td>
<td>1</td>
<td>0.807</td>
<td>0.903</td>
</tr>
<tr>
<td>Sol3</td>
<td>1</td>
<td>0.737</td>
<td>0.868</td>
</tr>
<tr>
<td>Sol4</td>
<td>1</td>
<td>0.737</td>
<td>0.868</td>
</tr>
</tbody>
</table>

### 6. Evaluation

We validate our work on the 161 semantic Web services of the travel domain from the OWL-S TC collection. The semantic descriptions of those services are extended with context information describing the execution context.
for which each service is best suited. Then those descriptions are published in the extended UDDI registry. We use the precision and recall measures to evaluate our work. The precision assesses the number of true relevant composite Web services identified among the returned set of services, while recall assesses the number of true relevant composite returned services among the real relevant composite services. These two concepts are often used because they reflect the user's point of view: if the precision is low, the user will be unsatisfied because he will have to waste time reading information that does not interest him. If the recall is low, the user will not have access to information he wanted.

In our work, a composite Web service is relevant if its score calculated by equation (2) is higher than 0.5. We conduct a series of experiments on a set of users to determine this threshold. We note that users are satisfied if the composite Web service score is equal or higher than 0.5. We set $w_{sem}$ and $w_{context}$ to 0.5. In Table 5, we show experimental results related to 5 queries. From this table we notice that the average precision is about 0.76674. This indicates that users are satisfied since returned composite Web service are relevant to their functional and non functional needs. The average recall is about 0.644 which means that the number of not returned relevant Web services is low.

7. Conclusion

This paper proposed a semantic adaptable Web services composition method based on semantic context-aware planning graph. Our composition method is composed of three stages. The first one is the graph planning construction based on the discovery and the selection of atomic Web services which satisfy the user's context and his preferences. In this step, semantic relations between inputs and outputs of Web services are exploited to enhance the planning graph with semantic information. The second step is the backward search in which a set of best composed Web services is extracted. The third one is to rank the set of extracted solutions based on their semantic and context scores.

Fig. 5. The planning graph.
In the future, we intend to test our composition method on more larger and complex collections of Web services and discuss its performance. We intend also to propose a formal verification method for composed semantic adaptable Web services. In this way, the context is considered in each step of the Web services lifecycle: description, publication, composition, formal verification.

### Table 5. Precision and recall for the travel domain of OWL-S TC collection.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>User's context</th>
<th>Returned composite Web services</th>
<th>Relevant returned services</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITY</td>
<td>LUXURYHOTEL</td>
<td>C1 (queryTravel2Context2.owl)</td>
<td>54</td>
<td>36</td>
<td>0.666</td>
<td>0.75</td>
</tr>
<tr>
<td>ACCOMMODATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTIVITY</td>
<td>HOTEL</td>
<td>C2 (queryTravel2Context4.owl)</td>
<td>27</td>
<td>19</td>
<td>0.7073</td>
<td>0.631</td>
</tr>
<tr>
<td>HOTEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTIVITY</td>
<td>BEDANDBREAKFAST</td>
<td>C3 (queryTravel2Context3.owl)</td>
<td>10</td>
<td>10</td>
<td>1.00</td>
<td>0.55</td>
</tr>
<tr>
<td>HOTEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTIVITY</td>
<td>HOTEL</td>
<td>C4 (queryTravel2Context2_1.owl)</td>
<td>12</td>
<td>9</td>
<td>0.75</td>
<td>0.666</td>
</tr>
<tr>
<td>BEDANDBREAKFAST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTIVITY</td>
<td>LUXURYHOTEL</td>
<td>C2 (queryTravel2Context4.owl)</td>
<td>7</td>
<td>5</td>
<td>0.714</td>
<td>0.625</td>
</tr>
<tr>
<td>ACCOMMODATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References