This is the published version of a paper published in *Computers & electrical engineering*.

Citation for the original published paper (version of record):

Mobile services discovery framework using DBpedia and non-monotonic rules.
*Computers & electrical engineering*
http://dx.doi.org/10.1016/j.compeleceng.2016.02.016

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-12065
Mobile services discovery framework using DBpedia and non-monotonic rules

Nasredine Cheniki a,∗, Abdelkader Belkhir a, Yacine Atif b

a Laboratory of Computing Systems, University of Sciences and Technology Houari Boumediene, USTHB, Algeria
b School of Informatics, University of Skövde, Sweden.

ABSTRACT

Mobile services are constantly evolving, thanks to improvements in performance of mobile devices and wireless networks. As a result, there is a need for and efficient supply of discovery processes that will even allow non-technical users and developers to publish, discover and access services in a mobile environment where non-functional properties (context and quality of service information) play an important role in the discovery process in conjunction to functional properties. In this paper, we propose a user-centric mobile services discovery framework that enriches functional descriptions of mobile services with semantic annotations from DBpedia knowledge (the semantically-structured version of Wikipedia) which covers multiple domains and provides lightweight ontologies. In addition, it offers open tools that can be used to simplify the provisioning and discovery of mobile services. The framework allows users to rank services using non-monotonic rules, which define their desired choices based on the context and quality of service information. Experimental results show that our framework provides efficient discovery results of efficient mobile services.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

The number of mobile phone subscribers is constantly increasing. According to PortioResearch [1], the trend has been transcending such that the prospect of subscribers by the end of 2016 is expected to reach 8.5 billion from the figure of 6.5 billion recorded at the beginning of the year 2013. It further indicates that the handset industry will continue to grow in the next couple of years as more users are getting broadband access to Internet services through their mobile devices. On the other hand, performance of mobile devices is constantly enhancing, with rapid growth in mobile communication networks. This rapid evolution has resulted in the wide proliferation of pervasive environments in which services are accessible, regardless of the time and geological locations. As they are improving consumers daily life, these services are massively expanding across our ecosystems; expected to provide novel experiences and personalization opportunities to the consumers. Hence, providing efficient discovery mechanisms to locate intended services in mobile and dynamically changing environments is expected to meet the challenges of these advances in mobile service provisioning.

Service discovery is the process of finding appropriate services by evaluating a service query [2]. This process usually consists of matching a user query to a set of service definitions, in order to retrieve targeted services. In mobile environ-

∗ Reviews processed and recommended for publication by the Editor-in-Chief by Guest Editor Dr. R. K. Murugesan.
E-mail address: nacei.cheniki@gmail.com (N. Cheniki).
http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
0045-7906/© 2016 Elsevier Ltd. All rights reserved.

Please cite this article as: N. Cheniki et al., Mobile services discovery framework using DBpedia and non-monotonic rules, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
mements, user discovery request have to take into account context information (such as location, time and preferences), as well as Quality of Service (QoS) information (such as price, rank and service availability). These criterions play a significant role in the discovery, while helping deliver the right services to the right users, at the right time, in the right place, and with the right QoS expectations.

Mobile service discovery based on semantic matching techniques has received significant attention from researchers in recent years. Considering concepts and their relationships, semantic matching could overcome the limitations of traditional matching models that are based on purely syntactic matching. Semantic matching relies on conceptual models describing specific domain knowledge, known as ontology. Ontologies are widely used to describe services as well as their context and their intrinsic QoS information. The formal language representation of these semantic attributes allows a common understanding and facilitates sharing, integration and reasoning of information [3].

Previous research works have used in either generic or domain-specific ontologies to enrich their mobile service descriptions [4]–[9]. These ontologies usually cover a restricted number of domains, are costly in terms of maintainability [10, 11] and some of them are considered to be over-expressive for semantic search and discovery processes. In addition, due to the variety of mobile services, they could not be well covered by domain-specific ontologies, and as domains evolve, old service descriptions become obsolete. On the other hand, most of domain ontologies are mono-language; as extra effort will be required to as much as consider multilingual discovery, for example by translating concepts and properties into a target language.

In this paper, we propose to overcome these limitations by annotating functional aspects of mobile services using DBpedia [10], a multi-domain knowledge base extracted from Wikipedia, to enable the design and development of a semantic Web services discovery framework based on structural containers (concepts, resources and categories). DBpedia covers multiple domains with a large number of concepts, instances and categories that can be used to describe a considerable range of services. It combines advantages of multi-domain and domain-specific ontologies by linking its instances to concepts from hand-crafted ontologies to enable rich semantic applications development. Besides, DBpedia evolves dynamically as Wikipedia changes. Hence, DBpedia is constantly updated with regular addition of new information. Our proposed discovery framework supports multi-languages, where users have the ability to search for mobile services across different languages, thanks to the globalized characteristics of DBpedia.

The proposed approach is user-centric, where the discovery solutions involve service providers who describe their services in terms of offered capabilities. Users issue requests using keywords, without getting involved in the semantic details that are used to annotate provisioned services for effective discovery. The framework provides a ranking mechanism of non-functional mobile services properties to sort them according to users context and the expectations of QoS information. The user defines a list of choices represented by non-monotonic rules in order to select services satisfying his/her context and QoS requirements, based on extended context ontology.

The contributions of this paper can be summarized as follows:

- A user-centric mobile services discovery framework based on the consumption of Linked-Data provided by DBpedia and non-monotonic rules. DBpedia enfolds cross-domain ontologies to describe functional properties (capabilities) of mobile services.

- To represent non-functional properties of mobile services, we propose an extended ontology to represent context and Quality of service information, to enable:

- A ranking and filter process for mobile services is based on the application of non-monotonic semantic Web rules where users represent their non-functional mobile service needs through semantic rules.

In this work, we consider mobile services as one that can be published in a mobile environment, for the purpose of being discovered by moving users from their mobile devices. Context information distinguishes these services from traditional ones. Service providers are able to announce the provision of concrete services that could be discovered from mobile devices, such as restaurant service, taxicab service, healthcare service or mechanic service; in addition to software services.

The remaining sections of this paper are organized as follows: Section 2 outlines some of the related research works. Section 3 introduces a motivational scenario illustrating the importance of mobile services discovery. Section 4 introduces MobiSO, a mobile services specification of an ontology to describe and annotate mobile services, their context and related QoS information. Section 5 discusses semantic annotations and the integration of DBpedia in mobile services specifications. Section 6 discusses semantic Web rules focusing on non-monotonic rules. Section 7 reveals the architecture of the proposed discovery framework. Section 8 provides a prototype implementation of the proposed discovery framework along with an overview of the used technologies and tools. This is followed by an evaluation of the proposed framework focusing on the discovery effectiveness and performance. Finally, Section 9 concludes the paper with a summary of our work and a proposed set of future research perspectives.

2. Related work

Traditional UDDI based discovery approaches have adhered to many limitations; for they have not been designed to support semantic search or adapt discovery processes to take into consideration mobile services context and QoS properties.

---

1 This article is a revised and extended version of [12].
These properties are very important to extend the scope of discovery beyond traditional fixed points, promoted by UDDI specification. Subsequently, it has drawn a great focus from a wide range of research communities. EASY [4] services are described using a proposed OWL based ontology called EASY-L. A set of conformance relations EASY-M (Matching) based on EASY-L are developed to perform context and QoS based service matching. The proposed discovery solution extends the base algorithm for semantic service matching developed by Paolucci et al.[13]. ConQo [5] enhances services discovery by considering context and QoS information. Authors proposed a new ontology for context information called WSMO-Context in order to extend a QoS-based ranking algorithm. Mustafa Adacal et al. [6] proposed an agent-based mobile Web services discovery approach where each service is described with a generic service ontology that allows service providers to refer to a specific domain and a service class with functional and non-functional attributes. A simple matching algorithm is proposed based on a matching service domain and related classes in addition to contextual and preferences properties. ProMWS [7] proposes pre-fetching strategies to predict usersâ query in order to enhance mobile peer-to-peer service discovery. The solution provides a simple semantic matching algorithm, which relies on a single common ontology. Toninelli et al.[8] describes services by their provided capabilities that are annotated through domain specific ontologies. Authors extend Paolucci et al. [13] matching algorithm to develop a semantic capability-based matching algorithm. Niazi and Mahmoud [9] integrated their own service ontology in Delivery Context Ontology released by W3C to develop a semantic matching and ranking algorithm which is based on user profile, preferences with device capabilities description using proposed ontology.

Previous works bring interesting solutions for mobile services discovery, however none of them tackled discovery aspect using semantic data from DBpedia to match functional properties of mobile services, and semantic non-monotonic rules so far, to rank their non-functional parameters.

3. Motivating scenario

Adam was traveling on the highway; where a crash occurred between him and a car driving in his front. Adam rushed out of the car after the shock to reassure the situation of the car owner he collided with, to find him completely fainted. Adam took out his smart phone to quickly look for the nearest mobile doctor on the same road using MobiSDiscovery (an application that facilitates mobile services discovery). The application contacted the server containing information about persons providing services who were within walking distance of the accident.

Unfortunately the application could not find any doctors; however, it found a nurse within a walking distance. Adam asked him to provide medical relief service, so a message alert was received by the nurse who opened the MobiSDiscovery application. The nurse received an urgent request for medical service and identified the place which asked him to help and headed directly to it. The nurse arrived and provided first aid to the injured before the ambulance arrived; which obviously required a long time by virtue of its distance from the scene. After check on the status of the patient, who was transferred to the hospital, Adam claimed that his car had gotten a malfunction and needed to be taken to the nearest mechanic. He used the same application to look for the nearest car carrier with the lowest price. Fortunately, he found the service he was looking for within walking distance from him. The service provider only accepts payment through credit card. Adam used the same application to look for software that permitted him to take benefit of the service. After successfully performing the payment, the summoned service is headed for the scene.

The above scenario illustrates benefits of semantic mobile services discovery. Semantic-based discovery offers highly efficient results for users by extending user query with more semantically-related terms and concepts (for example searching for a doctor may lead to any health-care stuff). In addition, taking context (such as geographic position) and quality of service (such as price) properties increased reliability with efficiency of the discovery issue. In this work we offer a discovery capability by providing semantic, context and quality service based discovery.

4. MobiSO, ontology for mobile services description

Many research efforts have been conducted to develop context and QoS information models for mobile services and applications. These models were initially created to express device capabilities and user preferences. They are based on eXtensible Markup Language (XML), Resource Description Framework (RDF) [14], Resource Description Framework Schema (RDFS) [15] or Ontology Web Language (OWL) [16]. Semantic Web technologies (RDF, RDFS, OWL) are widely used because they support processes of interoperability, sharing, integration and reasoning [3].

XML-based models do not provide any semantics and thus they have been rapidly substituted with semantically-enriched RDF(S)/OWL models. Composite Capabilities Preference Profile (CC/PP) [17] is a W3C recommended RDF model to describe device specifications and user preferences. It can be used by content providers to adapt their content for various devices. User Agent Profile (UAProf)² is a specialization of CC/PP usually used by content providers to output adequate format of different devices. RDF models allow common understanding and facilitate content customization. However, they still lack expressiveness and do not describe the semantic relations between its different components, resulting in limited semantic inferences.

Ontology-based models (mainly based on RDFS/OWL) empower RDF models to overcome the above limitations. Context Ontology Language (CoOL) [18] is a generic context description language created based on a proposed model called Aspect

Scale Context Model (ASC) to provide context interoperability and facilitate service discovery. However, this ontology is not suitable to describe abstract context information [19]. CoDAMoS [20] is another generic and extensible ontology that defines important reusable context ontologies such as User, Environment, Platform and Service.

In Web service domain, Web Service Modeling Ontology (WSMO) and OWL-S are two widely-used ontologies allowing semantic descriptions of Web services to enable automated service discovery and composition. WSMO-Context [5], WSMO-QoS [21] and WSMO-M (Mobile) [22] extend WSMO in order to support context and/or QoS information, while OWL-S has been extended [23] to add context attributes that enable contextual and inherent service compositions in pervasive environments. WSMO and OWL-S models are heavyweight solutions and present high complexity in terms of definition and processing.

The above models exhibit different drawbacks of contextual information. Authors in [19] proposed mIO! an ontology network composed of a set of ontologies developed to model information for contextual applications and services. This ontology includes 433 classes, 277 object properties, 156 datatype properties and 364 instances. It is actually made up of eleven ontologies, namely: Service, Provider, User, Environment, Location, Device, Time, Interface, Network, Source and Role. Many of these ontologies have reused existing ontologies such as FOAF, SOUPA, Delivery and CoDAMoS. mIO! Ontology Network was designed to facilitate the creation of extensions for concrete use cases.

We have developed an extension that fits our requirements called MobiSO (Mobile Services Ontology). In our ontology (Fig. 1), we differentiate between context information and QoS information, by defining two adequate classes ContextInformation and QoSInformation. MobileService class represents mobile services and extends the generic class Service. ConcreteService class is defined to distinguish concrete services from software services. MobileServiceProvider class models providers of mobile services.

ContextInformation class contains information such as location, address and current time, while QoSInformation class includes attributes such as price, service availability days and its rate. More information can be added from mIO! Ontology as needed; for instance, we can incorporate Device ontology to ContextInformation class for description of device capabilities.

In order to enrich semantically functional description of mobile services, we introduced three annotation properties: resourceReference, conceptReference and categoryReference. These properties will be used to map an external resource (individual), concept (class) or category reference to a service definition (described later in Section 5). This approach is inspired from SAWSDL (Semantic Annotations for WSDL and XML Schema) [24], which extends Web service description documents (WSDL) with pointers to external semantic definitions. Each service in MobiSO is represented with a set of capabilities that are described by external references. We adopt Capability-based service description as it allows the development of user-centric discovery algorithm [8] where developers as well as non-technical users provide their services.

Fig. 1. Essential parts of Mobile Service Ontology (MOBISO) Model.
Listing 1 illustrates our model with a description of a medical service associated with context and QoS information (using Turtle syntax). This service is identified by an instance of ServiceIdentification class, which contains a name property (lines 11, 15 and 16). The described service has context information, revealing its geographic position (lines 19, 22–28), and three supported languages (line 20). The quality of service information is described by service providing days (line 31), and its price (line 32–36).

5. Semantic annotations with DBpedia

Tim Berners Lee (the inventor of the Web) along with other colleagues [25] have introduced the semantic Web initiative that aims to structure Web contents to make them machine-accessible and understandable by adding descriptive information (meta-data) to existing Web resources (Web pages, Web services, images, etc.). This process is called semantic annotation, where semantically well-defined terms are associated to Web resources, in order to remove information ambiguity and allow software agents to; understand, exchange, integrate and discover information automatically. This initiative has been adopted in many research projects aiming at structuring Web contents. Wikipedia, the open gigantic source of knowledge publicly available on the Web, has been targeted by researchers to automatically extract its content and make it structured and machine-understandable. DBpedia [10] is one of the projects that extracts Wikipedia information in more than 100 languages, to build a structured multilingual and a multi-domain knowledge base.

Every Wikipedia article is transformed into a DBpedia resource, annotated with a set of properties extracted from the article page. Every resource is represented with unique, yet Uniform Resource Identifier (URI), to avoid ambiguity. Resources URIs are identified with the URI http://dbpedia.org/resource/.
Wikipedia categories are extracted and transformed into a semantically related categorization mechanism organized by SKOS \(^3\) and used to classify DBpedia resources. All extracted information is added to the knowledge base as RDF triples. DBpedia resources are already interlinked with more than thirty external datasets for various purposes, making it as a central nucleus for the Web of Data. For instance, to improve resources classification quality, YAGO2 \(^1\), UMBEL \(^4\) and WordNet \(^6\) have been introduced as typing schemas.

DBpedia project maintains also a community driven effort to develop a consistent ontology based on Wikipedia. The resulting ontology consists of 529 classes, 927 object properties, 1290 datatype properties and 116 specialized datatype properties.

DBpedia knowledge base is reachable via different access mechanisms; Linked Data principles \(^5\) which relies on HTTP and URL to retrieve resource descriptions, SPARQL endpoints which allow the execution of SPARQL queries over DBpedia knowledge base, or by downloading datasets sliced into different parts. More details about DBpedia project can be found in \(^10\).

In our work, we propose the use of DBpedia instead of single-domain ontologies to annotate mobile service descriptions in order to develop effective and universal semantic service discovery solution. We are driven by the following motivations:

- DBpedia covers multiple domains containing a large number of concepts and resources that allow annotations of a wide range of services in different domains. Each resource is referred with a unique URI which allows a non-ambiguous services annotation. In addition, each resource contains textual description (titles and abstracts) which helps in choosing the right service annotation.
- Contributors are constantly updating Wikipedia, either by adding new information or by modifying existing content. This will have an impact on DBpedia as well, since it depends on Wikipedia, and thus covering new resources and domains. Furthermore, DBpedia communities are actively working to improve DBpedia data quality.
- DBpedia resources are classified within different classification schemas, such as DBpedia Ontology, YAGO2 \(^1\), UMBEL, FreeBase.com, WordNet, etc. These hand-crafted ontologies provide highly precise knowledge that can be used to effectively annotate and search/match mobile services from different domains.
- DBpedia categories are organized and semantically related by SKOS vocabulary. They can be used to semantically arrange mobile services for effective retrieval.
- DBpedia extracts facts from Wikipedia, covering content in more than 100 languages. Thus, DBpedia is a multilingual knowledge base that we can use to develop a discovery algorithm supporting different languages.
- A set of tools are built around DBpedia knowledge base. DBpedia Spotlight \(^6\) is an automatic configurable annotation system for text documents, which detects DBpedia resources in a given text. We reckon that using this system to guide service providers in a semi-automatic or even automatic semantic annotation process of their services will lead to increased simplicity and flexibility of service provisioning. On the other hand, DBpedia lookup \(^7\) is a tool that provides resources related to a specific keyword. This tool helps users to specify their request in terms of keywords (which will be converted into corresponding resources) instead of diving into semantic languages details.
- Two kinds of ontologies exist already for semantic annotations: heavyweight and lightweight ontologies. Heavyweight ones are generally used to model complex and knowledge intensive domains such as biology or astronomy. Lightweight ones on the other hand are used for data integration or semantic search. DBpedia provides the second type of ontologies which usually lack expressiveness, may contain errors and induce omissions. However, they could fit specific applications such as semantic search where lack of expressiveness is tolerated for the benefit of scalability \(^27\).
- Due to performance and network communication evolutions, mobile user can not only act as consumers but also as service providers \(^28\). To ensure user-centric services consumption and provisioning, we believe that using lightweight ontologies and especially those provided by DBpedia, will enable this perspective, because users are more familiar with encyclopedic topics that they will use to annotate or discover services.

Listing 2 shows a set of mobile services that provide capabilities annotated using DBpedia resources, categories and YAGO concepts as well. Service1 is a concrete service that provides two capabilities, a medical clinic and biopsy. Each of which are annotated with two DBpedia resources: Clinic and Biopsy, with their associated categories: Biopsy, Clinics, Types of healthcare facilities. Service2 is also a concrete service that offers certain investigating facilities (medical tests) available in a given hospital. It is annotated with two DBpedia resources: Hospital and Medical_test in addition to one YAGO concept MedicalTests and set of categories.

Based on these DBpedia annotations, we have developed a semantic matchmaking algorithm which is discussed in Section 7.

---

3 Simple Knowledge Organization System, https://www.w3.org/2004/02/skos/.
4 http://umbel.org/.
5 http://5stardata.info/en/.

Please cite this article as: N. Cheniki et al., Mobile services discovery framework using DBpedia and non-monotonic rules, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
6. Non-monotonic semantic rules for mobile service discovery

In its mission to structure Web content and transform it into a Web of Data, W3C has achieved significant progress by standardizing RDF which allow the description of Web content and RDFS/OWL that is used to model knowledge of particular domains.

In order to balance expressive power and computation complexity, ontologies were introduced in the semantic Web, basically as restricted fragments of first order logic. To overcome this limitation while enhancing the representation capabilities and enabling powerful inferences on top of formal ontologies [29], semantic rules were introduced.

Rules in general have the form of conditions followed by conclusion, defined in two parts: antecedents and consequents. If all conditions in the antecedent clause are determined to be true, then all conclusions in the consequent clauses are implied or applied. The following example illustrates how the relation Uncle is represented with rules.

\[
\text{brother}(X, Y), \text{childOf}(Z, Y) \rightarrow \text{uncle}(X, Z)
\]

Two kinds of logic rules exist: monotonic and non-monotonic. Monotonic rules consider that conclusions remain valid even when the knowledge base is updated with additional information, which is not the case for non-monotonic rules. Non-monotonic rules are based on non-monotonic logic [30] that allow for reasoning with incomplete and conflicting information.

By combining DATALOG with parts of OWL, a semantic Rules definition Language (SWRL) is proposed to enable the description of Horn-Like rules extending syntax and semantics of OWL [31]. However, SWRL remains monotonic and has been extended [34] to support non-monotonicity and allow the definition of more expressive rules. This work empowers SWRL with four new operators: Not operator to express negative facts; NotExists quantifier to ask for missing facts in the knowledge base (when used in the antecedent of the rule) and remove facts (when used in the consequent); Dominance operator to establish priorities among rules; and Mutex operator to establish exclusions during rule executions.

In our work, we propose to use non-monotonic semantic rules for the following reasons:

- Rules reason on top of ontologies; new knowledge (in our case context and QoS information) is being inferred.
• Mobility and dynamicity of mobile users in mobile environments may lead to incomplete and/or ambiguous context information generally gathered from sensors [32]. Non-monotonic rules have the ability to deal with situations where incomplete knowledge may raise inconsistency and conflicts.

• There are numerous potential applications of non-monotonicity including personalization, brokering and bargaining [33], allowing users to express alternative choices and select the most appropriate services, thanks to the priority ranking of rules.

7. Proposed semantic discovery framework

In order to perform effective semantic mobile services discovery based on DBpedia and non-monotonic rules, we have proposed the following architecture (illustrated in Fig. 2). It is divided into three different parts namely; service provisioning, service request and service matchmaking, where each part is composed from a set of different components, the role of each one is described next.

7.1. Services provisioning manager

This component guides service providers in their provisioning process. This component guides service providers in their provisioning process. Service can typically be described with textual descriptions (in case of Concrete services) or with WSDL, OWL-S formats (in case of software services). For the second case this component tried to extract useful phrases or keywords describing the services. Textual description or extracted phrases/keywords are then passed to DBpedia Spotlight which returns their corresponding candidate annotations (in form of DBpedia resources), so the service provider chooses appropriate annotations that fit his service description. After that, the component gets all related information of selected annotations; such as concepts and categories hierarchy from DBpedia knowledge base, to store them into a local store (DBpedia cache).
7.2. Request handler

To ensure user-centric search approach and to overcome the problem of limited input capabilities of mobile devices, a user specifies his queries by typing keywords, where each keyword describes a desired service functionality or capability. Then, he chooses and edits semantic rules that specify his context and QoS requirements. A mobile application guides the user to accomplish the above task. The mobile application also gets last fresh context information from user device to generate appropriate user request which is sent to the “Request Handler” component. This component is responsible of parsing user request to extract keywords, in addition to rank rules with fresh contextual information, in order to deliver each extracted information to the appropriate component before starting the discovery process.

7.3. Keyword/resource translator

This component receives extracted keywords from “Request Handler” and translates them into corresponding DBpedia resources using DBpedia Lookup service or by issuing a SPARQL query on DBpedia cached data (using LOD manager component). Keywords can be specified in user language before their translation into equivalent English-based DBpedia resources. An additional language processing task may be performed (for example, by consulting dictionaries to get keyword synonyms), if no corresponding resource were found.

7.4. Services matchmaker component

The “Services Matchmaker” component receives user query in the form of resources from Keyword/Resource Translator component and attempts to match the query to find semantically related mobile services. **Algorithm 1** illustrates the matching process. It acquires an input list DBpedia resources describing desired capabilities, in order to output a list of semantically matched services. Each matched service has a semantic score value indicating its semantic relevance to the user’s request.

**Algorithm 1**: Semantic matching of Mobile services function.

<table>
<thead>
<tr>
<th>Input:</th>
<th>A set DBpedia resources $R_q$ each of which represents a user desired capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>set of matched services $S'$ ⊆ S</td>
</tr>
<tr>
<td>Function</td>
<td>$\text{Match}(R_q)$</td>
</tr>
<tr>
<td>1</td>
<td>$S = \text{get_all_services_by_annotation}(R_q)$</td>
</tr>
<tr>
<td>2</td>
<td>foreach $s \in S$ do</td>
</tr>
<tr>
<td>3</td>
<td>scores = 0</td>
</tr>
<tr>
<td>4</td>
<td>$R_i = \text{get_annotation_resources}(s)$ // return all resources used to annotate service $s$ capabilities</td>
</tr>
<tr>
<td>5</td>
<td>foreach $r' \in R_i$ do</td>
</tr>
<tr>
<td>6</td>
<td>foreach $r \in R_q$ do</td>
</tr>
<tr>
<td>7</td>
<td>scores += similarity($r$, $r'$)</td>
</tr>
<tr>
<td>8</td>
<td>end</td>
</tr>
<tr>
<td>9</td>
<td>end</td>
</tr>
<tr>
<td>10</td>
<td>if scores &gt; 0 then</td>
</tr>
<tr>
<td>11</td>
<td>s.semanticScore = $\log(1 + \text{scores})$</td>
</tr>
<tr>
<td>12</td>
<td>$S' = S' \cup {s}$</td>
</tr>
<tr>
<td>13</td>
<td>end</td>
</tr>
<tr>
<td>14</td>
<td>return $S'$</td>
</tr>
</tbody>
</table>

**Algorithm 1** starts by iterating through all services annotated by resources or categories from user’s request (retrieved using function $\text{get\_all\_services\_by\_annotations}$). The semantic score of each retrieved service is calculated using $\text{similarity}$ function. For each service, we get its capabilities resources, the function $\text{similarity}$ is used to calculate degree of semantic relevance between each service resource (from the set $R_i$) and request resource (from the set $R_q$). If returned score from $\text{similarity}$ function is positive, we apply a log to the final score to alleviate the score value.

Similarity function (**Algorithm 2**) calculates the semantic relevance between two resources based on Tversky model [34]. For the two compared resources, the function $\text{get\_concepts}$ is used to get their concepts, and the function $\text{get\_categories}$ returns the set of their categories (with regard to limited level of hierarchy $\ell$, since categories hierarchy form cycles). Then, the function $\text{Tversky}$ (**Algorithm 3**) is called to calculate the degree of relevance by computing commonalities and differences within the compared sets. The final score of the function $\text{similarity}$ is obtained by computing the average between concepts and categories matching scores.

We indicate that function $\text{get\_categories}$ returns super and sub categories of a given resource $r$, so they have the same importance during matchmaking process.

Please cite this article as: N. Cheniki et al., Mobile services discovery framework using DBpedia and non-monotonic rules, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
Algorithm 2: similarity function.

\[ \text{Input}: \text{Two DBpedia resources } r \text{ and } r' \]
\[ \text{Output}: \text{a similarity score indicating semantic relevance between the two resources } r \text{ and } r' \]

1. \textbf{Function} Similarity\((r, r')\)
2. \(\text{score} = 0\)
3. \(T_r = \text{get_concepts}(r)\) // returns ontology concepts that instantiate the resource \(r\)
4. \(T_{r'} = \text{get_concepts}(r')\)
5. \(C_r = \text{get_categories}(r, \ell)\) // return categories that classify the resource \(r\).
6. \(C_{r'} = \text{get_categories}(r', \ell)\) // in respect of limited number of hierarchy level \(\ell\)
7. \(S_{\text{concepts}} = \text{Tversky}(T_r, T_{r'})\)
8. \(S_{\text{categories}} = \text{Tversky}(C_r, C_{r'})\)
9. \(\text{score} = \text{AVG}(S_{\text{concepts}}, S_{\text{categories}})\)
10. \textbf{return} \(\text{score}\)

Algorithm 3: Tversky function.

\[ \text{Input}: \text{Two lists of features } F_1 \text{ and } F_2 \]
\[ \text{Output}: \text{a similarity value sim between 0 and 1} \]

1. \textbf{Function} Tversky\((r, s)\)
2. \(C = \text{commonElements}(F_1, F_2)\) // returns elements that are shared between the two sets
3. \(U_{F_1} = \text{uniqueElements}(F_1, F_2)\) // returns elements that belong to the first set and not to the second
4. \(U_{F_2} = \text{uniqueElements}(F_2, F_1)\)
5. \(\text{sim} = \frac{|C|}{|C| + |U_{F_1}| + |U_{F_2}|}\)
6. \textbf{return} \(\text{sim}\)

7.5. LOD manager component

The Linked Open Data Manager requests, filters and stores information about required resources from DBpedia. Matchmaking process based only on SPARQL queries usually gives rise to performance issues. One effective way to reduce the impact of heavy SPARQL queries is indexing services based on their annotation resources, concepts and categories. Afterwards, indexes are pre-loaded on the main memory to ensure scalable and effective matchmaking of mobile services.

7.6. Services ranking component

“Services Matchmaker” component produces a list of relevant services that match user needs. However, some of these services may not satisfy user context and QoS requirements. Therefore, we propose to rank matched services based on non-monotonic rules (introduced in Section 6), describing alternative choices of services based on context and QoS information. This component uses a rules reasoner engine that provides monotonic and non-monotonic rules inference service. It also uses context and QoS information manager to get required user context with QoS information values involved in ranking rules. So, this component starts by extracting required information from context and QoS database before execution of user provided rules. A set of predefined ranking rules can be offered to the user (for example rules that privilege nearest services), with the ability to customize and edit them to fit specific needs. It is also possible to allow users to exchange the rules created using adequate user interface.

Let us illustrate the following rules describing desired user ranking choices. Consider three rules \(r_1, r_2\) and \(r_3\) where \(r_1 > r_2, r_1 > r_3\), i.e. the first rule has high priority over \(r_2\) and \(r_3\) (\(r_1\) must be executed before \(r_2\) and \(r_3\)). The rule \(r_1\) eliminates any service that has price greater than 300, so rule \(r_2\) and \(r_3\) will be fired only on services satisfying \(r_1\). Rule \(r_2\) gives two points rank to any service priced between 70 and 150, while \(r_3\) gives only one point to services having price greater than 150.

\[ r_1: \text{Service}(s), \text{hasQoSInformation}(s, \text{QoSInfo}), \text{hasPrice}(\text{QoSInfo}, \text{price}), \text{price} > 150 \rightarrow \text{Eliminated}(s, \text{true}) \]
\[ r_2: \text{Service}(s), \text{Eliminated}(s, \text{false}), \text{hasQoSInformation}(s, \text{QoSInfo}), \text{hasPrice}(\text{QoSInfo}, \text{price}), \text{price} > 70, \text{price} < 150 \rightarrow \text{Rank}(s, 2) \]
\[ r_3: \text{Service}(s), \text{Eliminated}(s, \text{false}), \text{hasQoSInformation}(s, \text{QoSInfo}), \text{hasPrice}(\text{QoSInfo}, \text{price}), \text{price} > 150 \rightarrow \text{Rank}(s, 1) \]
It is worth mentioning that rules can be used to automatically eliminate services that impose mandatory requirements. For instance, if a user’s device does not support NFC (Near field communication) and a particular service requires this feature, it should be removed from resulting ranked service list.

7.7. Rules reasoner engine

Rules reasoner component provides semantic reasoning services to facilitate execution of rules for “Ranking component” and “Context and QoS manager”. It deduces and returns new context and/or QoS information by executing received semantic monotonic/non-monotonic rules.

7.8. Context and QoS manager

Context and QoS manager maintains users and services contextual and quality of service information. It not only collects, but also stores and updates context information in databases, in order to be retrieved as needed. In fact, this information is stored into an RDF store, since we have used ontology models. This component uses rules engine to reason upon context and QoS information in order to infer new knowledge. For example, if a service defines specific providing days, context and QoS manager can decide that this service is not available out of defined range, so it might get ignored during discovery process.

8. Implementation and evaluation

We have implemented the above framework focusing on the discovery process. It is worth noting that multilingual discovery is out of the scope of this paper. Interested reader may refer to [35] for more details about supporting multilingual service discovery. Each component of the framework was created as a separate module. Jena 8 is the core library for the framework. It is used to request Information about resources, categories and concepts via DBpedia SPARQL endpoints. 9. Service descriptions, context and QoS information are stored in a local Jena TDB RDF store, and retrieved using Jena ARQ and SPARQL. 11. Jena has a generic rules engine that we have used in conjunction with JNOMO 10 which extends Jena to support non-monotonic rules execution.

In the following section, we describe used test collection and how services are annotated. Then, we present comparative experimental results shedding light on how our framework improves discovery effectiveness of mobile services. We conclude this section with performance analysis.

8.1. Semantic matchmaking effectiveness

8.1.1. Test collection and semantic services annotation

To evaluate semantic matchmaking effectiveness, we rely on data provided by OWL-S TC 11 version 4. OWL-S TC includes a set of 1083OWL-S 1.1 semantic services from nine different domains (education, medical care, food, travel, communication, economy, weapons, geography and simulation). Also, it provides 42 test queries associated with binary and graded relevance sets to conduct performance evaluation experiments. In addition, we collect a part of our test collection set from online public repositories such as http://www.webapifinder.com/ and http://www.programmableweb.com.

Our collection set is made from 170 services; 123 out of which have been transformed from the OWL-S TC collection and 47 are accumulated from public repositories. Services are classified in the categories: food, medical care, travel, communication, economy, e-mailing and question/answer.

We annotate services by passing each service textual description to DBpedia Spotlight that returns annotation candidates for the service. Then, we pick out the best candidates that judge relevant to the service description. In some annotation cases, DBpedia Spotlight is incapable to retrieve the appropriate annotation resources. So, we try to get missed annotations by searching on DBpedia knowledge base.

During the annotation process we noticed that DBpedia is able to annotate almost all services. However, some concepts are missing for the annotation; for example PREPAREDFOOD concept do not have a corresponding resource in DBpedia. On the other hand, there were cases where annotation of OWL-S TC services could be improved; for instance in food_price_AnimalFoodservice.owl service description, food is used to annotate the input, where it could be annotated by Animal_feed resource from DBpedia.

In total, we’ve used 150 resources, 249 concepts and 38,904 categories (with all their related, super and sub categories) to annotate all services. It is obvious that categories are dominating references since almost every resource is classified into one or more categories. We noticed that concepts are less used compared to resources and categories because not all resources are instantiated with ontological concepts. The majority of concepts are from YAGO ontology as shown in Fig. 3.

---

9 http://dbpedia.org/sparql.
10 http://sourceforge.net/projects/jnomo/.

Please cite this article as: N. Cheniki et al., Mobile services discovery framework using DBpedia and non-monotonic rules, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
8.1.2. Discovery effectiveness criteria

To evaluate the accuracy of our discovery framework, we use precision and recall. Precision is the ratio of the number of relevant services retrieved to the total number of retrieved services. Recall is defined as the ratio of the number of relevant services retrieved to the total number of relevant services in the services directory. Their mathematical definitions are as follows:

\[
\text{Precision} = \frac{|\{\text{relevant retrieved services}\}|}{|\{\text{retrieved services}\}|}
\]

\[
\text{Recall} = \frac{|\{\text{relevant retrieved services}\}|}{|\{\text{all relevant services}\}|}
\]

In order to evaluate returned ranked services and make comparison between different retrieval systems, 11-point precision-recall curve is used. This graph plots the interpolated precision of an information retrieval (IR) system at 11 standard recall levels. If we have a system that contains n services, we calculate precision and recall for each top k retrieved service, where \(k \in [1, n]\). Then, we plot interpolated precision for each recall r level where \(r \in \{0.0, 0.1, 0.2, ..., 1.0\}\). The interpolated precision \(P_{\text{interp}}\) at recall level \(r\) is defined as the highest precision for any recall level \(r' \geq r\):

\[
P_{\text{interp}}(r) = \max_{r' \geq r} P(r')
\]

8.1.3. Evaluation results

To measure services retrieval effectiveness, we have calculated interpolated precision of 22 distinct requests over the 170 annotated services taking into consideration the top 20 results (\(k = 20\)). We execute each request for three different category levels. Then, we compare results with keyword-based search performed by Apache Lucene\(^{12}\) which is an open source tool that provides a Java based high-performance, full-featured text search engine library. Apache Lucene provides TF-IDF (Term Frequency-Inverse Document Frequency) which is scoring model to calculate documents relevance regarding a user query.

We rely on OWL-S TC binary relevance set to classify services as relevant or irrelevant, or we judge relevancy by asking whether we would be able to directly use the service to obtain at least part of the information or functionality that the request is interested in.

Fig. 4 illustrates obtained results plotted in 11-point curve. It shows that semantic search with different categories possess a higher result over keyword-based search. On other hand, using more categories hierarchy enables us to get more relevant services. We have performed semantic search for maximum three levels, the curve shows that incorporating more levels affect negatively on the result’s relevancy, because higher levels may include more generic and largely shared categories (level two in some cases overpass the third one).

Despite semantic search gives values that are highly relevant overall, we found in some queries that keyword search gives higher relevance due to the miss-instantiation or the bad classification of some DBpedia resources. Experiments show that proposed framework is able to retrieve semantically related services. For instance, when a user searches for a hospital service, other semantically closed services such as clinic and care organization appear in the search results which is not the case for keyword based search.

the semi relation between resources, the typing ...

8.2. Performance

In this subsection, we present performance evaluation results for services matchmaking and rules execution. Experiments were run on a machine equipped with Ubuntu 14.04 64 bit operating system, Java 7, Intel i7 2 GHz x 4 processor and 8GB RAM.

8.2.1. Matchmaking scalability

The experiments were conducted by executing 22 requests on Web services collection presented above; the average execution time of all request is then computed. For each DBpedia resource used to annotate services, we locally store all the related categories and concepts including their hierarchies. We’ve also indexed the services with their semantic data retrieved from DBpedia, to ensure discovery scalability as will be discussed in the followings.

At first, we tested matchmaking algorithm using only SPARQL queries. The analysis of the results (Fig. 5) reveals that matchmaking services is a time consuming process. It takes approximately 1.7 to 47 s depending on the number of used resources, and categories hierarchy levels. It’s obvious that the more we involve categories from upper and lower levels, the more time does the matching process take.

Please cite this article as: N. Cheniki et al., Mobile services discovery framework using DBpedia and non-monotonic rules, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
Thus, an indexing strategy\textsuperscript{13} of services matchmaking is crucial to ensure discovery scalability. We’ve indexed all services based on their resources, concepts and categories, in addition to all semantic DBpedia data used for the annotation. Afterwards, we re-tested the matchmaking, relying on the locally generated indexes that are pre-populated at load time, so no need to issue SPARQL queries to the local RDF store. This strategy shows a considerable gain in terms of execution time compared to SPARQL-based matchmaking. As it is shown in Fig. 6, it takes only 90 ms to 1.2 s. However, considering high level of categories ($\ell = 3$) still has a negative impact on matchmaking time while there is no considerable gain in the resulted quality (as we’ve stated before). As a consequence, we suggest limiting number of categories level to only two levels.

8.2.2. Non-monotonic context rules execution performance

To evaluate performance execution of non-monotonic rules describing users choices, we have executed a set of 10 rules containing more than 50 clauses. We ensure that 90% of the rules should be fired. Fig. 7 shows that rules execution time is acceptable, since number of candidate services for ranking process is decreased by semantic services matchmaking component.

\textsuperscript{13} We’ve used MapDB to implement the indexing strategy, http://mapdb.org/.

Please cite this article as: N. Cheniki et al., Mobile services discovery framework using DBpedia and non-monotonic rules, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.02.016
9. Conclusion and future work

Services discovery is an essential aspect to correlate both service providers and users. In mobile environments, developing an effective service discovery becomes a more challenging issue; while context and QoS information may provide distinct results. To respond to these requirements, we have proposed in this paper, an extended ontology called “MOBISO” used to describe context and QoS information of mobile services. To overcome keyword-based search limitations; a semantic discovery framework based on DBpedia, a multi-domain multilingual knowledge base, and on non-monotonic rules has been proposed. Our discovery framework exploits mobile services descriptions annotated with DBpedia resource, concepts and categories to retrieve semantically related services, each of which are ordered by their semantic relevance to user request. Then, it applies a ranking process based on contextual and QoS information defined through non-monotonic rules. Evaluation results of our discovery approach show that services search effectiveness is considerably ameliorated compared to keyword-based search. Besides, semantic rules provided a flexible mechanism, allowing users to filter and rank services according to their contextual and QoS needs. Our solution focuses on user-centricity; trying to take user away from details of semantic Web technologies, thanks to DBpedia and its tools such as DBpedia Lookup and DBpedia Spotlight.

As future work, we plan to deploy our framework to the cloud as it provides necessary computation resource to ensure scalable and reliable service discovery especially for semantic-based approaches. Moreover, we are convinced that combining syntactic search with semantic search will improve our proposed semantic service search. We plan to use more linked data to improve the discovery mechanism of DBpedia resources since some of them are not well described with adequate ontology concepts. We believe that including more annotation sources by consuming Linked Data, will improve discovery effectiveness.

References

Nasredine Cheniki received the Bachelor’s degree with distinction in computer systems from the university of 20 Août 1955-Skikda in 2009, and the Master’s degree in 2011. He is currently a PhD student in the Computer Science Department at the University of Sciences and Technology Houari Boumediene (USTHB) Algiers, where he is doing work in the area of mobile services discovery.

Abdelkader Belkhir is a full professor at the computer science department, faculty of electronic and computer science, USTHB University, Algiers. He leads a research team whose theme is web technology and computer security (LSI laboratory). His research field is web technology and computer security.

Yacine Atif works at Skvde University in Sweden. Earlier, he worked at Purdue University in the USA as a Post-Doc and then joined a faculty position at Nanyang Technological University (NTU) in Singapore. From 1999 to 2015 he worked at UAE University. He is an active researcher in research areas such as Semantic Web, Social Web, Mobile Web and the Web of Things.