Bootstrapping Ontologies for Web Services

Aviv Segev, *Member*, *IEEE*, and Quan Z. Sheng, *Member*, *IEEE*

Abstract—Ontologies have become the de-facto modeling tool of choice, employed in many applications and prominently in the semantic web. Nevertheless, ontology construction remains a daunting task. Ontological bootstrapping, which aims at automatically generating concepts and their relations in a given domain, is a promising technique for ontology construction. Bootstrapping an ontology based on a set of predefined textual sources, such as web services, must address the problem of multiple, largely unrelated concepts. In this paper, we propose an ontology bootstrapping process for web services. We exploit the advantage that web services usually consist of both WSDL and free text descriptors. The WSDL descriptor is evaluated using two methods, namely Term Frequency/Inverse Document Frequency (TF/IDF) and web context generation. Our proposed ontology bootstrapping process integrates the results of both methods and applies a third method to validate the concepts using the service free text descriptor, thereby offering a more accurate definition of ontologies. We extensively validated our bootstrapping method using a large repository of real-world web services and verified the results against existing ontologies. The experimental results indicate high precision. Furthermore, the recall versus precision comparison of the results when each method is separately implemented presents the advantage of our integrated bootstrapping approach.

Index Terms—Web services discovery, metadata of services interfaces, service-oriented relationship modeling.

1 Introduction

Ontologies are used in an increasing range of applications, notably the Semantic web, and essentially have become the preferred modeling tool. However, the design and maintenance of ontologies is a formidable process [1], [2]. Ontology bootstrapping, which has recently emerged as an important technology for ontology construction, involves automatic identification of concepts relevant to a domain and relations between the concepts [3].

Previous work on ontology bootstrapping focused on either a limited domain [4] or expanding an existing ontology [5]. In the field of web services, registries such as the Universal Description, Discovery and Integration (UDDI) have been created to encourage interoperability and adoption of web services. Unfortunately, UDDI registries have some major flaws [6]. In particular, UDDI registries either are publicly available and contain many obsolete entries or require registration that limits access. In either case, a registry only stores a limited description of the available services. Ontologies created for classifying and utilizing web services can serve as an alternative solution. However, the increasing number of available web services makes it difficult to classify web services using a single domain ontology or a set of existing ontologies created for other purposes. Furthermore, constant increase in the number of web services requires continuous manual effort to evolve an ontology.

The web service ontology bootstrapping process proposed in this paper is based on the advantage that a web service can be separated into two types of descriptions: 1) the Web Service Description Language (WSDL) describing "how" the service should be used and 2) a textual description of the web service in free text describing "what" the service does. This advantage allows bootstrapping the ontology based on WSDL and verifying the process based on the web service free text descriptor.

The ontology bootstrapping process is based on analyzing a web service using three different methods, where each method represents a different perspective of viewing the web service. As a result, the process provides a more accurate definition of the ontology and yields better results. In particular, the Term Frequency/Inverse Document Frequency (TF/IDF) method analyzes the web service from an internal point of view, i.e., what concept in the text best describes the WSDL document content. The Web Context Extraction method describes the WSDL document from an external point of view, i.e., what most common concept represents the answers to the web search queries based on the WSDL content. Finally, the Free Text Description Verification method is used to resolve inconsistencies with the current ontology. An ontology evolution is performed when all three analysis methods agree on the identification of a new concept or a relation change between the ontology concepts. The relation between two concepts is defined using the descriptors related to both concepts. Our approach can assist in ontology construction and reduce the maintenance effort substantially. The approach facilitates automatic building of an ontology that can assist in expanding, classifying, and retrieving relevant services, without the prior training required by previously developed approaches.

We conducted a number of experiments by analyzing 392 real-world web services from various domains. In particular, the first set of experiments compared the precision of the concepts generated by different methods. Each method supplied a list of concepts that were analyzed to evaluate how many of them are meaningful and could be related to the services. The second set of experiments compared the recall

- *A. Segev is with the Department of Knowledge Service Engineering, KAIST, Daejeon 305-701, Korea. E-mail: aviv@kaist.edu.*
- *Q.Z. Sheng is with the School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia. E-mail: qsheng@cs.adelaide.edu.au.*

Manuscript received 24 Dec. 2009; revised 23 Mar. 2010; accepted 27 May 2010; published online 14 Dec. 2010.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2009-12-0218.
Digital Object Identifier no. 10.1109/TSC.2010.51.

of the concepts generated by the methods. The list of concepts was used to analyze how many of the web services could be classified by the concepts. The recall and precision of our approach was compared with the performance of Term Frequency/Inverse Document Frequency and web based concept generation. The results indicate higher precision of our approach compared to other methods. We also conducted experiments comparing the concept relations generated from different methods. The analysis used the Swoogle ontology search engine [7] to verify the results.

The main contributions of this work are as follows:

- On a conceptual level, we introduce an ontology bootstrapping model, a model for automatically creating the concepts and relations "from scratch."
- On an algorithmic level, we provide an implementation of the model in the web service domain using integration of two methods for implementing the ontology construction and a Free Text Description Verification method for validation using a different source of information.
- On a practical level, we validated the feasibility and benefits of our approach using a set of real-world web services. Given that the task of designing and maintaining ontologies is still difficult, our approach presented in this paper can be valuable in practice.

The remainder of the paper is organized as follows: Section 2 discusses the related work. Section 3 describes the bootstrapping ontology model and illustrates each step of the bootstrapping process using an example. Section 4 presents experimental results of our proposed approach. Section 5 further discusses the model and the results. Finally, Section 6 provides some concluding remarks.

2 Related Work

2.1 Web Service Annotation

The field of automatic annotation of web services contains several works relevant to our research. Patil et al. [8] present a combined approach toward automatic semantic annotation of web services. The approach relies on several matchers (e.g., string matcher, structural matcher, and synonym finder), which are combined using a simple aggregation function. Chabeb et al. [9] describe a technique for performing semantic annotation on web services and integrating the results into WSDL. Duo et al. [10] present a similar approach, which also aggregates results from several matchers. Oldham et al. [11] use a simple machine learning (ML) technique, namely Naïve Bayesian Classifier, to improve the precision of service annotation. Machine learning is also used in a tool called Assam [12], which uses existing annotation of semantic web services to improve new annotations. Categorizing and matching web service against existing ontology was proposed by [13]. A context-based semantic approach to the problem of matching and ranking web services for possible service composition is suggested in [14]. Unfortunately, all these approaches require clear and formal semantic mapping to existing ontologies.

2.2 Ontology Creation and Evolution

Recent work has focused on ontology creation and evolution and in particular on schema matching. Many heuristics were proposed for the automatic matching of schemata (e.g., Cupid [15], GLUE [16], and OntoBuilder [17]), and several theoretical models were proposed to represent various aspects of the matching process such as representation of mappings between ontologies [18], ontology matching using upper ontologies [19], and modeling and evaluating automatic semantic reconciliation [20]. However, all the methodologies described require comparison between existing ontologies.

The realm of information science has produced an extensive body of literature and practice in ontology construction, e.g., [21]. Other undertakings, such as the DOGMA project [22], provide an engineering approach to ontology management. Work has been done in ontology learning, such as Text-To-Onto [23], Thematic Mapping [24], and TexaMiner [25] to name a few. Finally, researchers in the field of knowledge representation have studied ontology interoperability, resulting in systems such as Chimaera [26] and Protègè [27] . The works described are limited to ontology management that involves manual assistance to the ontology construction process.

Ontology evolution has been researched on domain specific websites [28] and digital library collections [4]. A bootstrapping approach to knowledge acquisition in the fields of visual media [29] and multimedia [5] uses existing ontologies for ontology evolution. Another perspective focuses on reusing ontologies and language components for ontology generation [30]. Noy and Klein [1] defined a set of ontology-change operations and their effects on instance data used during the ontology evolution process. Unlike previous work, which was heavily based on existing ontology or domain specific, our work automatically evolves an ontology for web services from the beginning.

2.3 Ontology Evolution of Web Services

Surveys on ontology techniques implementations to the semantic web [31] and service discovery approaches [32] suggest ontology evolution as one of the future directions of research. Ontology learning tools for semantic web service descriptions have been developed based on Natural Language Processing (NLP) [33]. Their work mentions the importance of further research concentrating on context directed ontology learning in order to overcome the limitations of NLP. In addition, a survey on the state-of-the-art web service repositories [34] suggests that analyzing the web service textual description in addition to the WSDL description can be more useful than analyzing each descriptor separately. The survey mentions the limitation of existing ontology evolution techniques that yield low recall. Our solution overcomes the low recall by using web context recognition.

3 The Bootstrapping Ontology Model

The bootstrapping ontology model proposed in this paper is based on the continuous analysis of WSDL documents and employs an ontology model based on concepts and relationships [35]. The innovation of the proposed bootstrapping model centers on 1) the combination of the use of two different extraction methods, TF/IDF and web based concept generation, and 2) the verification of the results using a Free Text Description Verification method by analyzing the

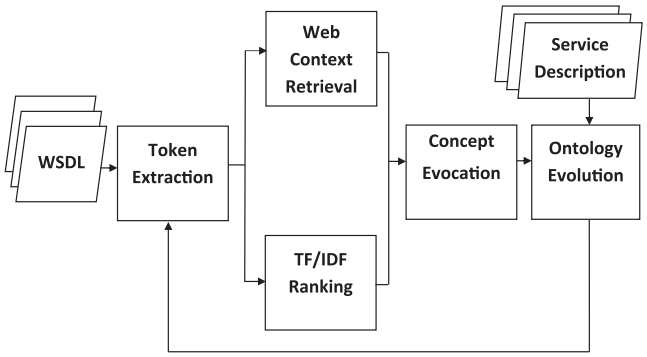

Fig. 1. Web service ontology bootstrapping process.

external service descriptor. We utilize these three methods to demonstrate the feasibility of our model. It should be noted that other more complex methods, from the field of Machine Learning (ML) and Information Retrieval (IR), can also be used to implement the model. However, the use of the methods in a straightforward manner emphasizes that many methods can be "plugged in" and that the results are attributed to the model's process of combination and verification. Our model integrates these three specific methods since each method presents a unique advantage—internal perspective of the web service by the TF/IDF, external perspective of the web service by the Web Context Extraction, and a comparison to a free text description, a manual evaluation of the results, for verification purposes.

3.1 An Overview of the Bootstrapping Process

The overall bootstrapping ontology process is described in Fig. 1. There are four main steps in the process. The *token extraction* step extracts tokens representing relevant information from a WSDL document. This step extracts all the *name* labels, parses the tokens, and performs initial filtering.

The second step analyzes in parallel the extracted WSDL tokens using two methods. In particular, *TF/IDF* analyzes the most common terms appearing in each web service document and appearing less frequently in other documents. *Web Context Extraction* uses the sets of tokens as a query to a search engine, clusters the results according to textual descriptors, and classifies which set of descriptors identifies the context of the web service.

The *concept evocation* step identifies the descriptors which appear in both the TF/IDF method and the web context method. These descriptors identify possible concept names that could be utilized by the ontology evolution. The context descriptors also assist in the convergence process of the relations between concepts.

Finally, the *ontology evolution* step expands the ontology as required according to the newly identified concepts and modifies the relations between them. The external web service textual descriptor serves as a moderator if there is a conflict between the current ontology and a new concept. Such conflicts may derive from the need to more accurately specify the concept or to define concept relations. New concepts can be checked against the free text descriptors to verify the correct interpretation of the concept. The relations are defined as an ongoing process according to the most common context descriptors between the concepts. After the ontology evolution, the whole process continues to the next WSDL with the evolved ontology concepts and relations. It should be noted that the processing order of WSDL documents is arbitrary.

```
<s:complexType name="Domain">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Country" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Zip" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="City" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="State" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Address" type="s:string" />
  </s:sequence>
 </s:complexType>
 <s:element name="GetDomainsByRegistrantName">
  <s:complexType>
    <s:element minOccurs="0" maxOccurs="1" name="FirstMiddleName" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="LastName" type="s:string" />
 <s:element name="GetDomainsByRegistrantNameResponse">
  <s:complexType>
    <s:element minOccurs="0" maxOccurs="1" name="GetDomainsByRegistrantNameResult"
type="s0:Domains" />
  </s:complexType>
 </s:element>
 <s:element name="Domains" nillable="true" type="s0:Domains" />
</s:schema>
<message name="GetDomainsByZipSoapIn">
```

Fig. 2. WSDL example of the service `DomainSpy`.

In the continuation, we describe each step of our approach in detail. The following three web services will be used as an example to illustrate our approach:

- `DomainSpy` is a web service that allows domain registrants to be identified by region or registrant name. It maintains an XML-based domain database with over 7 million domain registrants in the US.
- `AcademicVerifier` is a web service that determines whether an email address or domain name belongs to an academic institution.
- `ZipCodeResolver` is a web service that resolves partial US mailing addresses and returns proper ZIP Code. The service uses an XML interface.

3.2 Token Extraction

The analysis starts with token extraction, representing each service, $\mathcal{S}$, using a set of tokens called *descriptor*. Each token is a textual term, extracted by simply parsing the underlying documentation of the service. The descriptor represents the WSDL document, formally put as $\mathcal{D}^{\mathcal{S}}_{wsdl} = \{t_1, t_2, \ldots, t_n\}$, where t_i is a token. WSDL tokens require special handling, since meaningful tokens (such as names of parameters and operations) are usually composed of a sequence of words with each first letter of the words capitalized (e.g., `GetDomainsByRegistrantNameResponse`). Therefore, the descriptors are divided into separate tokens. It is worth mentioning that we initially considered using predefined WSDL documentation tags for extraction and evaluation but found them less valuable since web service developers usually do not include tags in their services.

Fig. 2 depicts a WSDL document with the token list bolded. The extracted token list serves as a *baseline*. These tokens are extracted from the WSDL document of a web service `DomainSpy`. The service is used as an initial step in our example in building the ontology. Additional services will be used later to illustrate the process of expanding the ontology.

Fig. 3. Example of the TF/IDF method results for `DomainSpy`.

All elements classified as *name* are extracted, including tokens that might be less relevant. The sequence of words is expanded as previously mentioned using the capital letter of each word. The tokens are filtered using a list of *stop-words*, removing words with no substantive semantics. Next, we describe the two methods used for the description extraction of web services: *TF/IDF* and *context extraction*.

3.3 TF/IDF Analysis

TF/IDF is a common mechanism in IR for generating a robust set of representative keywords from a corpus of documents. The method is applied here to the WSDL descriptors. By building an independent corpus for each document, irrelevant terms are more distinct and can be thrown away with a higher confidence. To formally define TF/IDF, we start by defining $freq(t_i, \mathcal{D}_i)$ as the number of occurrences of the token t_i within the document descriptor $\mathcal{D}_i$. We define the term frequency of each token t_i as

$$tf(t_i) = \frac{freq(t_i, \mathcal{D}_i)}{|\mathcal{D}_i|}. \quad (1)$$

We define $\mathcal{D}_{wsdl}$ to be the corpus of WSDL descriptors. The inverse document frequency is calculated as the ratio between the total number of documents and the number of documents that contain the term:

$$idf(t_i) = \log \frac{|\mathcal{D}|}{|\{\mathcal{D}_i \ : \ t_i \in \mathcal{D}_i\}|}. \quad (2)$$

Here, $\mathcal{D}$ is defined as a specific WSDL descriptor. The TF/IDF weight of a token, annotated as $w(t_i)$, is calculated as

$$w(t_i) = tf(t_i) \times idf^2(t_i). \quad (3)$$

While the common implementation of TF/IDF gives equal weights to the term frequency and inverse document frequency (i.e., $w = tf \times idf$), we chose to give higher weight to the *idf* value. The reason behind this modification is to normalize the inherent bias of the *tf* measure in short documents [36]. Traditional TF/IDF applications are concerned with verbose documents (e.g., books, articles, and human-readable webpages). However, WSDL documents have relatively short descriptions. Therefore, the frequency of a word within a document tends to be incidental, and the document length component of the TF generally has little or no influence.

The token weight is used to induce ranking over the descriptor's tokens. We define the ranking using a precedence relation $\preceq_{tf/idf}$, which is a partial order over $\mathcal{D}$, such that $t_l \preceq_{tf/idf} t_k$ if $w(t_l) < w(t_k)$. The ranking is used to filter the tokens according to a threshold that filters out words with a frequency count higher than the second standard deviation from the average weight of token w value. The effectiveness of the threshold was validated by our experiments. Fig. 3 presents the list of tokens that received a higher weight than the threshold for the `DomainSpy` service. Several tokens that appeared in the baseline list (see Fig. 2) were removed due to the filtering process. For instance, words such as "Response," "Result," and "Get" received below-the-threshold TF/IDF weight, due to their high IDF value.

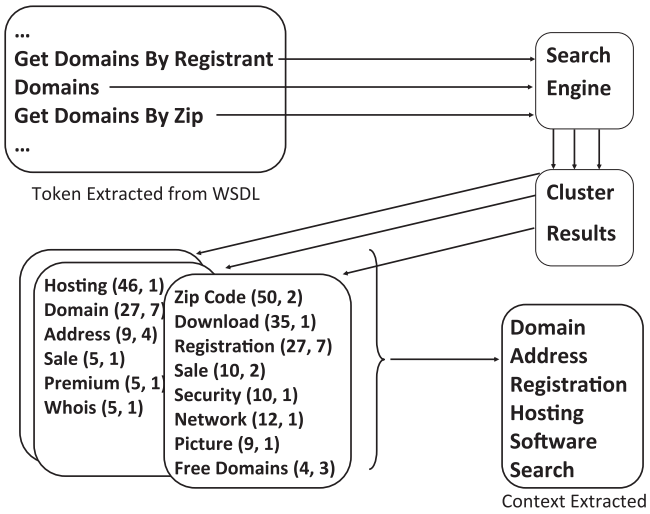

Fig. 4. Example of the context extraction method for `DomainSpy`.

3.4 Web Context Extraction

We define a context descriptor c_i from domain $\mathcal{DOM}$ as an index term used to identify a record of information [37], which in our case is a web service. It can consist of a word, phrase, or alphanumerical term. A weight $w_i \in \Re$ identifies the importance of descriptor c_i in relation to the web service. For example, we can have a descriptor $c_1 = Address$ and $w_1 = 42$. A *descriptor set* $\{\langle c_i, w_i \rangle\}_i$ is defined by a set of pairs, descriptors and weights. Each descriptor can define a different point of view of the concept. The descriptor set eventually defines all the different perspectives and their relevant weights, which identify the importance of each perspective.

By collecting all the different view points delineated by the different descriptors, we obtain the *context*. A context $\mathcal{C} = \{\{\langle c_{ij}, w_{ij} \rangle\}_i\}_j$ is a set of finite sets of descriptors, where i represents each context descriptor and j represents the index of each set. For example, a context $\mathcal{C}$ may be a set of words (hence $\mathcal{DOM}$ is a set of all possible character combinations) defining a web service and the weights can represent the relevance of a descriptor to the web service. In classic Information Retrieval, $\langle c_{ij}, w_{ij} \rangle$ may represent the fact that the word c_{ij} is repeated w_{ij} times in the web service descriptor.

The context extraction algorithm is adapted from [38]. The input of the algorithm is defined as tokens extracted from the web service WSDL descriptor (Section 3.2). The sets of tokens are extracted from elements classified as *name*, for example *Get Domains By Zip*, as described in Fig. 4. Each set of tokens is then sent to a web search engine and a set of descriptors is extracted by clustering the webpages search results for each token set.

The webpages clustering algorithm is based on the *concise all pairs profiling* (CAPP) clustering method [39]. This method approximates profiling of large classifications. It compares all classes pairwise and then minimizes the total number of features required to guarantee that each pair of classes is contrasted by at least one feature. Then each class profile is assigned its own minimized list of features, characterized by how these features differentiate the class from the other features.

Fig. 4 shows an example that presents the results for the extraction and clustering performed on tokens *Get Domains By Zip*. The context descriptors extracted include: $\{\langle ZipCode\ (50,2)\rangle, \langle Download\ (35,1)\rangle, \langle Registration\ (27,7)\rangle, \langle Sale\ (15,1)\rangle, \langle Security\ (10,1)\rangle, \langle Network\ (12,1)\rangle, \langle Picture\ (9,1)\rangle, \langle Free\ Domains(4,3)\rangle\}$. A different point of view of the concept can been seen in the previous set of tokens *Domains* where the context descriptors extracted include $\{\langle Hosting\ (46,1)\rangle, \langle Domain\ (27,7)\rangle, \langle Address\ (9,4)\rangle, \langle Sale\ (5,1)\rangle, \langle Premium\ (5,1)\rangle, \langle Whois\ (5,1)\rangle\}$. It should be noted that each descriptor is accompanied by two initial weights. The first weight represents the number of references on the web (i.e., the number of returned webpages) for that descriptor in the specific query. The second weight represents the number of references to the descriptor in the WSDL (i.e., for how many *name* token sets was the descriptor retrieved). For instance, in the above example, *Registration* appeared in 27 webpages and seven different *name* token sets in the WSDL referred to it.

The algorithm then calculates the sum of the number of webpages that identify the same descriptor and the sum of number of references to the descriptor in the WSDL. A high ranking in only one of the weights does not necessarily indicate the importance of the context descriptor. For example, high ranking in only web references may mean that the descriptor is important since the descriptor widely appears on the web, but it might not be relevant to the topic of the web service (e.g., *Download* descriptor for the *DomainSpy* web service, see Fig. 4). To combine values of both the webpage references and the appearances in the WSDL, the two values are weighted to contribute equally to final weight value.

For each descriptor, c_i, we measure how many webpages refer to it, defined by weight w_{i1}, and how many times it is referred to in the WSDL, defined by weight w_{i2}. For example, *Hosting* might not appear at all in the web service, but the descriptor based on clustered webpages could refer to it twice in the WSDL and a total of 235 webpages might be referring to it. The descriptors that receive the highest ranking form the context. The descriptor's weight, w_i, is calculated according to the following steps:

- Set all n descriptors in descending weight order according to the number of webpage references:

  $$\{\langle c_i, w_{i1}\rangle_{1\le i1\le n-1} \mid w_{i1} \le w_{i1+1}\}.$$

  Current References Difference Value, $\mathcal{D}(\mathcal{R})_i = \{w_{i1+1} - w_{i1}, _{1\le i1\le n-1}\}$.
- Set all n descriptors in descending weight order according to the number of appearances in the WSDL:

  $$\{\langle c_i, w_{i2}\rangle_{1\le i2\le n-1} \mid w_{i2} \le w_{i2+1}\}.$$

  Current Appearances Difference Value, $\mathcal{D}(\mathcal{A})_i = \{w_{i2+1} - w_{i2}, _{1\le i2\le n-1}\}$.
- Let $\mathcal{M}_r$ be the Maximum Value of References and $\mathcal{M}_a$ be the Maximum Value of Appearances:

  $$\mathcal{M}_r = \max_i\{\mathcal{D}(\mathcal{R})_i\},$$
  $$\mathcal{M}_a = \max_i\{\mathcal{D}(\mathcal{A})_i\}.$$

- The combined weight, w_i of the number of appearances in the WSDL and the number of references in the web is calculated according to the following formula:

  $$w_i = \sqrt{\left(\frac{2 * \mathcal{D}(\mathcal{A})_i * \mathcal{M}_r}{3 * \mathcal{M}_a}\right)^2 + (\mathcal{D}(\mathcal{R})_i)^2}. \quad (4)$$

The context recognition algorithm consists of the following major phases: 1) selecting contexts for each set of tokens, 2) ranking the contexts, and 3) declaring the current contexts. The result of the token extraction is a list of tokens obtained from the web service WSDL. The input to the algorithm is based on the *name* descriptor tokens extracted from the web service WSDL. The selection of the context descriptors is based on searching the web for relevant documents according to these tokens and on clustering the results into possible context descriptors. The output of the ranking stage is a set of highest ranking context descriptors. The set of context descriptors that have the top number of references, both in number of webpages and in number of appearances in the WSDL, is declared to be the context and the weight is defined by integrating the value of references and appearances.

Fig. 4 provides the outcome of the Web Context Extraction method for the `DomainSpy` service (see bottom right part). The figure shows only the highest ranking descriptors to be included in the context. For example, *Domain*, *Address*, *Registration*, *Hosting*, *Software*, and *Search* are the context descriptors selected to describe the `DomainSpy` service.

3.5 Concept Evocation

Concept evocation identifies a possible concept definition that will be refined next in the ontology evolution. The concept evocation is performed based on context intersection. An ontology concept is defined by the descriptors that appear in the intersection of both the web context results and the TF/IDF results. We defined one descriptor set from the TF/IDF results, tf/idf_{result}, based on extracted tokens from the WSDL text. The *context*, $\mathcal{C}$, is initially defined as a descriptor set extracted from the web and representing the same document. As a result, the ontology concept is represented by a set of descriptors, c_i, which belong to both sets:

$$Concept = \{c_1, \ldots, c_n | c_i \in tf/idf_{result} \cap c_i \in \mathcal{C}\}. \quad (5)$$

Fig. 5 displays an example of the concept evocation process. Each web service is described by two overlapping circles. The left circle displays the TF/IDF results and the right circle the web context results. The possible concept

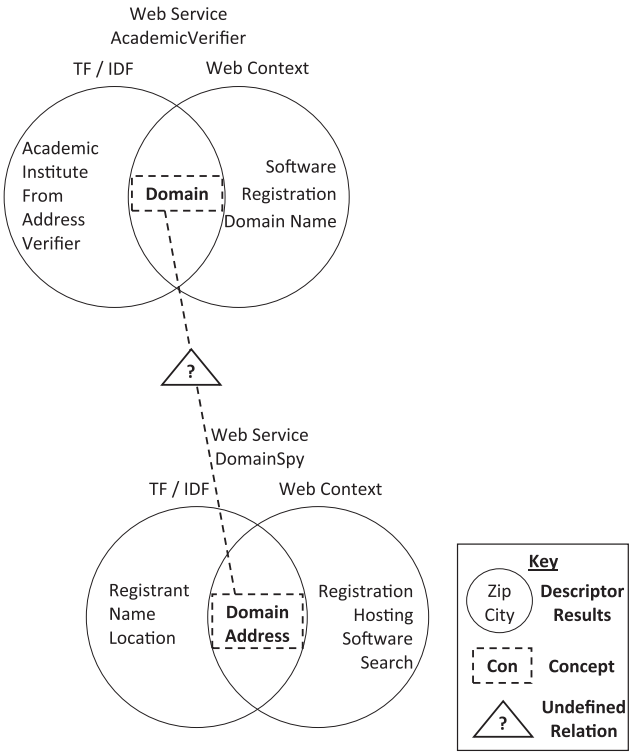

Fig. 5. Concept evocation example.

identified by the intersection is represented in the overlap between both methods. The unidentified relation between the concepts is described by a triangle with a question mark. The concept that is based on the intersection of both descriptor sets can consist of more than one descriptor. For example, the `DomainSpy` web service is identified by the descriptors *Domain* and *Address*. For the `Academic-Verifier` web service, which determines whether an email address or web domain name belongs to an academic institution, the concept is identified as *Domain*. Stemming is performed during the concept evocation on both the set of descriptors that represent each concept and the set of descriptors that represent the relations between concepts. The stemming process preserved descriptors *Registrant* and *Registration* due to their syntactical word structure. However, analyzing the decision from the domain specific perspective, the decision "makes sense," since one describes a person and the other describes an action.

A context can consist of multiple descriptor sets and can be viewed as a metarepresentation of the web service. The added value of having such a metarepresentation is that each descriptor set can belong to several ontology concepts simultaneously. For example, a descriptor set $\{\langle Registration, 23\rangle\}$ can be shared by multiple ontology concepts (Fig. 5) that are related to the domain of web registration. The different concepts can be related by verifying whether a specific web domain exists, web domain spying, etc., although the descriptor may have different relevance to the concept and hence different weights are assigned to it. Such overlap of contexts in ontology concepts affects the task of web service ontology bootstrapping. The appropriate interpretation of a web service context that is part of several ontology concepts is that the service is relevant to all such concepts. This leads to the possibility of the same service belonging to multiple concepts based on different perspectives of the service use.

The concept relations can be deduced based on convergence of the context descriptors. The ontology concept is described by a set of contexts, each of which includes descriptors. Each new web service that has descriptors similar to the descriptors of the concept adds new additional descriptors to the existing sets. As a result, the most common context descriptors that relate to more than one concept can change after every iteration. The sets of descriptors of each concept are defined by the union of the descriptors of both the web context and the TF/IDF results. The *context* is expanded to include the descriptors identified by the web context, the TF/IDF, and the concept descriptors. The expanded context, $Context_e$, is represented as the following:

$$Context_e = \{c_1, \ldots, c_n | c_i \in tf/idf_{result} \cup c_i \in \mathcal{C}\}. \quad (6)$$

For example, in Fig. 5, the `DomainSpy` web service context includes the descriptors: *Registrant*, *Name*, *Location*, *Domain*, *Address*, *Registration*, *Hosting*, *Software*, and *Search*, where two concepts are overlapping with the TF/IDF results of *Domain* and *Address*, and in addition TF/IDF adds the descriptors: *Registrant*, *Name*, and *Location*.

The relation between two concepts, Con_i and Con_j, can be defined as the context descriptors common to both concepts, for which weight w_k is greater than a cutoff value of a:

$$Re(Con_i, Con_j) = \{c_k | c_k \in Con_i \cap Con_j, w_k > a\}. \quad (7)$$

However, since multiple context descriptors can belong to two concepts, the cutoff value of a for the relevant descriptors needs to be predetermined. A possible cutoff can be defined by TF/IDF, Web Context, or both. Alternatively, the cutoff can be defined by a minimum number or percent of web services belonging to both concepts based on shared context descriptors. The relation between the two concepts *Domain* and *Domain Address* in Fig. 5 can be based on *Domain* or *Registration*. In the example displayed in Fig. 5, the value of the cutoff weight was selected as $a = 0.9$, and therefore all descriptors identified by both the TF/IDF and the Web Context methods with weight value over 0.9 were included in the relation between both concepts. The TF/IDF and the Web context each have different value ranges and can be correlated. A cutoff value of 0.9, which was used in the experiments, specifies that any concept that appears in the results of both the Web context and the TF/IDF will be considered as a new concept. The ontology evolution step, which we will introduce next, identifies the conflicts between the concepts and their relations.

3.6 Ontology Evolution

The ontology evolution consists of four steps including:

1. building new concepts,
2. determining the concept relations,
3. identifying relations types, and
4. resetting the process for the next WSDL document.

Building a new concept is based on refining the possible identified concepts. The evocation of a concept in the previous step does not guarantee that it should be integrated with the current ontology. Instead, the new possible concept should be analyzed in relation to the current ontology.

ServiceObjects

XMethods

DomainSpy

http://ws2.serviceobjects.net/ds/domainspy.asmx?WSDL

Introducing DOTS DomainSpy, an XML-based domain database that allows you to identify domain registrants by region or registrant name. DOTS DomainSpy is a catalog of over 7 million domain registrants in the US. DOTS DomainSpy provides you with powerful demographics such as: registrant name, address, city, state, zip code, phone number, fax numbers, web site, email and much more!

Fig. 6. Textual description example of service `DomainSpy`.

The descriptor is further validated using the textual service descriptor. The analysis is based on the advantage that a web service can be separated into two descriptions: the WSDL description and a textual description of the web service in free text. The WSDL descriptor is analyzed to extract the context descriptors and possible concepts as described previously. The second descriptor, $\mathcal{D}^{S}_{desc} = \{t_1, t_2, \ldots, t_n\}$, represents the textual description of the service supplied by the service developer in free text. These descriptions are relatively short and include up to a few sentences describing the web service. Fig. 6 presents an example of free text description for the `DomainSpy` web service. The verification process includes matching the concept descriptors in simple string matching against all the descriptors of the service textual descriptor. We use a simple string-matching function, $match_{str}$, which returns 1 if two strings match and 0 otherwise.

Expanding the example in Fig. 7, we can see the concept evocation step on the top and the ontology evolution on the bottom, both based on the same set of services. Analysis of the `AcademicVerifier` service yields only one descriptor

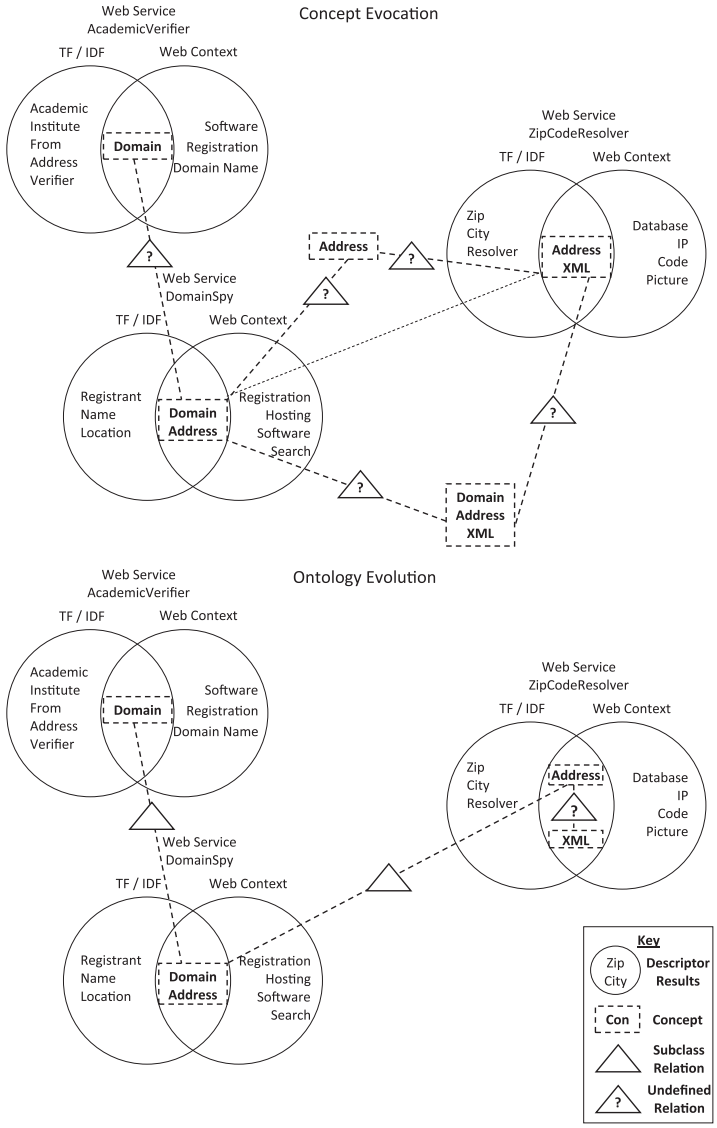

Fig. 7. Example of web service ontology bootstrapping.

1: For each Web service
2: Extract tokens from WSDL
3: TF/IDF_{result} = Apply TF/IDF algorithm to $\mathcal{D}_{wsdl}$
4: $WebContext_{result}$ = Apply Web Context algorithm to $\mathcal{D}_{wsdl}$
5: $PossibleCon_i = TF/IDF_{result} \cap WebContext_{result}$
6: If ($PossibleCon_i \subseteq \mathcal{D}_{desc}$)
7: $Con_i = TF/IDF_{result} \cap WebContext_{result}$
8: $PossibleRel_i = TF/IDF_{result} \cup WebContext_{result}$
9: For each concept pair Con_i, Con_j
10: If ($Con_i \subseteq Con_j$)
11: Con_i subclass Con_j
12: Else
13: $Re(Con_i, Con_j) = PossibleRel_i \cap PossibleRel_j$

Fig. 8. Ontology bootstrapping algorithm.

as a possible concept. The descriptor *Domain* was identified by both the TF/IDF and the web context results and matched with a textual descriptor. It is similar for the *Domain* and *Address* appearing in the `DomainSpy` service. However, for the `ZipCodeResolver` service both *Address* and *XML* are possible concepts but only *Address* passes the verification with the textual descriptor. As a result, the concept is split into two separate concepts and the `ZipCodeResolver` service descriptors are associated with both of them.

To evaluate the relation between concepts, we analyze the overlapping context descriptors between different concepts. In this case, we use descriptors that were included in the union of the descriptors extracted by both the TF/IDF and the Web context methods. Precedence is given to descriptors that appear in both concept definitions over descriptors that appear in the context descriptors. In our example, the descriptors related to both *Domain* and *Domain Address* are: *Software*, *Registration*, *Domain*, *Name*, and *Address*. However, only the *Domain* descriptor belongs to both concepts and receives the priority to serve as the relation. The result is a relation that can be identified as a subclass, where *Domain Address* is a subclass of *Domain*.

The process of analyzing the relation between concepts is performed after the concepts are identified. The identification of a concept prior to the relation allows in the case of *Domain Address* and *Address* to again apply the subclass relation based on the similar concept descriptor. However, the relation of *Address* and *XML* concepts remains undefined at the current iteration of the process since it would include all the descriptors that relate to `ZipCodeResolver` service. The relation described in the example is based on descriptors that are the intersection of the concepts. Basing the relations on a minimum number of web services belonging to both concepts will result in a less rigid classification of relations.

The process is performed iteratively for each additional service that is related to the ontology. The concepts and relations are defined iteratively as more services are added. The iterations stop once all the services are analyzed.

To summarize, we give the ontology bootstrapping algorithm in Fig. 8. The first step includes extracting the tokens from the WSDL for each web service (line 2). The next step includes applying the TF/IDF and the Web Context to extract the result of each algorithm (lines 3-4). The possible concept, $PossibleCon_i$, is based on the intersection of tokens of the results of both algorithms (line 5). If the $PossibleCon_i$ tokens appear in the document descriptor, $\mathcal{D}_{desc}$, then $PossibleCon_i$ is defined as concept, Con_i. The model evolves only when there is a match between all three methods. If $Con_i = \emptyset$, the web service will not classify a concept or a relation. The union of all token results is saved as $PossibleRel_i$ for concept relation evaluation (lines 6-8). Each pair of concepts, Con_i and Con_j, is analyzed for whether the token descriptors are contained in one another. If yes, a subclass relation is defined. Otherwise the concept relation can be defined by the intersection of the possible relation descriptors, $PossibleRel_i$ and $PossibleRel_j$, and is named according to all the descriptors in the intersection (lines 9-13).

4 Experiments

4.1 Experimental Data

The data for the experiments were taken from an existing benchmark repository provided by researchers from University College Dublin. Our experiments used a set of 392 web services, originally divided into 20 different topics such as: courier services, currency conversion, communication, business, etc. For each web service, the repository provides a WSDL document and a short textual description.

The concept relations experiments were based on comparing the methods results to existing ontologies relations. The analysis used the Swoogle ontology search engine[1] results for verification. Each pair of related terms proposed by the methods is verified using Swoogle term search.

4.2 Concept Generation Methods

The experiments examined three methods for generating ontology concepts, as described in Section 3:

- *WSDL Context.* The Context Extraction algorithm described in Section 3.4 was applied to the *name* labels of each web service. Each descriptor of the web service context was used as a concept.
- *WSDL TF/IDF.* Each word in the WSDL document was checked using the TF/IDF method as described in Section 3.3. The set of words with the highest frequency count was evaluated.
- *Bootstrapping.* The concept evocation is performed based on context intersection. An ontology concept can be identified by the descriptors that appear in the intersection of both the web context results and the TF/IDF results as described in Section 3.5 and verified against the web service textual descriptor (Section 3.6).

4.3 Concept Generation Results

The first set of experiments compared the precision of the concepts generated by the different methods. The concepts included a collection of all possible concepts extracted from each web service. Each method supplied a list of concepts that were analyzed to evaluate how many of them are meaningful and could be related to at least one of the services. The precision is defined as the number of relevant (or useful) concepts divided by the total number of concepts generated by the method. A set of an increasing number of web services was analyzed for the precision.

Fig. 9 shows the precision results of the three methods (i.e., Bootstrapping, WSDL TF/IDF, and the WSDL Context). The X-axis represents the number of analyzed web

1. http://swoogle.umbc.edu.

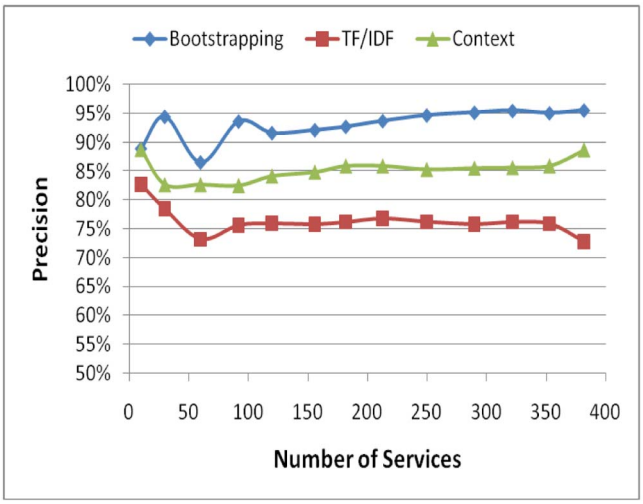

Fig. 9. Method comparison of precision per number of services.

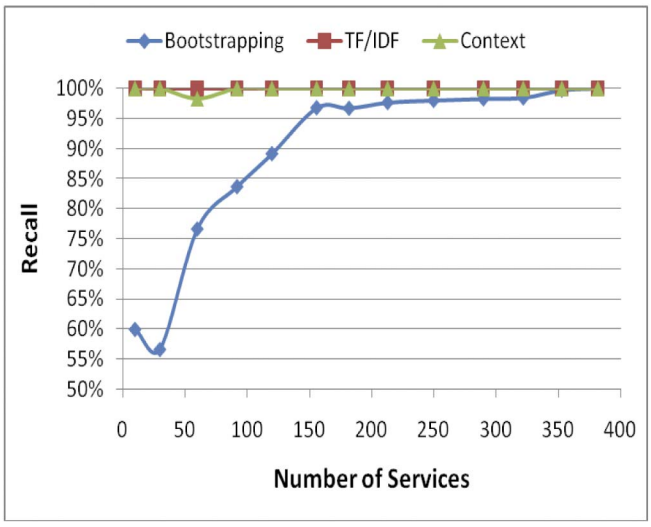

Fig. 10. Method comparison of recall per number of services.

services, ranging from 1 to 392, while the Y-axis represents the precision of concept generation.

It is clear that the Bootstrapping method achieves the highest precision, starting from 88.89 percent when 10 services are analyzed and converging (stabilizing) at 95 percent when the number of services is more than 250. The Context method achieves an almost similar precision of 88.76 percent when 10 services are analyzed but only 88.70 percent when the number of services reaches 392. In most cases, the precision results of the Context method are lower by about 10 percent than those of the Bootstrapping method. The TF/IDF method achieves the lowest precision results, ranging from 82.72 percent for 10 services to 72.68 percent for 392 services, lagging behind the Bootstrapping method by about 20 percent. The results suggest a clear advantage of the Bootstrapping method.

The second set of experiments compared the recall of the concepts generated by the methods. The list of concepts was used to analyze how many of the web services could be classified correctly to at least one concept. Recall is defined as the number of classified web services according to the list of concepts divided by the number of services. As in the precision experiment, a set of an increasing number of web services was analyzed for the recall.

Fig. 10 shows the recall results of the three methods, which suggest an opposite result to the precision experiment. The Bootstrapping method presented an initial lowest recall result starting from 60 percent at 10 services and dropping to 56.67 percent at 30 services, then slowly converging to 100 percent at 392 services. The Context and TF/IDF methods both reach 100 percent recall almost throughout. The nearly perfect results of both methods are explained by the large number of concepts extracted, many of which are irrelevant. The TF/IDF method is based on extracting concepts from the text for each service, which by definition guarantees the perfect recall. It should be noted that after analyzing 150 web services, the bootstrapping recall results remain over 95 percent.

The last concept generation experiment compared the recall and the precision for each method. An ideal result for a recall versus precision graph would be a horizontal curve with high precision value; a poor result has a horizontal curve with a low precision value. The recall-precision curve is widely considered by the IR community to be the most informative graph showing the effectiveness of the methods.

Fig. 11 depicts the recall versus precision results. Both the Context method and the TF/IDF method results are displayed at the right end of the scale. This is due to the nearly perfect recall achieved by the two methods. The Context method achieves slightly better results than does the TF/IDF method. Despite the nearly perfect recall achieved by both methods, the Bootstrapping method dominates the Context method and the TF/IDF method. The comparison of the recall and precision suggests the overall advantage of the Bootstrapping method.

4.4 Concept Relations Results

We also conducted a set of experiments to compare the number of true relations identified by the different methods. The list of concept relations generated from each method was verified against the Swoogle ontology search engine. If, for each pair of related concepts, the term option

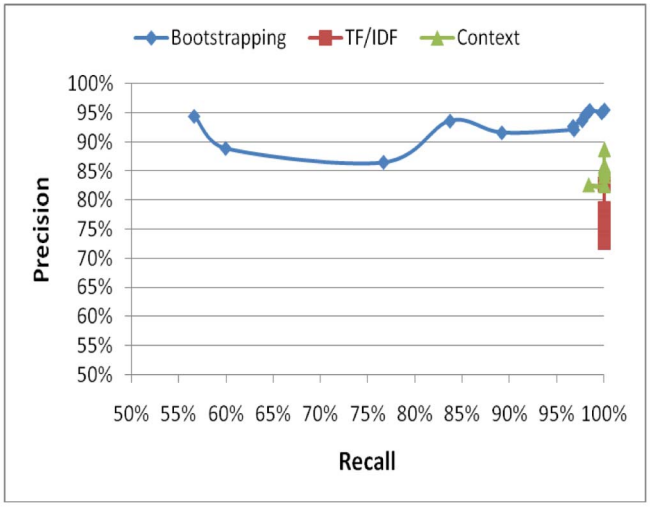

Fig. 11. Method comparison of recall versus precision.

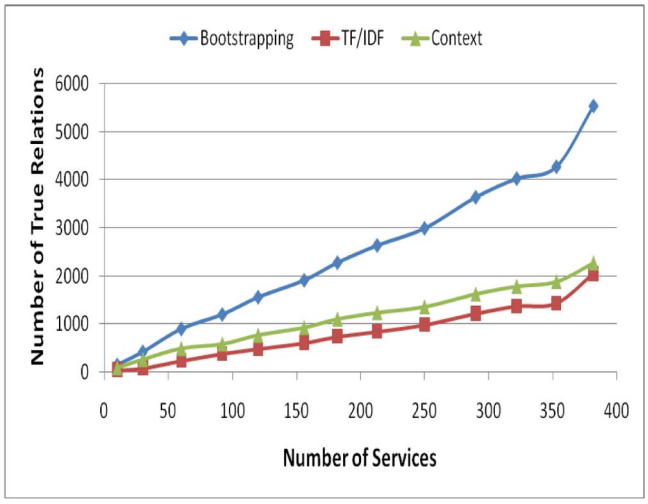

Fig. 12. Method comparison of true relations identified per number of services.

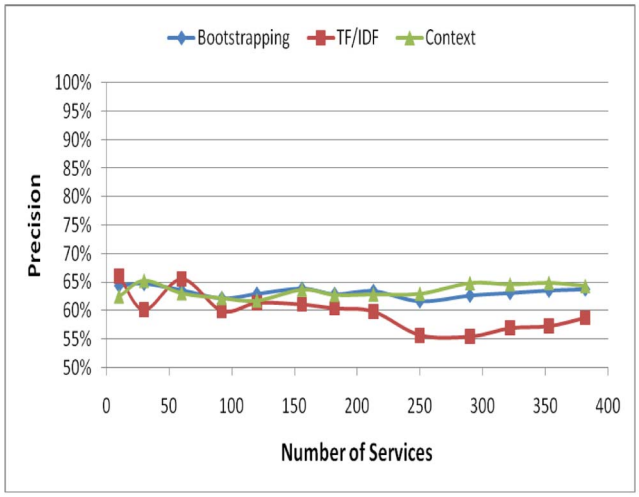

Fig. 13. Method comparison of relations precision per number of services.

of the search engine returns a result, then this relation is counted as a true relation. We analyzed the number of true relations results since counting all possible or relevant relations would be dependent on a specific domain. The same set of web services was used in the experiment.

Fig. 12 displays the number of true relations identified by the three methods. It can be seen that the bootstrapping method dominates the TF/IDF and the Context methods. For 10 web services, the number of concept relations identified by the TF/IDF method is 35 and by the Context method 80, while the Bootstrapping method identifies 148 relations. The difference is even more significant for 392 web services where the TF/IDF method identifies 2,053 relations, the Context method identifies 2,273 relations, and the Bootstrapping method identifies 5,542 relations.

We also compared the precision of the concept relations generated by the different methods. The precision is defined as the number of pairs of concept relations identified as true against the Swoogle ontology search engine results divided by the total number of pairs of concept relations generated by the method. Fig. 13 presents the concept relations precision results. The precision results for 10 web services are 66.04 percent for the TF/IDF, 64.35 percent for the bootstrapping, and 62.50 percent for the Context. For 392 web services the Context method achieves a precision of 64.34 percent, the Bootstrapping method 63.72 percent, and TF/IDF 58.77 percent. The average precision achieved by the three methods is 63.52 percent for the Context method, 63.25 percent for the bootstrapping method, and 59.89 percent for the TF/IDF.

From Fig. 12, we can see that the bootstrapping method correctly identifies approximately twice as many concept relations as the TF/IDF and Context methods. However, the precision of concept relations displayed in Fig. 13 remains similar for all three methods. This clearly emphasizes the ability of the bootstrapping method to increase the recall significantly while maintaining a similar precision.

5 Discussion

We have presented a model for bootstrapping an ontology representation for an existing set of web services. The model is based on the interrelationships between an ontology and different perspectives of viewing the web service. The ontology bootstrapping process in our model is performed automatically, enabling a constant update of the ontology for every new web service.

The web service WSDL descriptor and the web service textual descriptor have different purposes. The first descriptor presents the web service from an internal point of view, i.e., what concept best describes the content of the WSDL document. The second descriptor presents the WSDL document from an external point of view, i.e., if we use web search queries based on the WSDL content, what most common concept represents the answers to those queries.

Our model analyzes the concept results and concept relations and performs stemming on the results. It should be noted that other techniques of clustering could be used to limit the ontology expansion, such as clustering by synonyms or minor syntactic variations.

Analysis of the experiment results where the model did not perform correctly presents some interesting insights. In our experiments, there were 28 web services that did not yield any possible concept classifications. Our analysis shows that 75 percent of the web services without relevant concepts were due to no match between the results of the Context Extraction method, the TF/IDF method, and the free text web service descriptor. The rest of the misclassified results derived from input formats that include special, uncommon formatting of the WSDL descriptors and from the analysis methods not yielding any relevant results. Of the 28 web services without possible classification, 42.86 percent resulted from mismatch between the Context Extraction and the TF/IDF. The remaining web services without possible classification derived from when the results of the Context Extraction and the TF/IDF did not match with the free text descriptor.

Some problems indicated by our analysis of the erroneous results point to the substring analysis. 17.86 percent of the mistakes were due to limiting the substring concept checks. These problems can be avoided if the substring checks are performed on the results of Context Extractions versus the TF/IDF and vice versa for each result and if, in

addition, substring matching of the free text web service description is performed.

The matching can further be improved by checking for synonyms between the results of the Context Extractions, the TF/IDF, and free text descriptors. Using a thesaurus could resolve up to 17.86 percent of the cases that did not yield a result. However, using substring matching or a thesaurus in this process to expand the results of each method could lead to a drop in the integrated model precision results.

Another issue is the question of what makes some web services more relevant than others in the ontology bootstrapping process. If we analyze a relevant web service as a service that can add more concepts to the ontology, then each web service that belongs to a new domain has greater probability of supplying new concepts. Thus, an ontology evolution could converge faster if we were to analyze services from different domains at the beginning of the process. In our case, Figs. 9 and 10 indicate that the precision and recall of the number of concepts identified converge after 156 randomly selected web services were analyzed. However, the number of concepts relations continues to grow linearly as more web services are added, as displayed in Fig. 12.

The iterations of the ontology construction are limited by the requirement to analyze the TF/IDF method on all the collected services since the inverse document frequency method requires all the web services WSDL descriptors to be analyzed at once while the model iteratively adds each web Service. This limitation could be overcome by either recalculating the TF and IDF after each new web service or alternatively collecting an additional set of services and reevaluating the IDF values. We leave the study of the effect on ontology construction of using the TF/IDF with only partial data for future work.

The model can be implemented with human intervention, in addition to the automatic process. To improve performance, the algorithm could process the entire collection of web services and then concepts or relations that are identified as inconsistent or as not contributing to the web service classification can be manually altered. An alternative option is introducing human intervention after each cycle, where each cycle includes processing a predefined set of web services.

Finally, it is impractical to assume that the simplified search techniques offered by the UDDI make it very useful for web services discovery or composition [40]. Business registries are currently used for the cataloging and classification of web services and other additional components. UDDI Business Registries (UBR) serve as the central service directory for the publishing of technical information about web services. Although the UDDI provides ways for locating businesses and how to interface with them electronically, it is limited to a single search criterion [41]. Our method allows the main limitations of a single search criterion to be overcome. In addition, our method does not require registration or manual classification of the web services.

6 Conclusion

The paper proposes an approach for bootstrapping an ontology based on web service descriptions. The approach is based on analyzing web services from multiple perspectives and integrating the results. Our approach takes advantage of the fact that web services usually consist of both WSDL and free text descriptors. This allows bootstrapping the ontology based on WSDL and verifying the process based on the web service free text descriptor.

The main advantage of the proposed approach is its high precision results and recall versus precision results of the ontology concepts. The value of the concept relations is obtained by analysis of the union and intersection of the concept results. The approach enables the automatic construction of an ontology that can assist, classify, and retrieve relevant services, without the prior training required by previously developed methods. As a result, ontology construction and maintenance effort can be substantially reduced. Since the task of designing and maintaining ontologies remains difficult, our approach, as presented in this paper, can be valuable in practice.

Our ongoing work includes further study of the performance of the proposed ontology bootstrapping approach. We also plan to apply the approach in other domains in order to examine the automatic verification of the results. These domains can include medical case studies or law documents that have multiple descriptors from different perspectives.

References

[1] N.F. Noy and M. Klein, "Ontology Evolution: Not the Same as Schema Evolution," *Knowledge and Information Systems,* vol. 6, no. 4, pp. 428-440, 2004.

[2] D. Kim, S. Lee, J. Shim, J. Chun, Z. Lee, and H. Park, "Practical Ontology Systems for Enterprise Application," *Proc. 10th Asian Computing Science Conf. (ASIAN '05),* 2005.

[3] M. Ehrig, S. Staab, and Y. Sure, "Bootstrapping Ontology Alignment Methods with APFEL," *Proc. Fourth Int'l Semantic Web Conf. (ISWC '05),* 2005.

[4] G. Zhang, A. Troy, and K. Bourgoin, "Bootstrapping Ontology Learning for Information Retrieval Using Formal Concept Analysis and Information Anchors," *Proc. 14th Int'l Conf. Conceptual Structures (ICCS '06),* 2006.

[5] S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, S. Melzer, R. Moller, S. Montanelli, and G. Petasis, "Ontology Dynamics with Multimedia Information: The BOEMIE Evolution Methodology," *Proc. Int'l Workshop Ontology Dynamics (IWOD '07), held with the Fourth European Semantic Web Conf. (ESWC '07),* 2007.

[6] C. Platzer and S. Dustdar, "A Vector Space Search Engine for Web Services," *Proc. Third European Conf. Web Services (ECOWS '05),* 2005.

[7] L. Ding, T. Finin, A. Joshi, R. Pan, R. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A Search and Metadata Engine for the Semantic Web," *Proc. 13th ACM Conf. Information and Knowledge Management (CIKM '04),* 2004.

[8] A. Patil, S. Oundhakar, A. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework," *Proc. 13th Int'l World Wide Web Conf. (WWW '04),* 2004.

[9] Y. Chabeb, S. Tata, and D. Belad, "Toward an Integrated Ontology for Web Services," *Proc. Fourth Int'l Conf. Internet and Web Applications and Services (ICIW '09),* 2009.

[10] Z. Duo, J. Li, and X. Bin, "Web Service Annotation Using Ontology Mapping," *Proc. IEEE Int'l Workshop Service-Oriented System Eng. (SOSE '05),* 2005.

[11] N. Oldham, C. Thomas, A.P. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework with Machine Learning Classification," *Proc. First Int'l Workshop Semantic Web Services and Web Process Composition (SWSWPC '04),* 2004.

[12] A. Heß, E. Johnston, and N. Kushmerick, "ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services," *Proc. Third Int'l Semantic Web Conf. (ISWC '04),* 2004.

[13] Q.A. Liang and H. Lam, "Web Service Matching by Ontology Instance Categorization," *Proc. IEEE Int'l Conf. on Services Computing (SCC '08),* pp. 202-209, 2008.

[14] A. Segev and E. Toch, "Context-Based Matching and Ranking of Web Services for Composition," *IEEE Trans. Services Computing,* vol. 2, no. 3, pp. 210-222, July-Sept. 2009.
[15] J. Madhavan, P. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 49-58, Sept. 2001.
[16] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to Map between Ontologies on the Semantic Web," *Proc. 11th Int'l World Wide Web Conf. (WWW '02),* pp. 662-673, 2002.
[17] A. Gal, G. Modica, H. Jamil, and A. Eyal, "Automatic Ontology Matching Using Application Semantics," *AI Magazine,* vol. 26, no. 1, pp. 21-31, 2005.
[18] J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy, "Representing and Reasoning about Mappings between Domain Models," *Proc. 18th Nat'l Conf. Artificial Intelligence and 14th Conf. Innovative Applications of Artificial Intelligence (AAAI/IAAI),* pp. 80-86, 2002.
[19] V. Mascardi, A. Locoro, and P. Rosso, "Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation," *IEEE Trans. Knowledge and Data Eng.,* doi:10.1109/TKDE.2009.154, 2009.
[20] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi, "A Framework for Modeling and Evaluating Automatic Semantic Reconciliation," *Int'l J. Very Large Data Bases,* vol. 14, no. 1, pp. 50-67, 2005.
[21] B. Vickery, *Faceted Classification Schemes.* Graduate School of Library Service, Rutgers, The State Univ., 1966.
[22] P. Spyns, R. Meersman, and M. Jarrar, "Data Modelling versus Ontology Engineering," *ACM SIGMOD Record,* vol. 31, no. 4, pp. 12-17, 2002.
[23] A. Maedche and S. Staab, "Ontology Learning for the Semantic Web," *IEEE Intelligent Systems,* vol. 16, no. 2, pp. 72-79, Mar./Apr. 2001.
[24] C.Y. Chung, R. Lieu, J. Liu, A. Luk, J. Mao, and P. Raghavan, "Thematic Mapping—From Unstructured Documents to Taxonomies," *Proc. 11th Int'l Conf. Information and Knowledge Management (CIKM '02),* 2002.
[25] V. Kashyap, C. Ramakrishnan, C. Thomas, and A. Sheth, "TaxaMiner: An Experimentation Framework for Automated Taxonomy Bootstrapping," *Int'l J. Web and Grid Services,* Special Issue on Semantic Web and Mining Reasoning, vol. 1, no. 2, pp. 240-266, Sept. 2005.
[26] D. McGuinness, R. Fikes, J. Rice, and S. Wilder, "An Environment for Merging and Testing Large Ontologies," *Proc. Int'l Conf. Principles of Knowledge Representation and Reasoning (KR '00),* 2000.
[27] F.N. Noy and M.A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," *Proc. 17th Nat'l Conf. Artificial Intelligence (AAAI '00),* pp. 450-455, 2000.
[28] H. Davulcu, S. Vadrevu, S. Nagarajan, and I. Ramakrishnan, "OntoMiner: Bootstrapping and Populating Ontologies from Domain Specific Web Sites," *IEEE Intelligent Systems,* vol. 18, no. 5, pp. 24-33, Sept./Oct. 2003.
[29] H. Kim, J. Hwang, B. Suh, Y. Nah, and H. Mok, "Semi-Automatic Ontology Construction for Visual Media Web Service," *Proc. Int'l Conf. Ubiquitous Information Management and Comm. (ICUIMC '08),* 2008.
[30] Y. Ding, D. Lonsdale, D. Embley, M. Hepp, and L. Xu, "Generating Ontologies via Language Components and Ontology Reuse," *Proc. 12th Int'l Conf. Applications of Natural Language to Information Systems (NLDB '07),* 2007.
[31] Y. Zhao, J. Dong, and T. Peng, "Ontology Classification for Semantic-Web-Based Software Engineering," *IEEE Trans. Services Computing,* vol. 2, no. 4, pp. 303-317, Oct.-Dec. 2009.
[32] M. Rambold, H. Kasinger, F. Lautenbacher, and B. Bauer, "Towards Autonomic Service Discovery—A Survey and Comparison," *Proc. IEEE Int'l Conf. Services Computing (SCC '09),* 2009.
[33] M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt, "Learning Domain Ontologies for Semantic Web Service Descriptions," *Web Semantics,* vol. 3, no. 4, pp. 340-365, 2005.
[34] M. Sabou and J. Pan, "Towards Semantically Enhanced Web Service Repositories," *Web Semantics,* vol. 5, no. 2, pp. 142-150, 2007.
[35] T.R. Gruber, "A Translation Approach to Portable Ontologies," *Knowledge Acquisition,* vol. 5, no. 2, pp. 199-220, 1993.
[36] S. Robertson, "Understanding Inverse Document Frequency: On Theoretical Arguments for IDF," *J. Documentation,* vol. 60, no. 5, pp. 503-520, 2004.
[37] C. Mooers, *Encyclopedia of Library and Information Science,* vol. 7, ch. Descriptors, pp. 31-45, Marcel Dekker, 1972.
[38] A. Segev, M. Leshno, and M. Zviran, "Context Recognition Using Internet as a Knowledge Base," *J. Intelligent Information Systems,* vol. 29, no. 3, pp. 305-327, 2007.
[39] R.E. Valdes-Perez and F. Pereira, "Concise, Intelligible, and Approximate Profiling of Multiple Classes," *Int'l J. Human-Computer Studies,* pp. 411-436, 2000.
[40] E. Al-Masri and Q.H. Mahmoud, "Investigating Web Services on the World Wide Web," *Proc. Int'l World Wide Web Conf. (WWW '08),* 2008.
[41] L.-J. Zhang, H. Li, H. Chang, and T. Chao, "XML-Based Advanced UDDI Search Mechanism for B2B Integration," *Proc. Fourth Int'l Workshop Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS '02),* June 2002.

Aviv Segev received the PhD degree from Tel-Aviv University in management information systems in the field of context recognition in 2004. He is an assistant professor in the Knowledge Service Engineering Department at the Korea Advanced Institute of Science and Technology (KAIST). His research interests include classifying knowledge using the web, context recognition and ontologies, knowledge mapping, and implementations of these areas in the fields of web services, medicine, and crisis management. He is the author of over 40 publications. He is a member of the IEEE.

Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales, Sydney, Australia. He is a senior lecturer in the School of Computer Science at the University of Adelaide. His research interests include service-oriented architectures, web of things, distributed computing, and pervasive computing. He was the recipient of the 2011 Chris Wallace Award for Outstanding Research Contribution and the 2003 Microsoft Research Fellowship. He is the author of more than 90 publications. He is a member of the IEEE and the ACM.