ISSN: 2321-7782 (Online)
Volume 3, Issue 9, September 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Ontology Bootstrapping Implementation for Semantic Web Services to Automate WSDL

Sasanka Bhimavaram[1]
Research Scholar, Department of Computer Science
Sri Venkateswara University,
Tirupati, India

Dr. P. Govindarajulu[2]
Professor, Department of Computer Science
Sri Venkateswara University,
Tirupati, India

Abstract: Ontology Bootstrapping is based on a set of textual sources, such as web services, must address the problem of multiple and largely isolated concepts. Web Information Extraction is essential to many application areas, such as ontology mapping and bootstrapping Semantic Web etc. Study shows that instances are usually backed by fundamental relational databases in majority Web sites. Based on this examination, two approaches for Instance Extraction are illustrated in this research. The objective of this research is to bootstrap instances, rich ontologies from Web documents. In this paper, we intend an ontology bootstrapping process for web services. We abuse the improvement that web services typically consist of both WSDL and free text descriptors. Our proposed Improved Enhanced Traversal method for automation of ontology bootstrapping procedure integrates the results of both methods and applies a third method to validate the concepts using the service free text descriptor, thus offering a more accurate classification of ontologies.

Keywords: Ontology, Semantic Web, WSDL, Web Services.

I. Introduction

We distinguish between ontologies in a wider sense and in a narrow sense. Similar to the term "information classification in a wider sense" and the preceding observation of ontologies contains various aspects, which either cannot be formal at all or which cannot be formal for realistic reasons. For illustration, it might be too burdensome to form the interests of all the persons involved. Regarding the official part of ontologies, we utilize a particular construction. The primary parts describe the structural properties that practically all ontology languages demonstrate. Remind that we do not classify any syntax here, but basically refer to these structures as a slightest general denominator for various logical languages[4,7].

Definition: Let L be a logical language having a formal semantics in which implication regulations can be articulated. An intangible ontology is a formation O: = (C; _C; R; _; _R; IR) consisting of

» two disjoint sets C and R these fundamentals are called concepts and relations,

» a fractional order ≤C on C, called concept hierarchy or classification,

» a function £:R→ C×C called signature,

» a partial order ≤R on R where r1 ≤ R r2 implies £(r1) ≤C×C £(r2) , for r1, r2 2 € R, called relation hierarchy. And a set IR of inference regulations articulated in the logical language L.

The function dom: R→C with dom(r):=π1(£(r)) gives the domain of r.

The function range: R→C with range(r):=π2(£(r)) gives its range.

To join the theoretical ontology formation to natural language we use a plain demonstration of the lexical level. Consequently, we define a lexicon for our conceptual ontology as follows:

1.1 Ontology Learning

Ontology Learning is a mechanism that automatically supports the process of extracting and maintaining ontology from a particular set of data. Thus, it applies machine learning techniques on particular data for the routine extraction of data structures that are projected to and distinguished by the ontology engineer.

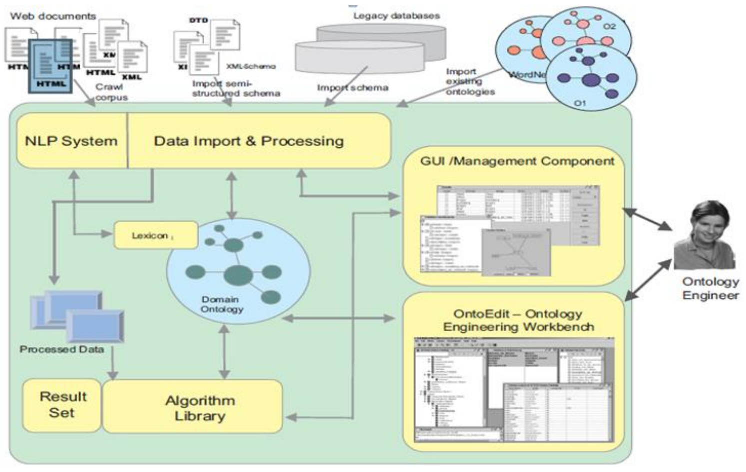


Figure 1: Architecture for Ontology Learning

We here only provide a violent sketch of the Ontology Learning approach by introducing our standard architecture for extracting and maintaining ontologies from particular data, in particular natural language texts. We have recognized three main core architectural mechanisms [2,5]. There are,

A. A standard management module dealing with assignment of tasks and constituting the infrastructure backbone.

B. A data essential and processing component working on input data from the Web counting, in exacting the natural language processing.

C. An algorithm records functioning on the output of the data import and processing module as well as the ontology structures sketched.

II. Literature Review

2.1 WSDL

Web Services Description Language (WSDL) makes it straightforward to obtain the benefits of SOAP by providing an approach for Web service providers and users of such services to work together simply. For Web services, it's much more difficult. Initially we have to agree on a regular design for specifying interfaces. Without WSDL, we can determine the calling syntax from credentials that must be provided, or by examining chain messages. Either way, a human will have to be concerned and so the development is level to inaccuracy. With WSDL, I can automate the creation of proxies for Web services in an accurately and platform independent approach. WSDL file is a convention between client and server[9,11].

2.2 WSDL Document Structure

When trying to comprehend any XML document, it helps to have a block diagram. The diagram illustrates the construction of WSDL, which is an XML document, by screening the interaction among the sections that make up a WSDL document. The WSDL document can be separated into two groups of sections. The top group is comprised of Abstract and Conceptual Definitions, while the bottom group consists of tangible Descriptions. The abstract and conceptual sections define SOAP messages in a stage and language autonomous approach. They do not hold any machine or language precise essentials. This helps describe a rest of services that some, miscellaneous Web sites can execute. Site exact matters such as series are then relegated to the bottom sections, which contain tangible descriptions.

2.3 Abstract and Conceptual Definitions:

Types

Machine and language autonomous structure definitions.

Messages

Contains function inputs parameters detach from outputs or document descriptions.

PortTypes

Refers to message definitions in Messages segment to illustrate function signatures (operation name, input parameters and output parameters).

Concrete Descriptions and Bindings

Specifies binding(s) to each and every operation in the PortTypes section.

Services

Specifies port address (es) of each and every binding.

In this backgrounder I will be using standard XML terminology to describe the WSDL document. The declaration "element" refers to an XML element and the word "attribute" refers to an element attribute. Thus:

<element attribute="attribute-value">contents</element>

III. ALGORITHMS FOR COMPUTATION

The subsumption relation stimulates the computation of model hierarchy. In this component, we suppose that subsumption costs are dominating the classification costs and are significantly superior to the costs incurred by further operations. The subsequent methods discover the computation of model hierarchy and are taken from.

A. Brute Force Method

B. Traversal Method

3.1 Brute Force Method

The top search for Brute Force method computes for every atomic concept c its predecessors and can be explained as follows:

Blindly tests c $\cup$ x for all x 2 X_i, where X_i is atomic concept.

Definition: (Top Search) Top Search returns a set of immediate predecessors in X_i for a given element c. computes for every atomic concept its successors and is done in a dual way.

3.2 Traversal Method

In order to avoid many of the “brute force” method comparisons and in its place of testing the new element c blindly with all elements; in the top search phase of “simple traversal” method, c is pushed down the tree, and in the bottom search, phase c is pushed up, stopping when immediate predecessors or successors have been determined.

a) Top Search

Algorithm 1: Top Search Phase of the Traversal Method

Step 1: top_search(c,x)

```
Step 2: mark(x, “visited”)

Step 3: for all y € successors(x) do

Step 4: if top_subs(y,c) then

Step 5: if pos-succ→pos-succ {y}

Step 6: result →x

Step 7: else

Step 8: for all y € pos-succ do

Step 9: if y not marked as “visited”

Step 10: then result→result Ŭ top_search(c,y)

Step 11: stop
```

Algorithm 2: Simple top subsumption of the traversal method

```
Step 1: top_sub?(y,c)=

Step 2: if y marked as ‘positive’ then

Step 3: result→true

Step 4: else if y marked as ‘negative’ then

Step 5: result→false

Step 6: else if subs?(y,c) then

Step 7: mark(y,’positive’)

Step 8:result→true

Step 9:else

Step 10:mark(y,’negative’)

Step 11:result→false

Step 12: Stop
```

Starting at the top of the hierarchy, for each concept x of X_i (x is child of top), it is determined whether x has an immediate successor y satisfying c ∪ y. If there are such successors, they are considered as well. Otherwise, x is added to the result list of the top search. To avoid multiple visits of elements and multiple comparison of the same element with c, this algorithm employs one label to indicate whether a node was “visited” and another label to indicate whether the subsumption test was “positive” or “negative” has not been made. The top-search is depicted in Figure 3.1. The figure only shows the details for the upper half of the tree, as traversal is done from top. The top-search procedure is illustrated in Algorithm 1, and will be explained next. The algorithm gets two arguments as its input:

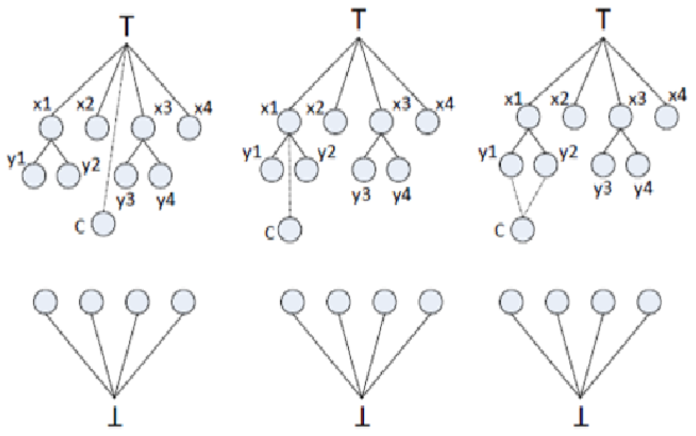


Figure 2: Insert a concept c through top search

a) Bottom Search

Bottom Search is symmetric to Top Search, therefore it is performed in a dual way. This means that if the tree is rotated upside down, then bottom is like top. Bottom Search is illustrated in Figure. The figure shows only the lower half of the tree, as bottom search traversal is done bottom-up. Starting at the bottom of the hierarchy, for each concept x of X_i (x is parent of bottom), it checks whether x has an immediate predecessor z which satisfies c $\cup$ z. If there are such predecessors, they are considered as well. The bottom-up traversal is done till the immediate successors of c are found[1,8,16].

Bottom search optimizations can be applied dual to top search. This optimization is achieved by a modification of the "enhanced-bottom-search" procedure (which is dual to "enhanced-top-search").

IV. Implementation And Results

4.1 Web Service Composition

Web Service composition is the process to find a new services S which consists of set of constituent web services {S1, S2.......Sn}. Every component web service is mapped to a set of real web services.

A. *Semantic Web Service Composition and Calculating Description Rate*

In this module, we evaluate semantic web service effort by focusing on its core mechanism as follows:

a) Semantic Web Services (we will suppose without loss of generality that each service refers to a single operation),

b) Their Semantic Links (also known as Causal Links) as a Formal way of representing their semantic connections

c) A way to model a composition through its constituent Semantic links.

B. *Algorithm: Web Service Composition*

Input: Service .Request

Output: Service Description

Step 1: Begin

Step 2: Read Service Request D_a for Service 1

Step 3: Read Service Request D_b for Service 2

Step 4: Combined set of services $D_{(a+b)}$

Step 5: Mapped Service Request and Associate WSDL file

Step 6: Display Service Description and Related terms

Step 7: Stop

4.2 Improved Enhanced Traversal Method

A traversal scheme over an extensive signature with word variables (Σ_π,w) is a coincidence of literals, where every literal has the form w1 = w2 (word equation), w1≡ w2 (traversal equation) or w € R (regular constraint), being w_i € (Σ_π ∪ w) words with variables and R ⊆ Σ_π a regular language.

A solution of a traversal scheme is a word swap σ :w→ Σ_π such that

1. σ (w1) = σ (w2) for any word equation w1 = w2,

2. σ (w1) and σ (w2) are both traversal sequences of the same term, for any traversal equation w1 ≡ w2 and

3. σ (w) belongs to R, for any regular constraint w € R.

Traversal method is improved with the scheme of traversal signature addition to the top search method. Then the Improved Enhanced Traversal method performs much better than the "brute force" system but it does not utilize all the available information. Therefore in the "Improved Enhanced Traversal" method, initially we can take improvement of the tests that have been performed during the top search; second, in the bottom search we can use all the information collected during the top search.

A. Top Search

B. Using negative information

C. Using positive information

Algorithm: Improved Top search phase of the "Improved Enhanced Treaversal" method

```
Step 1: enhanced_top_subs?(y,c)= $\Sigma_\pi$,w
Step 2: if y marked as 'positive' then
Step 3: result of w1≡ w2 → true
Step 4: else if y marked as 'negative' then
Step 5: result → false
Step 6: else if for all z € predecessors(y) always enhanced_top_subs(z,c)
Step 7: subs?(y,c) then w1 = w2
Step 8: mark (y,'positive')
Step 9: result R ⊆ $\Sigma_\pi$ → true
Step 10: else if mark(y, 'negative')
Step 11: else σ: w→ $\Sigma_\pi$
Step 12: result → false
```

Algorithm**:** first it tests y1, then checks x1 but before testing x1, its direct predecessor y2 is tested. The negative result of this test prevents x1,. . . , xn from being tested. However, the top search using positive information tests n+2 nodes in order to place c in the hierarchy: first y1 and then its successors x1,. . . , xn and finally y2.

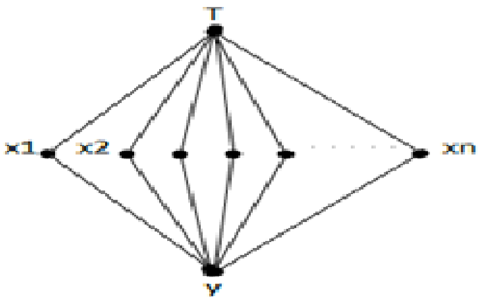


Figure 3: The new element c is a direct successor of y.

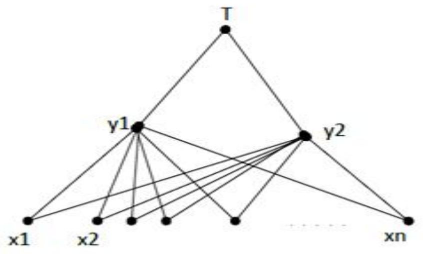


Figure 4: The new element c is direct successor of y1, but not a successor of y2, x1,…..,xn.

4.3 WSDL File

Let's dive right into a WSDL file to see its structure and how it works. Please be aware that this is an instance of a WSDL document to illustrate its most salient features.

```
<?xml version="1.0" encoding="UTF-8" ?>

<definitions name="FooSample"

targetNamespace="http://tempuri.org/wsdl/"

xmlns:wsdlns="http://tempuri.org/wsdl/"

xmlns:typens="http://tempuri.org/xsd"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:stk="http://schemas.microsoft.com/soap-toolkit/wsdl-extension"

xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>

<schema targetNamespace="http://tempuri.org/xsd"

xmlns="http://www.w3.org/2001/XMLSchema"

xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

elementFormDefault="qualified" >
```

```
</schema>

</types>

<message name="Simple.foo">

<part name="arg" type="xsd:int"/>

</message>

<message name="Simple.fooResponse">

<part name="result" type="xsd:int"/>

</message>

<portType name="SimplePortType">

<operation name="foo" parameterOrder="arg" >

<input message="wsdlns:Simple.foo"/>

<output message="wsdlns:Simple.fooResponse"/>

</operation>

</portType>

<binding name="SimpleBinding" type="wsdlns:SimplePortType">

<stk:binding preferredEncoding="UTF-8" />

<soap:binding style="rpc"

transport="http://schemas.xmlsoap.org/soap/http"/>

<operation name="foo">

<soap:operation

soapAction="http://tempuri.org/action/Simple.foo"/>

<input>

<soap:body use="encoded" namespace="http://tempuri.org/message/"

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />

</input>

<output>

<soap:body use="encoded" namespace="http://tempuri.org/message/"

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />

</output>

</operation>

</binding>

<service name="FOOSAMPLEService">
```

```
<port name="SimplePort" binding="wsdlns:SimpleBinding">

<soap:address location="http://carlos:8080/FooSample/FooSample.asp"/>

</port>

</service>

</definitions>
```

The initial row declares that the document is XML. Although it is not requisite, it helps the XML parser to determine whether to parse the WSDL file at all or indicate an error. The subsequent line is the root element in the WSDL document: <definitions>. There are a number of namespace attributes and declarations attached to the root element, as well as in the <schema> child element of the <types> element. The <types> element comprises the Types section. This section may be omitted if there are no data types that have to to be declared. In the WSDL, there are no application definite types confirmed but I use the types section anyhow just to state schema namespaces used in the document.

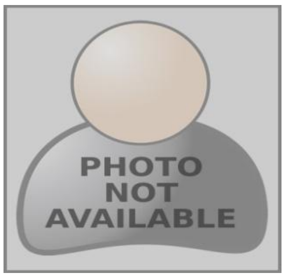
V. Conclusion

In this paper we have presented a wide ranging approach for bootstrapping an ontology based information mining system with the help of machine learning. We are applied the top search method to enhanced traversal equation with anticipated structural design which has been instantiated with WSDL the application of our approach will emerge in the Semantic Web, in the area of semantic annotation and metadata generation. We have designed and implemented Improved Enhanced Traversal algorithm for semantic web service automation using web service description language for better classification performance and accuracy measurement of ontology for web based services. In future work our proposed approach can be extended for improvement in the following highlighted aspects. Some of these open works were not explored due to time constraints and some were noticed while conducting the experimental evaluation for extending Informed Partitioning to run with larger ontologies.

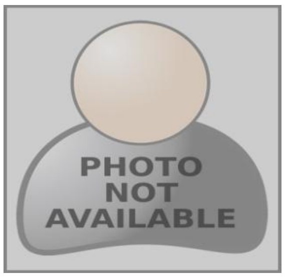
References

1. M. Ehrig and S. Staab, "QOM - Quick Ontology Mapping", In Proceedings of International Semantic Web Conference, pp.683-697, 2004.
2. J. Euzenat and P. Shvaiko, "Ontology Matching", Springer-Verlag, Heidelberg (DE), 2007.
3. D. Fensel, "Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce", Springer, 2001.
4. Motik, B., Horrocks, I.: OWL datatypes: Design and implementation. In: Proc. Of the Int. Semantic Web Conf. (ISWC-08). LNCS, vol. 5318, pp. 307–322, Springer 2008.
5. Sirin, E.—Parsia, B.—Cuenca Grau, B.—Kalyanpur, A.—Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics, Vol. 5, 2007.
6. Hassina Nacer Talantikite, Djamil Aissani, and Nacer Boudjlida, "Semantic annotations for web services discovery and composition," Computer Standards & Interfaces, vol. 31, no. 6, pp. 1108-1117, November 2009.
7. McIlraith, S., Son, T.C., Zeng, H. *Semantic web services*. IEEE Intell. Syst. 16(2), 46–53, 2001.
8. [Sheth and Meersman 2002] A. Sheth and R. Meersman, "Amicalola Report: Database and Information Systems Research Challenges and Opportunities in Semantic Web and Enterprises," ACM SIGMOD Record, Vol. 31, No. 4, pp. 98-106,2002.
9. McIlraith, S., Martin, D., "Bringing semantics to Web Services".IEEE Intelligent Systems, 2003.
10. Paliwal A. V, Adam, N., & Bornhoevd, C., Adding Semantics through Service Request Expansion & Latent Semantic Indexing, IEEE SCC 2007, July 9-13, Salt Lake City,Utah,USA,2007.
11. Straccia U., Troncy R., oMAP: Combining Classifiers for Alignment Automatically OWL Ontologies. In Proceedings of the 6th International Conference on Web Information Systems Engineering, pp. 133-147, 2005.
12. Tang J., Li J., Liang B., Huang X., Li Y., Wang K., Using Bayesian Decision for Ontology Mapping. Journal of Web Semantics, 4(1):243-262, 2006
13. Iv_ancsy, R. and Vajk, I, Frequent pattern mining in web log data. Acta Polytechnica,2006.
14. Lars Hernquist. Vectorization of tree traversals. J. Comput. Phys., 87:137–147, March 1990.
15. B. Glimm, I. Horrocks, B. Motik, and G. Stoilos. Optimising ontology classification. In Proc. of the 9th Int. Semantic Web Conf. (ISWC 2010), pages 225–240, 2010.
16. D. Tsarkov. Improved algorithms for module extraction and atomic decomposition. In Description Logics, 2012.

AUTHOR(S) PROFILE



Sasanka Bhimavaram, has completed M.Sc from Sri Venkateswara University, Tirupati, Andhra Pradesh, India. Now Pursuing Ph.D in the department of Computer Science, Sri Venkateswara University, Tirupati, India. He Published research papers in International Journals. His areas of interests include Data Mining, Web Technology, Semantic Web Mining, Cloud Computing, etc.



Dr. P. Govindarajulu, worked as a professor in the department of Computer Science, Sri Venkateswara University, Tirupati. He has published several research papers in various national, International journals, conferences and workshops. His research areas of interests include Image Processing, Data Mining, Cloud Computing, Web Mining, etc.