International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 04 | Apr-2016 www.irjet.net p-ISSN: 2395-0072

EFFECTIVE SEMANTIC CLASS LEARNING FROM THE WEB WITH ONTOLOGY

[*1]Mr. Venkataramanan .K [*2]Dr.Arutchelvan .G

*[*1]M.Phil Research Scholar, Department of Computer Science Adhiparasakthi College of Arts and Science , Kalavai, TamilNadu, India.*

*[*2]M.Phil Guide, Department of Computer Science Adhiparasakthi College of Arts and Science, Kalavai, TamilNadu, India.*

Abstract - ***In this paper we introduce a novel unsupervised ontology learning approach, which can be used to automatically derive reference ontology from a corpus of web services for annotating semantically the Web services in the absence of core ontology. Our approach relies on shallow parsing technique from natural language processing in order to identify grammatical patterns of web service message element/part names and exploit them in construction of the ontology. The generated ontology is further enriched by introducing relationships between similar concepts. The experimental results on a set of global Web services indicate that the proposed ontology learning approach generates an ontology, which can be used to automatically annotate around 52% of element part and field names in a large corpus of heterogeneous Web services.***

Key Words: *Ontology Learning, Web Services Annotation, NLP. Heterogeneous web services, Ontology based webservices, RDF.*

I. INTRODUCTION

The vision of web service technology to expose functionality of on-line services for system-to-system communication has resulted in deployment of considerable number of services on the Web. At the same time the Semantic Web initiative has provided methods, tools and knowledge structures for processing semantically enriched web services. Unfortunately, due to complexity of providing semantic information to web services, the visionary view of semantic web services has not been well accepted neither by industry nor governmental sector where semantic web services technologies could have the major impact. Hence vast majority of public web services lack semantic information and this, complemented with the increasing number of available web services, is the main obstacle in using semantic technologies either for exploiting or analyzing the existing web services. In the absence of core ontologies, annotation of existing web services is dependent on ontology development and ontology learning techniques.

The latter refers to applying machine learning techniques for automatic discovery and creation of ontological knowledge [12]. In addition to outstanding ontology development obstacles [12] (being time-consuming and Labor-intensive), ontology acquisition at the Web scale, such as we are aiming for, imposes extra burden primarily due to the large dataset size, heterogeneity of data and dynamicity of Web. Moreover, ontology acquisition solely from web service descriptions is a resource-demanding and error-prone task since majority of WSDL elements, which need annotations, lack textual documentation (around 95% of elements in our collection of ca 15 000 WSDL documents have no human-readable documentation attached). Furthermore, often the syntax of WSDL element names does not convey correct and complete picture of underlying semantics [1]. There have been efforts [14] [7] both in academia and industry to invent solutions for (semi) automatically annotating existing web services with standard semantic descriptions. The applicability of such solutions is hampered mainly by the annotation cost, as reported by Küngas and Dumas [2].

In this paper we first propose an unsupervised method for domain-independent ontology learning from web services corpus derived from a set of WSDL documents describing available Web services. The main purpose of the constructed ontology would be to facilitate semantic annotation of WSDL documents and XML schema for further analysis and usage of the Web services. The recall and precision of our approach is enhanced by utilization of natural language processing (NLP) techniques and linguistics resources (thesauri and acronym tables). The constructed ontology is then used to support automated annotation of data structure definitions in XML Schema and Web service interfaces in WSDL documents by using the heuristic-based automated annotation as proposed by Küngas and Dumas [2]. One of the specific applications of

the constructed annotations is to increase the quality of Web services match-making. Matching of web services can then be used either during composition of new or analysis of existing web services.

The rest of this paper is organized as follows. In Section 2 we introduce our ontology development methodology and discuss requirements and design issues for the reference ontology. In Section 3 we present our ontology learning method, whereas the evaluation results are captured in Section 4. Finally, Section 5 reviews related work, while conclusions and discussion on future work are presented in Section 6.

II. ONTOLOGY DEVELOPMENT METHODOLOGY

Our ontology development methodology is inspired from the ROD model proposed by Zhou [12] and is an incremental methodology consisting of multiple iterations. Accordingly, the methodology is based on a cycle of three consequence phases: design, learning and validation. While the ontology design phase involves identification of domain resources and analysis of requirements, the ontology learning phase embodies the core ontology learning and construction techniques. Ontology validation and evaluation of the generated ontology is the last phase of the cycle. After completion of a cycle and inspection of results, we will attempt to improve the results by integrating the resulting ontology with other ontologies and incorporating extra domain resources before executing operation iteration.

In the ontology design phase, we identify the objectives and requirements for the target ontology, and determine applicability of relevant domain resources in our case. In addition, to comply with general characteristics of an ideal ontology [13] (e.g. clarity, coherence, extendibility, etc), the target ontology needs to satisfy the following requirements with respect to objectives of web services analysis:

1. To maximize interoperability among web services (i.e. to increase number of matching web services);
2. To maximize the quantity of annotated web service elements;
3. To be evolvable and allow incremental ontology learning over time in order to accommodate frequent changes/updates in the web services domain.

III. ONTOLOGY LEARNING PROCESS

We follow a bottom-up approach for ontology learning by starting from processing the WSDL documents and gradually derive top-level ontological concepts and relations. As shown in Fig. 1, the ontology learning process consists of three steps where each step, in turn, is a pipeline of several tasks. The first step is mostly about extraction of relevant textual content and subsequent syntactic refinement, while the second step exploits the results of the first step to infer ontological concepts, relationships and instances. The last step deals with organization of the discovered concepts and relationships to improve the quality of discovered knowledge. In the following we explain in detail the activities involved in each step.

3.1 Information Elicitation

a) Term Extraction

Ontology learning from web service description can be performed at different levels of granularity, starting from the finest (XML schema leaf element names which are either of built-in XSD types or defined basic types) until more general levels with operations and services. Resulting ontologies can be used then to annotate the elements at the same level of granularity as the input elements. The focus of this work is based primarily on the finest granularity since once the finest elements of web services are semantically annotated, the resulting annotations can be propagated to coarse-grained elements [2]. Thus, first we will extract the list of fine-grained element names of the whole dataset (a corpus of WSDL documents). Next, out of the extracted list we choose a subset of most frequently presented element names, as proposed by Küngas and Dumas [2], for seeding ontology learning process. The extracted terms usually consist of multiple words (compound words or phrases).

b) Syntactic Refinement

The extracted terms may contain punctuations, shortened words, and abbreviations, misspelled and irrelevant words. Thus we need to normalize the terms to improve the quality of identified ontological concepts and relations in the generated ontology. Syntactic refinement task is constructed by using the following methods:

1. *Term Tokenization.* In context of schema leaf nodes, the extracted terms usually follow Camel case or Pascal case form, or separated by underlines and punctuations, which facilitate the tokenization process. We use these conventions for segmentation of terms into constituting tokens. The irrelevant words (such as single characters) and non-alphanumeric characters are also eliminated from the set of discovered tokens.

2. *Cleavage of Shortened Words and Abbreviations.* A term or the constituting words may refer to a domain terminology reflected as shortened word (e.g. pwd stands for password) or abbreviation (such as ASIN stands for Amazon Standard Identification Number). We utilize an auxiliary table for resolving such words into their corresponding complete syntactic forms.

3. *Known Compound Noun Determination.* The purpose of this method is to discover compound nouns (a sequential combination of two or more words) which convey a special meaning in general (e.g. first name) or in the underlying domain (e.g. login name). The compound nouns are treated as single units in the all subsequent word processing stages.

4. *Word Lemmatization*. Next, the words are transformed into their lemmas in order to look up them in the dictionary and later to provide unified naming conventions for labeling identified ontological concepts and relationships. Words which do not exist in dictionary are marked as stop words.

3.2 Ontology Discovery

This step concerns with exploited techniques, resources and tools for identifying the ontological concepts and relations from those set of refined terms resulted from previous step. The outcome of this step is our preliminary knowledge base (ontology + respective instances). The step consists of the following tasks.

a) Pattern-based Semantic Analysis

We exploit the syntactic regularity patterns observed when composing a term out of multiple words. According to this observation vast majority of web service element names are noun phrases [1] while around 79% of the noun phrases in English language can be classified into one of two following patterns as reported by [3]:

- Pattern#1: $(\text{Noun}_1)+ \ldots+(\text{Noun}_n)$

e.g. *CustomerId*

- Pattern#2: $(\text{Adjective}_1)+\ldots +(\text{Noun}_n)$

e.g. *SupportedImageType*

The patterns are highlighting the grammatical role of constituting words and their part of speech act. Both types of information (grammatical role and part of speech) can be extracted by employing grammatical dependency parser tools (e.g. MINIPAR tool [9]), which in addition discover dependency relationships of a given phrase. A dependency relationship is an asymmetric binary relation between a word, called *head,* and another word (or a set of words) named *modifier*, where *head* reveals the most emphasized word of the phrase (e.g. *Customer Id* would be resolved to pair {*head*: Identifier, *modifier*: Customer}). From dependency parser perspective, the above-mentioned patterns are identified as follows:

- **Pattern#1:** $(N|\text{Word}_n)$ $[(nn)(N|\text{Word}_1) + .. +(nn)(N|\text{Word}_{n-1})]$
- **Pattern#2:** $(N|\text{Word}_n)$ $[(mod)(A|\text{Word}_1)+\ldots+(nn)(N|\text{Word}_{n-1})]$

In these patterns the first part, which is placed inside parentheses, refers to *head* of the phrase while the *modifier* segment is placed inside square brackets. Moreover, the words are annotated with their part of speech act (*N* for a noun and *A* for an adjective) and also grammatical roles (*mod* for an adjective relation, *nn* for a noun-noun relation). We harvest only terms complying with Pattern#1 or Pattern#2 as they provide the main ingredients for ontology learning.

b) Term Disambiguation

If all words in a term are determined as stop words, or when the *head* part is not a noun, then the term is considered as a vague term. These kinds of terms are disambiguated by replacing terms with respective operation names in WSDL from which input /output element names the particular term was extracted from. If the term appears in multiple operations, then we replace the single term with a concatenation of multiple operation names. We repeat the entire syntactic processing stages with new content but this time we simply discard ambiguous terms.

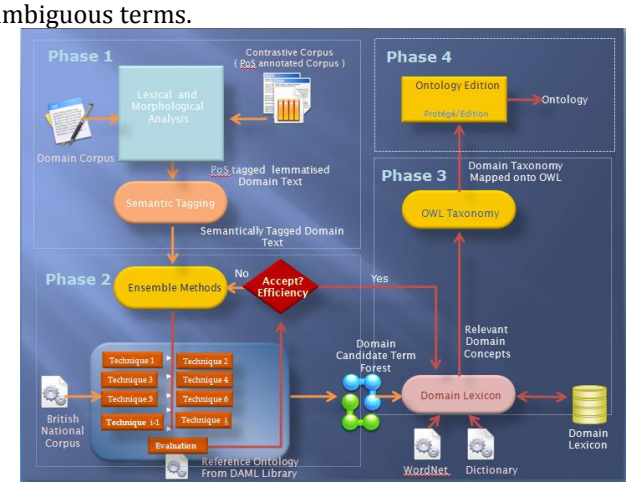

Fig. 1: Ontology Learning Steps

c) Class and Relation Determination

We rely on the following rules (Rule-1 and Rule-2) to exploit output of dependency parsing of each term to capture ontological classes and object property relationships. Construction of these rules is based on the following observations:

- According to Bourigault and Jacquemin [5] single-word terms denote broader concepts than multi-word terms. They appear more frequently in corpora and are therefore more appropriate for statistical clustering. In contrast to single-word terms that are too ambiguous and too generic, multi-word terms are more interesting for ontological motivation as they present finer concepts in domains. As single-word terms denote broader concepts than multi-word terms, and a compound noun inherits most of its semantic from its *head* [4], then we assume that the

concept representing the *head* word subsumes the concept generalizing the entire term. Moreover, since *head* words cannot be decomposed further, we will regard them as concrete concepts in the ontology.

- The relation between *head* and *modifier* segment in case of *noun-noun (nn)* relationship resembles from grammatical point of view a kind of possessive authority for the *head* segment over an entire term. Based on this observation, has*Property* relationships among discovered concepts are asserted similarly to Guo et al.

Based on the aforementioned observations we introduce the following ontological concept identification rules:

Rule-1: *Terms, which are subject to Pattern#1 are initiating the assertion of following ontological concepts and relationships:*

- *Word*$_1$ hasProperty *Term,*
- *Term* subclassOf *Header.*

In Rule-1, *Term*, *Word*$_1$ and *Header* are all referring to concepts in an ontology. For example, term *SessionKeyIdentifier* complies with Rule-1, so the following axioms are added to the ontology: 1) *Session* isA *Class*, 2) *SessionKeyIdentifier* isA *Class*, 3) *Identifier* isA *Class*, 4) *Session* hasProperty *SessionKeyIdentifier, 5) SessionKeyIdentifier* subClassOf *Identifier.*

Rule-2: *Terms, which are subject to Pattern#2 are initiating the assertion of following ontological concepts and relationships*:

- *Term* subClassOf *Header*

In Rule-2, *Term* and *Header* are referring to concepts in an ontology. For example, term *SupportedType* complies with Rule-2, so the following axioms are added to the ontology: 1) *Type* isA *Class*, 2) *SupportedType* isA *Class, 3) SupportedType* subClassOf *Type.*

Using Rule-1 and Rule-2, we will generate an ontology automatically from the corpus of element names extracted from a set of web services descriptions in WSDL. In the last step, the initial set of (original) terms extracted from a collection of web service descriptions are assigned to their respective ontological representation as individuals.

3.3 Ontology Organization

In order to improve the quality and usability of generated ontology, the resulting ontology is investigated to determine extra relationship between concepts or to remove the redundant ones. In this work, we utilize lexical similarity between labels of ontology classes and augment the ontology with a*ssimilator* relationship indicating that the classes on both side of this relationship convey a similar lexical semantic .We employ WordNet digital dictionary [10] and a WordNet lexical similarity library [8] to measure similarity between labels. We adopt an unsupervised agglomerative clustering approach to obtain clusters of similar classes. The clustering algorithm starts by putting every single data point (label of each concept) in one cluster to set up the initial clusters. Then, it measures pair-wise similarity distance of data points in one cluster against those belonging to other clusters and at each step merges two closest (most similar) clusters. The clustering process finishes whenever a single cluster remains or the similarity distance between two closest clusters does not meet a threshold. During our experiments, while manually evaluating the resulting ontologies, we observed that a threshold value of 85% is the minimum reasonable distance value. The similarity distance between two clusters is based on average dictionary-based affinity between entries of two clusters. The complexity of distance computation is of $O(n^2)$ due to need of cross-examination of each entry in one cluster against those in other clusters. From lexical analysis point of view, entries within a cluster are forming a synonym set. Thus, concepts in one cluster are pair-wise augmented with *is Similar To* relationship (e.g. Image *isSimilarTo* Picture).

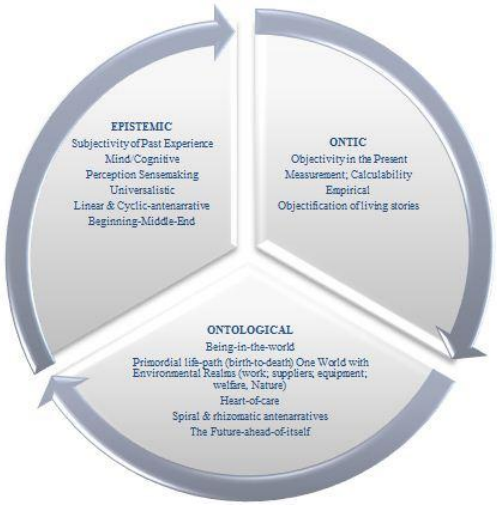

Fig. 2: Life-Path of Ontology Organizations

IV. ONTOLOGIES ON THE SEMANTIC WEB

We present an algorithm that provides natural language (NL) paraphrases for OWL Ontologies on the Semantic Web Our goal is to ensure both fluency (readability) and accuracy of the output , in terms of preserving the meaning conveyed by its description logic formalism. The approach described is a generic domain-independent one,and is completely automated.

With the advent of OWL, and its subset OWL-DL, semantic web content is backed by a precisely-defined Description Logic (DL). This property means that the meaning of semantic web content will always be clear and potentially useful to an intelligent agent, or reasoner-equipped software application. However, concept definitions (OWL

Classes) are specified in the language of logic, requiring humans to understand this logical language in order to decipher the meaning of concepts. For end users of semantic web enabled applications, this may pose a usability problem in many important circumstances, effectively creating a barrier for entry into the semantic web. To remove this barrier, we have designed and implemented a procedure for generating near Natural Language (NL) paraphrases.

For a procedure such as ours to be widely useful, it has to be not only robust but also domain-independent, able to work with a large number of the concepts and ontologies available. A domain-independent solution is desirable because it can immediately make use of the numerous OWL ontologies that already exist, modeling everything from clinical and en- vironmental information (e.g., NCI and JPL) to personal interests andrelationships (e.g., FOAF).

APPLICATIONS

Semantic Annotation: Recently, numerous tools for semantically annotating text, images, video etc have been developed. Most of these tools use ontologies for driving the annotation process, allowing users to link their data with entities in the ontology. In order to support accurate and speedy annotation, NL description of the classes can be provided in order to explain the meaning of the con cept and to point out its correct usage.

Web-Service Advertising: OWL-S based semantic web-services advertise themselves as instances of the service-profile. Rendering NL paraphrases of these service-profile instances can make web-service descriptions more accessible to end-users.

Web-Policy (/Rules) Description: In, the authors showed that

Web-Service policies can be represented in OWL (using syntactic sugar rules). However translating the WS-Policy operators (wsp:All, wsp:ExactlyOne) in OWL produced some non-trivial, complex class expressions. Policy developers new to OWL might find it difficult to specify constraints and capabilities of their web services when working with these class expressions. NL paraphrases of the policies will make their meaning more accessible, thereby reducing the possibility of error, without losing the intended semantics.

V EVALUATION

The proposed ontology learning mechanism is implemented in Java by utilizing WordNet 3.0 as our reference dictionary, JWSL [8] library for measuring similarity between words, and MINIPAR dependency parser [9] for identifying the patterns. The generated ontology is represented in OWL format. During our experiments we used the set of ca 15000 WSDL documents from http://www.soatrader.com/web-services as a representative set of Web services. The evaluation data-set includes a sample subset of the services such that the frequency of input and output element names covers 20% (1858 unique terms) of names in the entire collected dataset. In order to validate correctness of the generated ontology, we manually constructed an ontology, using a methodology developed by Küngas and Dumas [2]. We refer to this handcrafted ontology as *golden ontology* in the rest of this paper.

As the generated ontology should also satisfy web service analysis requirements, we need to perform ontology evaluation from two perspectives. First, from ontology perspective we evaluate general ontological properties and validate the quality of a generated ontology against the golden ontology. Second, from web service annotation perspective we examine the quality and quantity of annotated web services using automatically generated ontology with respect to the golden ontology. We leave the latter evaluation case for the future work and focus on the former perspective in the rest of this paper. We perform evaluation of the automatically constructed ontology in two stages. While in the first stage, evaluation is performed over ontological classes, in the second stage ontological instances (WSDL/XSD leaf node elements) are used in evaluation. Fig. 2 presents the number of concepts and their instances in the automatically generated ontology and the golden ontology as well as the quantity of linguistically common concepts between the two ontologies and their instances.

5.1 Concept-level Comparison

Out of 1853 unique terms in our evaluation data-set, our ontology learning system managed to process 1601 terms and assign them to their representative concepts (1813 concept) while the rest of the terms were ignored due to different reasons (i.e. containing meaningless names, not complying with the determined patterns, etc). Clearly the number of concepts in generated ontology is larger with respect to the number of concepts in the golden ontology, since in our approach new concepts emerge due to following reasons. First, as a result of measuring linguistic and dictionary-based differences between underlying terms rather than considering actual semantics of terms (e.g. “legalDisclaimer” and “TermsAndConditions” where both convey same meaning while their ontological representation leads to several classes). Second, ontological concept discovery rules (Rule-1, Rule-2) break down a compound noun into several interrelated concepts. Hence, proportionally larger number of concepts is expected to be generated by our system compared to the number of instances.

For concept-level ontology comparison we exploited Falcon-AO [11], which aligns ontologies in two phases: first linguistic then structural (graph) matching. Authors of

Falcon-AO have pointed out that their tool cannot use structural information for ontology alignment purpose if the underlying ontology is very large such as in our case. Hence, the ontology alignment result produced by Falkon-AO solely represents linguistics similarity between aligned ontologies. While the percentage of similar concepts over the two ontologies is only about 62% with respect to the concepts in the golden ontology, the number of instances captured by those common concepts is relatively high around 71% (1313 instances out of 1853). Since instance level evaluation can be performed over more than two third of the entire data set, a reasonable assessment can be expected despite of deficiencies of concept matching. Because the concepts in the golden ontology are not augmented with any other relations (i.e. object properties), we did not perform any comparison at this level.

5.2 Instance-level Comparison

For instance-level comparison we compute precision (P) and recall (R) metrics to show the quality of our term classification approach only for those instances, which are assigned to the common concepts, which are presented by the last grey column in Fig. 2. Let's consider E as the set of instances belonging to those common concepts in a golden ontology, C (correct) as the number of instances in set E, which are classified correctly in the generated ontology, I (incorrect) as the number of instances in E which are misplaced (i.e. they are not assigned to the same concepts as instructed by the golden ontology) and M (missing) as the number of instances which appear in set E but they are not classified under the common concepts in generated ontology. Based on these definitions, we compute the precision and recall as following:

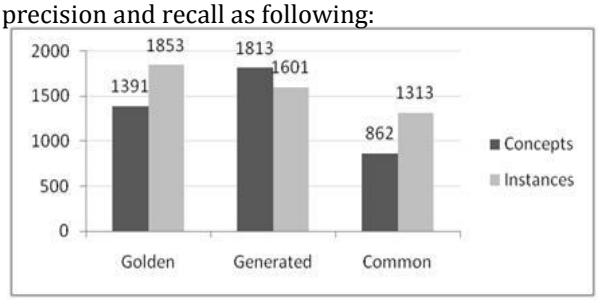

Fig 3: Quantity of concepts & instances in generated ontology compared to golden ontology.

The size of E in our system is 1313, which equals $C+M+I$ where C = 968, I = 60 and M = 278. According to (1), we have precision of 85% and recall of 78%. High precision is achieved due to the syntactic quality of terms, following the syntactic patterns which we used during harvesting, and finally complying with respect to dictionary meanings in WordNet [10]. The quality of syntactic processing is also boosted by utilization of an auxiliary table to uncover known acronyms and compound nouns, which consequently improves both recall and precision.

VI RELATED WORKS

Proposed mechanisms for (semi-)automatic annotation and matching of web services are aiming for machine learning techniques and they diverge in availability of external resources, training data sets, quality and quantity of dataset and main purpose of annotation. Some machine-learning-based approaches such as [7] ,proposed by Heβ et al. need initially to train their system in order to generalize (semantic of training data) and predict semantic labels for (similar) unseen web services.

As we target a large repository of absolutely not-annotated ad-hoc web services from different domains, applicability of such techniques is not clear. Similarly to our approach, Guo et al. [1] leveraged relation between words in phrases to establish ontological relationships between acquired concepts. While the authors tackle pair-wise service matching solution by aligning the generated ontology fragments, we intend to create an ontology to be utilized for analysis of web services. In addition, Guo et al. [1] take advantage of active domain experts and knowledge of web services domains in annotation. Neither of these two resources are practically available in our case due to size of the data set and lack of additional meta-knowledge about services.

In a slightly similar work, Sabou et al. [6] described an automatic extracting method that learns domain ontologies from textual documentation attached to web services. Due to the fact that around 95% of web services in our data set come with no textual documentation, the applicability of their approach is not applicable in our case.

Several tools for semantic annotation of web services and transformation to semantic web service representations such as OWL-S by ASSAM [7], and WSDL-S/SA-WSDL [15] by Radiant [14] have been proposed. The aforementioned tools follow a semi-automatic approach for selecting the most appropriate domain ontology (for annotation purpose) and then mapping WSDL elements to respective ontological concepts. Due to explicit expert user intervention and reliance on pre-determined domain ontologies for annotation purpose, applicability of such solutions for large-scale annotation of web services is impractical despite of the fact that these solutions tend to provide high-quality annotations.

VII CONCLUSIONS AND FUTURE WORK

In this paper we presented an ontology learning approach to be used for matching web services in large scale in the absence of core ontology. The preliminary results show that the generated ontology captures correctly around 52% of entire data set, hence, providing a reasonable basis

for web services matching in practical solutions. Our approach generates ontologies, which can be used for automated construction of annotation heuristics such as used by Küngas and Dumas [2] in their semi-automatic cost-effective semantic annotation methodology for web services interfaces. Thus one of the contributions of the ontology learning approach presented in this paper is to reduce the number of man-hours required in a cost-effective annotation scheme even further. As a future work we are planning to enhance the proposed ontology learning approach such that better coverage of annotations could be achieved automatically.

We have presented an algorithm that generates concise, accurate NL paraphrases for OWL Concepts based on a variety of NLP techniques and implemented it in an ontology engineering toolkit, SWOOP. We have conducted a promising preliminary user evaluation, and plan to conduct formal user studies to fully evaluate the contribution of our work.

REFERENCES

1. Guo, H., Ivan, A., Akkiraju, R., Goodwin, R.: Learning Ontologies to Improve the Quality of Automatic Web Service Matching, In: IEEE Int.Conference on Web Services (ICWS 2007), pp. 118--125 (2007).
2. Küngas, P., Dumas, M.: Cost-Effective Semantic Annotation of XML Schemas and Web Service Interfaces. In: IEEE Int. Conference on Services Computing, IEEE Computer Society, pp. 372--379 (2009).
3. European Collaborative Clamour Project: Linguistic Work Package Report.: http://www.statistics.gov.uk/methods_quality/clamour/coordination/downloads/Clamourdec2000IRn2.doc (2000).
4. Lieber, R.: Morphology and Lexical Semantics. Cambridge University Press (2004).
5. Bourigault, D., Jacquemin, C.: Term extraction + term clustering: an integrated platform for computer-aided terminology. In: 9th Conference on European Chapter of the Association for Computational Linguistics, pp. 15--22, ACL, Morristown, NJ, (1999).
6. Sabou, M., Wroe, C., Goble, C., Mishne, G.: Learning domain ontologies for Web service descriptions: An experiment in bioinformatics. In Proceedings of the 14th international Conference on World Wide Web, pp. 190--198, ACM, Japan (2005).
7. Heß, A., Johnston, E., Kushmerick, N.: ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services. LNCS, vol. 3298, pp. 320--334, Springer (2004).
8. Pirrò, G., Seco, N.: Design, Implementation and Evaluation of a New Similarity Metric Combining Feature and Intrinsic Information Content. LNCS, vol.5332, pp. 1271--1288, Springer-Verlag (2008).
9. Lin, D.: Dependency-based Evaluation of MINIPAR. Workshop on the Evaluation of Parsing Systems, First Int. Conf. on Learning Resources and Evaluation, Spain (1998).
10. Miller, G. A.: WordNet: A Lexical Database for English. Communications of the ACM, Vol. 38, No. 11: 39--41 (1995).
11. Hu, W., Qu, Y.: Falcon-AO: A practical ontology matching system. Web Semantic, vol. 6, no. 3, pp. 237--239 (2008).
12. Zhou, L.: Ontology learning: State of the art and open issues. Information Technology and Management, vol. 8, pp. 241--252 (2007).
13. Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing, Journal of Human Computer Studies, vol. 43, no. 5/6, pp. 907--928 (1993).
14. Radiant: WSDL-S/SAWSDL Annotation Tool. http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1.
15. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: SAWSDL: Semantic Annotations for WSDL and XML Schema, In: IEEE Internet Computing, vol. 11, No. 6, pp.60--67, (2007).