ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services

Andreas Heß and Nicholas Kushmerick

Computer Science Department, University College Dublin, Ireland
{andreas.hess, eddie.johnston, nick}@ucd.ie
http://moguntia.ucd.ie/projects/annotator

1 Introduction

The vision of Semantic Web Services is to provide the means for fully automated discovery, composition and invocation of loosely coupled software components. One of the key efforts to address this "semantic gap" is the well-known OWL-S ontology [1].

However, software engineers who are developing Web Services usually do not think in terms of ontologies, but rather in terms of their programming tools. Existing tools for both the Java and .NET environments support the automatic generation of WSDL. We believe that the semantic service web will flourish only when similar tools existed to (semi-) automatically generate OWL-S or a similar form of semantic metadata.

In this demo we present a tool called ASSAM—Automated Semantic Service Annotation with Machine Learning—that addresses these needs. This extended abstract is structured as follows: In the next section, we will introduce the ASSAM annotator application. In section 3, we describe the machine learning algorithm behind ASSAM. Finally, we present some related work and discuss planned future extensions to our current application.

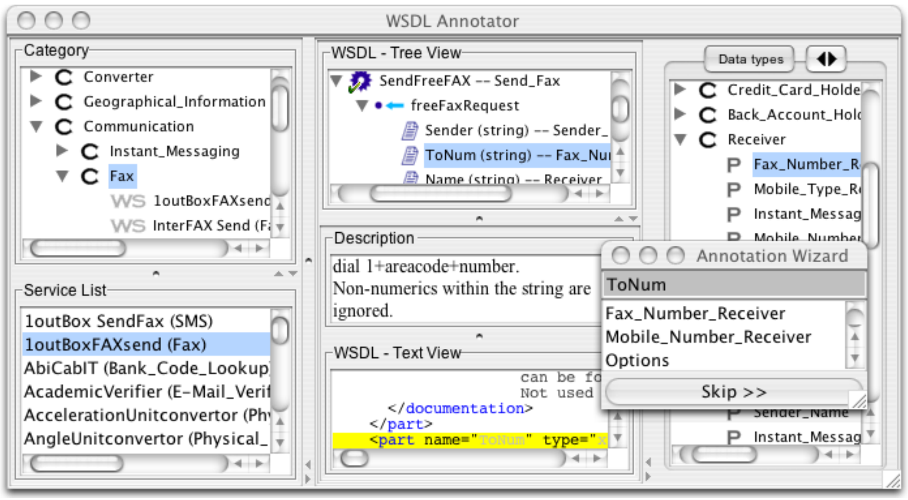
2 ASSAM: A Tool for Web Service Annotation

The ASSAM WSDL annotator is a tool that enables the user to semantically annotate a Web Service using a point-and-click interface. The key feature of ASSAM is the ability to suggest which ontological class to use to annotate each element in the WSDL. The recommendations are based on a machine learning algorithm.

Use Cases. ASSAM is designed primarily for users who want to annotate many similar services. Typically, these will be end users wanting to integrate several similar Web Services into his or her business processes. But the annotation task might also be performed by a centralized semantic Web Service registry.

Our tool could also be useful for programmers who are only interested in annotating a single Web Service they have created.[1] In order to make his or her

[1] Thanks to Terry Payne who pointed out this use case.



dial 1+areacode+number. Non-numerics within the string are ignored.

```
                    can be fo
                    Not used
      </documentation>
    </part>
    <part name="ToNum" type="x
```

Fig. 1. ASSAM uses machine learning techniques to semi-automatically annotate Web Services with semantic metadata.

service compatible with existing services, a developer might want to annotate it with the same ontology that has already been used for some other Web Services. The developer could import the existing Web Services in ASSAM and use them as training data in order to obtain recommendations on how to annotate his or her own service.

Functionality Fig. 1 shows the ASSAM application. Note that our application's key novelty—the suggested annotations created automatically by our machine learning algorithm—is shown in the small pop-up window.

The left column in the main window contains a list of all Web Services currently and the category ontology. Web Services can be associated with a category by clicking on a service in a list and then on a node in the category tree. When the user has selected a service and wants to focus on annotating it this part of the window can be hidden.

The middle of the window contains a tree view of the WSDL. Port types, operations, messages and complex XML schema types are parsed from the WSDL and shown in a tree structure. The original WSDL file is also shown as well as plain text descriptions from the occasional documentation tags within the WSDL or a plain text description of the service as a whole, such as often offered by a UDDI registry or a Web Service indexing web site.

Note that in the ASSAM GUI the parts that are not used can be hidden. During annotation of a single service, the user can for example hide the category ontology and the list of all services. Also, if the user is not interested in the WSDL code, this view can be hidden, too.

Once the annotation is done it can be exported in OWL-S. The created OWL-S consists of a profile, a process model, a grounding and a concept file if complex types where present in the WSDL. Note that this also includes XSLT transformations as needed in the OWL-S grounding to map between the traditional XML Schema representation of the input and output data and the OWL representation.

Limitations. Because we do not handle composition and workflow in our machine learning approach, the generated process model consists only of one atomic process per operation. For the OWL-S export, we do not use the annotations for the operations at the moment, as there is no direct correspondence for the domain of an operation.

3 Machine Learning

The key feature of the ASSAM annotator is the annotation assistant that suggests annotations to the user based on a machine learning algorithm. We cast the problem of classifying the components of a Web Service as a text classification problem. Our tool learns from Web Services with existing semantic annotation. Given this training data, a machine learning algorithm can generalize and predict semantic labels for previously unseen Web Services. We use an iterative relational classification algorithm to exploit the internal structure of a Web Service in a way to improve classification accuracy: For example, the list of possible datatypes is dependent on the category of the Web Service as a whole.

In a mixed-initiative setting, the suggestions made by the assistant do not have to be perfectly accurate to be helpful. In fact, the classification task is quite hard, because the domain ontologies can be very large. But for that reason it is already very helpful for a human annotator if he or she would have to choose only between a small number of ontological concepts rather than from the full domain ontology.

For a more detailed explanation of our algorithms and experimental results, the reader is referred to our paper at the ISWC main conference [2] and our paper at the European Conference on Machine Learning [3].

4 Related Work

Paolucci et al addressed the problem of creating semantic metadata (in the form of OWL-S) from WSDL [4]. However, because WSDL contains no semantic information, this tool provides just a syntactic transformation. The key challenge is to map the XML data used by traditional Web Services to classes in an ontology.

Currently, Patil et al [5] are also working on matching XML schemas to ontologies in the Web Services domain. They use a combination of lexical and structural similarity measures. They assume that the user's intention is not to annotate similar services with one common ontology, rather they also address the problem of choosing the right domain ontology among a set of ontologies.

Sabou [6] addresses the problem of creating suitable domain ontologies in the first place. She uses shallow natural language processing techniques to assist the user in creating an ontology based on natural language documentation of software APIs.

Sudhir Agarwal et al. [7] created a tool that assists the user in annotating semantic Web Services by using plain text descriptions found on web sites describing the service.

5 Future Development

We are planning various improvement for the next version of ASSAM to reach production quality. For example, at the moment, we do not do inference checks on the annotations. Also, as mentioned above, OWL-S defines atomic processes in terms of preconditions and effects. Instead of assigning a domain to an operation, future versions of ASSAM could instead assign preconditions and effects. However, the techniques used in ASSAM are independent of the actual syntax of the modeling language that is exported. Future versions could also export different formats like WSMO/WSML. We are also planning to include our OATS algorithm as described in [8] in future versions of ASSAM.

Acknowledgments. This research was supported by grants SFI/01/F.1/C015 from Science Foundation Ireland, and N00014-03-1-0274 from the US Office of Naval Research.

References

1. The DAML Services Coalition: OWL-S 1.0. White Paper (2003)
2. Heß, A., Johnston, E., Kushmerick, N.: Assam: A tool for semi-automatically annotating semantic web services. In: 3rd International Semantic Web Conference, Hiroshima, Japan (2004)
3. Heß, A., Kushmerick, N.: Iterative ensemble classification for relational data: A case study of semantic web services. In: European Conference on Machine Learning, (Pisa, Italy)
4. Paolucci, M., Srinivasan, N., Sycara, K., Nishimura, T.: Towards a semantic choreography of web services: From WSDL to DAML-S. In: Int. Conf. for Web Services. (2003)
5. Patil, A., Oundhakar, S., Sheth, A., Verma, K.: Meteor-s web service annotation framework. In: 13th Int. WWW Conf., New York, USA (2004)
6. Sabou, M.: From software APIs to web service ontologies: a semi-automatic extraction method. In: 3rd International Semantic Web Conference, Hiroshima, Japan (2004)
7. Agarwal, S., Handschuh, S., Staab, S.: Surfing the service web. In: 2nd International Semantic Web Conference, Sanibel Island, FL, USA (2003)
8. Johnston, E., Kushmerick, N.: Aggregating web services with active invocation and ensembles of string distance metrics. In: 14th International Conference on Knowledge Engineering and Knowledge Management, Whittlebury Hall, Northamptonshire, UK (2004)