Corpus-based Web Services Matchmaking

Ziming Zhuang, Prasenjit Mitra, Anuj Jaiswal

School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA
{zzhuang, pmitra}@ist.psu.edu, ajaiswal@psu.edu

Abstract

Discovering Web services and clustering them based on their functionalities are important problems with few existing solutions. Users may search for Web services using keywords and receive services that semantically match the keywords. Semantic Web service matchmaking, proposed to enhance the precision of matchmaking using syntactical cues, is generally based upon semantic service descriptions in ontology markup languages as add-ons or replacements to the underlying WSDL descriptors. Ways to improve the performance of direct matchmaking in WSDL, however, remains less studied. In this paper, we introduce a novel corpus-based method to facilitate matchmaking in WSDL files. We show that our method can identify semantically similar Web services with satisfactory recall.

Introduction

Web services are defined as self-contained and modular applications that are accessible via the Web and provide a set of functionalities to either businesses or individuals [Tsalgatidou and Pilioura, 2002]. Today the ubiquity of the World Wide Web, combined with the reusable software component paradigm, has rendered Web services the ideal medium to facilitate collaboration and integration among service providers.

An effective and efficient service discovery mechanism is critical to employ the full potential of Web services. An established solution is the WSDL/UDDI combination which is supported and implemented by some of the major players in the industry. This solution is mostly a keyword-based WSDL repository that supports registration and search through the UDDI interface. However, because the underlying rationale of this solution is simple pattern matching on certain fields of the WSDL entries that do not contain semantics, it relies heavily on the shared taxonomy between service providers and consumers. Semantic-rich solutions, such as OWL-S, are thus proposed to mitigate such problems by offering formal web service descriptions of service specifications. Yet, there remains a lot of work to be done in order to bridge the gap between the widely implemented WSDL/UDDI registries and the more advanced semantic Web service discovery mechanism.

In this paper, we propose a novel corpus-based Web-services' matchmaker directly targeting the WSDL descriptions. This is an effort to enhance the precision of Web service discovery using the well-established WSDL/UDDI architecture, without involving an additional level of semantic markup.

Related Studies

A series of studies have suggested that WSDL is not able to describe sufficiently the semantic aspect of Web services. [Tsalgatidou and Pilioura, 2002] argues that although WSDL/UDDI contains certain semantic information about the service providers and syntactic information about the services, the discovery mechanism itself is not elaborated, mainly based on exact pattern matching on keywords. [Zhang et al., 2002] points out three existing problems in the current WSDL/UDDI discovery mechanism: insufficient results due to simple search rationale, the level of precision highly depends on services categorization and no support for search across multiple UDDI registries.

A number of efforts have been put forward to maximally exploit the implicit semantic information resided in the current WSDL/UDDI infrastructure. [Dong et al., 2004] describes *Woogle*, the system to offer more refined Web services similarity search capability. The system first clusters the parameter names into semantically meaningful concepts, which are then used to decide the similarity between inputs/outputs sets and operations. [Kawamura et al., 2004] presents another deployed system, in which a series of four filters are used to refine the set of search results. These filters cover the usage of service namespaces, textual information, inputs/outputs of services, and the subsumption relationship of the I/O constraints.

OWL-S represents the efforts to specify a computer-interpretable semantic markup language based upon which to implement an ontology-enhanced search interface that can be used for automatic service discovery, e.g. [Paolucci et al., 2002] and [Bansal and Vidal, 2003]. Although the OWL-S based approaches show promising results and are in accord with the bigger picture of the Semantic Web, one major disadvantage of such approaches is that, because they are not supported by the currently established WSDL/UDDI infrastructure that is already supported by many software and tools, they require additional

translation processes or may even have to create a new Web services registry from scratch.

Corpus-based Matchmaking

Our algorithm is based on an algorithm for identifying the semantic similarity between the two WSDL files. Using the Web as an effective corpus, the Web service matcher looks at all possible pairs of elements from the two WSDL files it is matching and assigns a similarity score to each pair; based upon such scores, it later generates the overall similarity score for the two WSDL files.

The first step in the process is to use a word-similarity table, generated by a word relater (described later in this section), to compute the similarity of the elements based on the similarity of the pairs of words in the elements' names. For example, given two elements "Get Stock Quote" and "Price by Stock", we see that *match(Stock, Stock) = 1.0*. Similarly, we have *match(Quote, Price) = 0.4*. Therefore, we calculate the similarity between the two strings as: match("Get Stock Quote", "Price by Stock") = (1 + 0.4)/2 = 0.7, with the denominator being the number of words in the string with fewer words.

This similarity score of two name strings is then normalized with respect to the highest score generated by the matcher. If the generated similarity score is above the threshold, then the two concepts are said to match. The word-similarity table used in the above algorithm contains the similarity scores of all pairs of words that appear in the element names. To construct this table, we derive the word similarity scores using a corpus-based matching algorithm after checking whether the element names are spelled similarly. A corpus of Web documents belonging to the functional domain of the WSDL files being matched is used as the context for the word relator. Intuitively, the names of the elements that appear in the WSDL should also appear in the Web documents. The word relator calculates word-similarity scores based on the similarity of the context in which the words appear in the documents.

We identify the context in which a word, w, appears by looking at words that appear in a 1000-character neighborhood (500 before and 500 after) of all occurrences of w in documents in the corpus, and chose all words that are complete in this window. The number of rows in the context vector, V_w, of a word w is equal the number of words in the corpus. Let $V_w[i] = c$. This implies that the i^{th} word in the corpus occurs with a frequency c in the 1000-character neighborhood of the word w. The cosine of such normalized context vectors of two words gives a measure of the similarity of context in which the two words appear. We use this similarity measure to generate a table of word similarities that is then used by the linguistic matcher.

Ideally, we would have one corpus associated with one Web service functional domain, where the documents in the corpus use the terms in the exact sense as it is used in the Web service description. We generated a corpus by searching the web via the Google API using 5 keywords describing the functional domain of the Web services that we were about to match. Typically, a corpus of 200 pages proved adequate to produce good matches.

Having generated the similarity scores in pair-wise for all the names of the elements in the two WSDL documents, the next step is to calculate the overall similarity. This score is used to determine whether the two Web services are similar and to what degree. Here, a dampening factor α_i is used to reflect the different importance of each level in the WSDL tree. Intuitively, levels closer to the root are considered more important when calculating similarity scores. For example, the similarity of the "service" level should have more impact than that of the "message" level. So we assign different α_i in order to weight the similarity scores of different levels. In our implementation, we empirically define α_i as follows:

$$\alpha_0 = 0.8 \text{ (for the root level)}$$
$$\alpha_{i+1} = \alpha_i / 2 \text{ (for each additional level)}$$

Due to space constraints, we cannot fully describe the algorithms here. They are available upon request.

The Baseline Matcher

We have implemented a baseline matcher, which bears some similarities to the work presented in [Wang and Stroulia, 2003], to evaluate the performance of the corpus-based system through comparison. In the baseline matcher, we generate similarity scores based upon the lexical references provided by WordNet v2.0[1]. To calculate the overall similarity score for the two WSDL files, the baseline matcher follows a bottom-up approach. By matching all elements (inputs/outputs, messages, operations, and services) in a pair-wise fashion between the two WSDL files, the baseline matcher returns the overall similarity score calculated as the sum of all individual pair-wise scores.

The algorithm is available upon request.

Evaluation and Discussion

70 WSDL files (35 similar pairs) are downloaded from SalCentral Web Services Search Engine[2]; each pair belongs to the same functional domain according to the "Search by Method Names" index. We manually pre-processed the files, cleaning the irregularly named

[1] http://wordnet.princeton.edu/
[2] http://www.salcentral.com/QuickLink.aspx

elements and recovering the abbreviations in tags. Furthermore, multi-words element names are broken up into a series of words. Then these files are used as test data for both the corpus-based matcher and the baseline matcher.

We define recall R as the metrics for the performance of the matchers:

$$R = M_{sim} / N_{sim},$$

where M_{sim} is the number of Web services considered by the matcher to be similar, and N_{sim} is the number of similar Web services. In our experiment, N_{sim} equals to 35.

System	Average similarity score	Recall
Baseline	0.561	0.427
Corpus-based	0.624	0.522

Table 1. Results of evaluation

We use a cut-off value to distinguish between similar and dissimilar pairs. We set it to be 0.5 for both systems. The corpus-based matcher has outperformed the baseline matcher in recall by 22%, achieving a moderately satisfying performance.

Two interesting issues regarding the corpus-based matcher are discussed here. First, reading and understanding WSDL files are somewhat like reading and understanding program codes, both containing a lot of irregular spelling, abbreviations, and acronyms. So it's very time-consuming and labor-intensive to manually pre-process the WSDL files. Second, when the name of an element is consisted of multiple words, those words are considered a unit when the system calculates the similarity score between this element and another. A better way to handle multi-word element names, which can better capture the inherent semantics, remains to be studied.

Conclusion and Future Work

In this paper, we present a corpus-based Web services matcher which uses the Web as an effective corpus to facilitate direct matchmaking between WSDL files. With an initial evaluation, we believe our contributions to the current literature are twofold: first, we have proposed a novel corpus-based method to match Web services, and evaluation shows it's promising, outperforming a baseline matcher; second, we have demonstrated the possibility of directly matching WSDL files without the need for additional levels of semantic markup.

Future works are as follows. First, while the current system works best for elements named in single words or meaningful phrases, we plan to develop an effective way to automate the pre-process of the WSDL files, which can also handle complex names (e.g. abbreviations, acronyms, and multi-words names) as well. Second, in the current system only semantic similarities are considered; structural information, such as parent-child relationship, number of children, etc., is ignored. Such information may be valuable in order to determine whether two services are similar or not. We plan to develop a structure matcher component which generates similarity scores representing the structural similarity between two WSDL files.

A full version of this paper is available online through the first author's website.

References

Bansal, S., and Vidal, J. 2003. Matchmaking of web services based on the DAML-S service model. In Proceedings of *the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 926 - 927. Melbourne, Australia, 2003.

Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J. 2004. Similarity Search for Web Services. In Proceedings of *the 30th VLDB Conference*, pp. 372 – 383. Toronto, Canada, 2004.

Kawamura, T., Blasio, J., Hasegawa, T., Paolucci, M., and Sycara, K. 2004. Public Deployment of Semantic Service Matchmaker with UDDI Business Registry. In Proceedings of *the 3rd International Semantic Web Conference (ISWC2004)*, LNCS 3298, pp. 752-766, 2004. Springer-Verlag, Berlin.

Paolucci, M., Kawamura, T., Payne, T., Sycara, K. 2002. Importing the Semantic Web in UDDI. *Lecture Notes In Computer Science*, Vol. 2512, pp. 225 - 236. Springer-Verlag London, UK, 2002.

Tsalgatidou, A. and Pilioura, T. 2002. An Overview of Standards and Related Technology in Web Services. *Distributed and Parallel Databases*, Vol. 12, pp. 135 -162.

Wang, Y., and Stroulia, E. 2003. Flexible Interface Matching for Web-Service Discovery. In Proceedings of *the Fourth International Conference on Web Information Systems Engineering*, pp. 147. 2003.

Zhang, L., Li, H., Chang, H., Chao, T. 2002. XML-based Advanced UDDI Search Mechanism for B2B Integration. *The 4th International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS)*, 2002.