Circular Context-Based Semantic Matching to Identify Web Service Composition

Aviv Segev
National Chengchi University
NO.64, Sec.2, ZhiNan Rd., Wenshan District
Taipei City 11605, Taiwan
asegev@nccu.edu.tw

ABSTRACT

This paper provides initial analysis in identifying possible Web services composition using context-based semantic matching. Context-based semantic matching allows service composition to be expanded beyond simple term matching and reduces erroneous compositions. A common method of context extraction is used to compare two types of service description, textual and WSDL. The approach is unique since it provides the Web service designer with an explicit numeric estimation of the extent to which a composition "makes sense." Motivation for the work is displayed with examples from Web services in the field of business.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services; H.3.1 [**Information Storage And Retrieval**]: Content Analysis and Indexing; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Semantic networks

General Terms

Algorithms, Design, Measurement

Keywords

Web services, Context, Semantics

1. INTRODUCTION

Businesses are beginning to view Web-services as a complementary field to their core businesses model. Businesses allow other organizations to integrate services which hook on to their main business service or alternatively produce a Web service as additional products which could be provided as a separate product to be easily integrated by clients.

In recent years, the use of services to compose new applications from existing modules has gained momentum. In part, this is due to the availability of simple standards such as WSDL for Web services. Also, the drive to utilize better existing resources (such as code reuse in organizations [6]) has motivated organizations to seek technologies to maintain code repositories and code composition. Service composition in a distributed heterogeneous environment immediately raises issues of integration. Services are autonomous units of code, independently developed and evolved, and therefore lack homogeneous structure beyond that of its interface (*e.g.*, as described in WSDL). Even there, while the interface may be a common one in the sense that all services have input, output, and some description of the service internals, heterogeneity of ways to define parameters and to describe internal processing (typically done as free text in WSDL) encumbers straightforward integration.

Consequently, context-based semantic matching for Web services composition has gained momentum. The research on the ability to match Web services has focused on semantic meaning rather than on the syntactic meaning. Works that relate to service integration have focused on various aspects of the problem. Semantic Web services were proposed to overcome interface heterogeneity. Using languages such as OWL-S [1], Web services are extended with an unambiguous description by relating properties such as input and output parameters to common concepts and by defining the execution characteristics of the service. The concepts are defined in *Web ontologies* [2], which serve as the key mechanism to globally define and reference concepts. Finally, an analysis of different techniques for mapping Web services to ontologies was performed based on text processing [5].

The ability to consider a composition of Web services would require analyzing the relation of the context of each service to other services. In this work it is proposed to analyze the context of each service in a circular two-way method. Each Web service context will be evaluated according to its proximity to other services and the proximity of each of the other services to the current service. Figure 1 displays the circular service description context analysis.

The analysis is based on the advantage that a Web service can be separated into two descriptions: the WSDL and a textual description of the Web service in free text. Each Web textual description and WSDL will be extracted using a known context analysis method [12]. The overlap of each service textual description context will be analyzed versus other Web services WSDL. This will form the circular analysis for each service which will allow a two-way context proximity comparison. The method proposed yields a numeric estimation of the extent to which a composition should be considered. The next section details the service analysis method. Section 3 displays some empirical experiences with matching Web services based on the method. Section 4 discusses aspects of the algorithm performance.

CSSSIA 2008, April 22, Beijing, China.

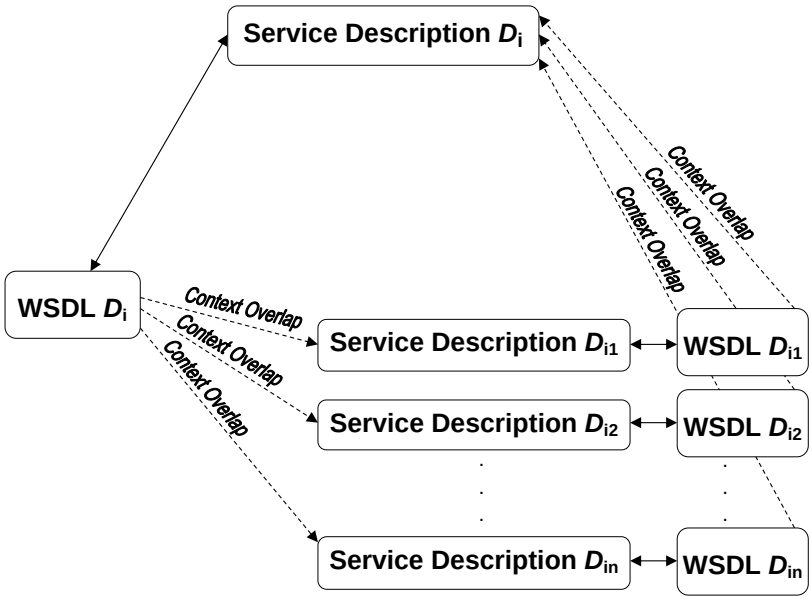

Figure 1: Circular service description context analysis

Section 5 describes the related work. Finally, Section 6 displays the conclusion and future work.

2. SERVICE ANALYSIS

Analyzing the Web services composition, we assume that each Web service is described using a textual description (which is part of the meta-data within UDDI registries), and a WSDL document describing the syntactic properties of the service interface. Figure 2 depicts an example of these two descriptions. These descriptions serve as the input to the analysis process.

2.1 Initial Analysis

The analysis starts with token extraction, representing each service, S, using two sets of tokens, called *descriptors*. Each token is a textual term, extracted by simple parsing of the underlying documentation of the service. The first descriptor represents the WSDL document, formally put as $D^{wsdl} = \{t_1, t_2, \ldots\}$. The second descriptor, $D^{desc} = \{t_1, t_2, \ldots\}$, represents the textual description of the service. WSDL tokens require special handling, since meaningful tokens (such as parameter names and operation names) are usually composed of a sequence of words, with the first letter of each word capitalized (*e.g.*, RetrieveIncidentsByPassportResult). Therefore, the tokens are divided into separate tokens. The tokens are filtered using a list of *stop-words*, removing words with no substantive semantics. For instance, the tokens *get*, *post*, and *result* are common in many WSDL documents.

An illustration of the baseline token list is depicted in Figure 3. These tokens were extracted from the WSDL document. All elements classified as name were extracted. The sequence of words was expanded as previously mentioned using the capital letter of each word.

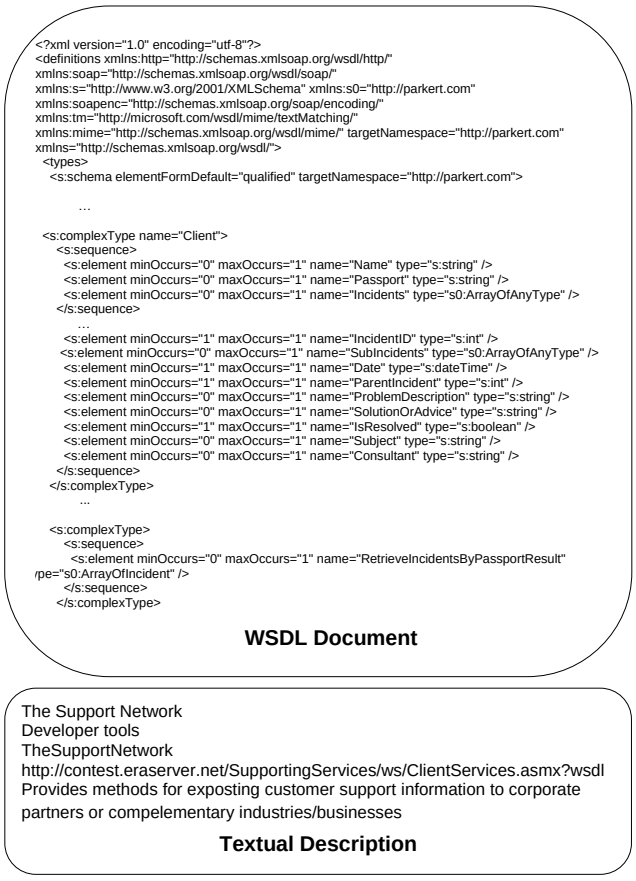

Figure 2: An Example: Exposing Customer Support Web Service

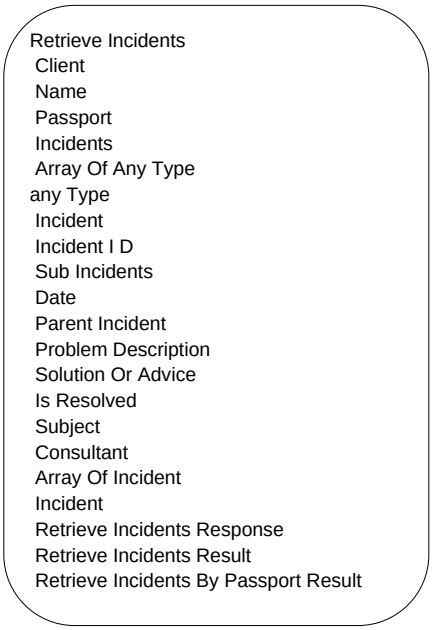

Figure 3: Initial Processing Example of the Exposing Customer Support Service

The next section presents some intuition regarding motivation for choosing the method of context extraction. Nevertheless, this choice is more or less arbitrary. Other methods for text extractions can be used, borrowing from the vast literature of Information Retrieval (IR) [11] and Machine Learning (ML) [9].

2.2 Context Extraction

The extraction process uses the World Wide Web as a knowledge base to extract multiple contexts for the tokens. Extraction is used to filter out biased tokens, to provide a more precise ranking, and to extend the service descriptors. The algorithm input is defined as a set of textual propositions representing the service description. The result of the algorithm is a set of *contexts* - terms which are related to the propositions. The context recognition algorithm was adapted from [12] and consists of the following three steps:

1. **Context retrieval:** Submitting each token to a Web-based search engine. The contexts are extracted from the Internet and clustered according to the results.

2. **Context ranking:** Ranking the results according to the number of references to the keyword, the number of Web sites that refer to the keyword, and the ranking of the Web sites.

3. **Context selection:** Finally, the set of contexts for the textual proposition, defined as the *outer context*, $\mathcal{C}$, is assembled.

The algorithm can formally be defined as follows: Let $\mathcal{D} = \{P_1, P_2, ..., P_m\}$ be a set of textual propositions representing a document, where for all P_i there exists a collection of descriptor sets forming the context $\mathcal{C}_i = \{\langle c_{i1}, w_{i1}\rangle, ..., \langle c_{in}, w_{in}\rangle\}$ so that $ist(\mathcal{C}_i, P_i)$ is satisfied. McCarthy [8] defines a relation $ist(\mathcal{C}, P)$, asserting that a proposition P is true in a context $\mathcal{C}$. In our case the adopted algorithm uses the corpus of WSDL descriptors, $\mathcal{D}^{wsdl}$, and textual description, $\mathcal{D}^{desc}$, as propositions P_i, and the contexts describing the WSDL as tokens c_i with their associated weight w_{i1}. The context recognition algorithm identifies the outer context $\mathcal{C}$ defined by:

$$ist(\mathcal{C}, \bigcap_{i=1}^{m} ist(\mathcal{C}_i, P_i)).$$

The input to the algorithm is a stream, in text format, of information. The context recognition algorithm output is a set of contexts that attempts to describe the current scenario most accurately. The algorithm attempts to reach results similar to those achieved by a human when determining the set of contexts that describe the current scenario (the Web service in our case). For example, Figure 4 provides the outcome of the Web context extraction.

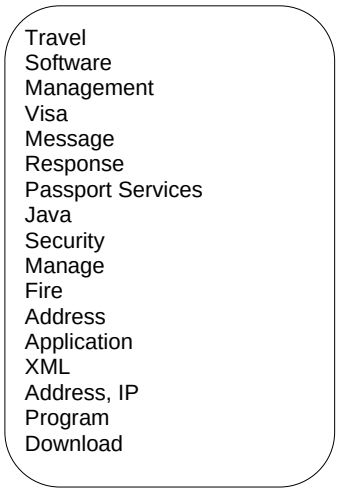

Figure 4: An Example of the Web Context

An advantage of the Web context extraction approach over simple token matching methods is the ability to add new possible contexts, textual descriptors, of the Web service which do not appear in the original text. This ability derives from the use of the Internet as a knowledge base for collecting the possible descriptors while other methods limit the descriptors to keywords appearing in the WSDL. For example, the Web context extraction described in Figure 4 includes the descriptors of "Management" and "Software" which did not appear in the original WSDL description displayed in Figure 3.

2.3 Context Overlap and Composition

To analyze a set of Web services and identify which services would be more likely to be composed, we analyze the overlap between the services based on their context. We compare each Web service WSDL descriptor context with another Web service textual descriptor context, as described in Figure 1.

Let $WS = \{D_1, D_2, ..., D_n\}$ define a set of Web services descriptions which are analyzed for composition. We denote Context Overlap (CO) as:

$$CO(D_i^{wsdl}, D_j^{desc}) = |\{c_k \epsilon D_i^{wsdl} \bigcap c_k \epsilon D_j^{desc}\}|$$

which defines the number of overlapping context descriptors, c_k, of Web service D_i WSDL descriptor with context descriptors of Web service D_j textual descriptor, and similarly for the reversed $CO(D_j^{wsdl}, D_i^{desc})$. $COMP_{ij}$ computes the composition likelihood, the proximity, between two given Web services:

$$COMP(D_i, D_j) = \sqrt{CO(D_i^{wsdl}, D_j^{desc})^2 + CO(D_j^{wsdl}, D_i^{desc})^2}$$

$COMP(WS)$ computes the two services from a given set with the highest likelihood of being composed to be:

$$COMP(WS) = Max_j\,(Max_i(COMP(D_i, D_j)))$$

The technique can be used iteratively to identify the top n services which could be considered for composition. Alternatively, a threshold could be used for the $COMP$ value to define the number of compositions which should be considered.

3. EXPERIENCES WITH MATCHING WEB SERVICES

The experiences are based on the data taken from an existing benchmark repository, of several hundred Web services, provided by researchers from University College Dublin.[1] The preliminary experiments use a set of 23 representative Web services, originally defined under the topic of business. For each Web service the repository provided a WSDL document and a short textual description.

The experiments used two different methods for context extraction, as described in Section 2:

Description Context The Context Extraction algorithm described in Section 2.2 was applied to the textual description of the Web services. Each descriptor of the Web service context was used as a token.

Name Context The Context Extraction algorithm was applied to the *name* labels of each Web service. Each descriptor of the Web service context was used as a token.

The Context Overlap of each set of two Web services was calculated according to the algorithm described in Section 2.3. The algorithm was applied twice iteratively on the same data and the two top ranking pairs of Web services considered for composition were identified.

Figure 5 displays the results of the context overlap for all of the Web services. The horizonal axis displays all of the Web services composition pairs. The vertical axis displays the context overlap value for each composition. Two composition results are prominent with high values of 6 and 7.

The highest value for suggested composition was received for composing a Web service that provides methods of exposing customer support information to corporate partners or complementary industries/businesses (displayed in Figure 2) with a Web service that provides a bar code for any product number or ID. The composition suggested by the algorithm enables the customer support information service

[1]http://moguntia.ucd.ie/repository/ws2003.html

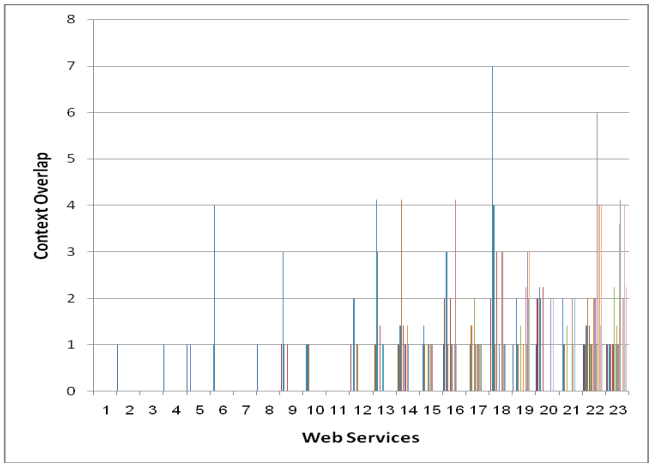

Figure 5: Web Services Context Overlap

to be improved by using the bar code instead of multiple information classification ID's such as "Incident ID" and "Retrieve Incidents By Passport Result" which appear in the original WSDL description in Figure 2.

4. DISCUSSION

The initial results indicate that the context overlap can be used to identify possibilities of Web services composition. The method not only analyzes whether the two services should be considered for composition but also supplies a numeric estimation of the extent to which a composition can be made in comparison to the other services.

The effort required to analyze all possible combinations for a large quantity of Web services is time consuming. The numeric estimation of the extent of the composition allows a Web service engineer to prioritize the analysis of each possible composition. The algorithm ability to evaluate the composition according to the context based semantic matching approach that uses the Web as a knowledge source extends the role of Web services in new directions. The algorithm thus integrates two points of view, the "internal" view, the given WSDL description, and the "external" view, the text description of the different services from which we are looking for possible compositions.

The complexity of the context Web-based method is $o(an)$ where n represents the number of input cycles such as each line of text. The a represents a constant limiting the number of top ranking results from each cycle of the algorithm. The execution time of the context based method is slow due to the need to access the Web search engine for every line of input extracted from the WSDL and can reach between to 3-4 minutes for very long WSDL documents. However, since each web service only needs to be classified once in its life time, performance is less important than accuracy. The complexity of the context overlap and composition method is $o(n^2)$ where n represents each of the Web services examined for composition. However, since this step is based on matching concepts the execution time is negligible compared to the context extraction step.

5. RELATED WORK

Patil et al. [10] present a combined approach towards automatic semantic annotation of services. The approach relies on several matchers (string matcher, structural matcher, and synonym finder), which are combined using a simple aggregation function. A similar method, which also aggregates results from several matchers, is presented by Duo et al. [4]. The current research aims at a coarser-grain task and therefore different methods for the preliminary evaluation were chosen. However, the methods suggested in these works will be evaluated in future research.

ASSAM [7] is a tool for semi-automatic annotation of Web services. It uses learning techniques in order to narrow down possible concepts, helping a human user to manually tag the service. The approach currently presented supplies a numeric estimation in order to assess the possible composition.

Woogle [3], by Dong et al., is a search engine for Web services. It accepts keyword queries and returns results according to information in WSDL documents, such as message parameters. While some of the matching algorithms used by Woogle are relevant to the current work, Woogle matches keywords while the current work explores the matching of formal concepts. However, further empirical evidence for some of the conclusions of Dong et al., namely the effectiveness of clustering keywords according to their mutual distance in the WSDL file is provided.

6. CONCLUSIONS

The research displays initial results in context-based semantic matching for possible Web services composition. The method compares context extracted from each Web service based on its WSDL description to all other Web services textual description context. The initial experiences based on the analysis of context overlap between different Web services show promising results of identifying possible composition suggestions.

The research described so far is a work-in-progress. The intention is to extend the experiments to a larger corpus of services. Additional possible directions of research include an analysis of other options of context overlap for possible compositions, such as context overlap based only on textual description overlap, and the matching of textual description to WSDL, the reversed operation to the one in the current research.

7. ACKNOWLEDGMENTS

The research work was partially supported by the Sayling Wen Cultural & Educational Foundation through the Service Science Research Center (SSRC) at National Chengchi University in Taiwan.

8. REFERENCES

[1] A. Ankolekar, D. Martin, Z. Zeng, J. Hobbs, K. Sycara, B. Burstein, M. Paolucci, O. Lassila, S. Mcilraith, S. Narayanan, and P. Payne. DAML-S: Semantic markup for web services. In *Proceedings of the International Semantic Web Workshop (SWWS)*, July 2001.

[2] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL web ontology language reference. W3C candidate recommendation, W3C, 2004.

[3] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Simlarity search for web services. pages 372–383, 2004.

[4] Z. Duo, L. Juan-Zi, and X. Bin. Web service annotation using ontology mapping. In *SOSE '05: Proceedings of the IEEE International Workshop*, pages 243–250, Washington, DC, USA, 2005. IEEE Computer Society.

[5] A. Gal, A. Segev, and E. Toch. Semantic methods for service categorization - an empirical study. In *Proceedings International Workshop on Semantic Data and Service Integration (SDSI 2007)*, 2007.

[6] M. Gu, A. Aamodt, and X. Tong. Component retrieval using conversational case-based reasoning. In *Proceedings of the International Conference on Intelligent Information Systems (ICIIP 2004)*, pages 21–23, Beijing, China, Oct. 2004.

[7] A. Heß, E. Johnston, and N. Kushmerick. ASSAM: A tool for semi-automatically annotating semantic web services. In *International Semantic Web Conference*, pages 320–334, 2004.

[8] J. McCarthy. Notes on formalizing context. *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.

[9] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[10] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. Meteor-s web service annotation framework. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 553–562, New York, NY, USA, 2004. ACM Press.

[11] C. J. V. Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[12] A. Segev, M. Leshno, and M. Zviran. Context recognition using internet as a knowledge base. *Journal of Intelligent Information Systems*, 29(3):305–327, 2007.