Improving Performance of Web Services Query Matchmaking with Automated Knowledge Acquisition

Chaitali Gupta, Rajdeep Bhowmik, Michael R. Head, Madhusudhan Govindaraju, Weiyi Meng
Department of Computer Science, State University of New York (SUNY) at Binghamton, NY
{cgupta1, rbhowmi1}@binghamton.edu {mike, mgovinda, meng}@cs.binghamton.edu

Abstract

There is a critical need to design and develop tools that abstract away the fundamental complexity of XML-based Web services specifications and toolkits, and provide an elegant, intuitive, simple, and powerful query-based invocation system to end users. Web services based tools and standards have been designed to facilitate seamless integration and development for application developers. As a result, current implementations require the end user to have intimate knowledge of Web services and related toolkits, and users often play an informed role in the overall Web services execution process. We employ a self-learning mechanism and a set of algorithms and optimizations to match user queries with corresponding operations in Web services. Our system uses Semantic Web concepts and Ontologies in the process of automating Web services matchmaking. We present performance analysis of our system and quantify the exact gains in precision and recall due to the knowledge acquisition algorithms.[1]

Key Words: Web services, Matchmaking, Semantic Web, Ontology, Information Extraction

1. Introduction

The Web services model has emerged as a standard for representation, discovery, and invocation of services in a distributed environment. A Web service can be defined as an interface to application functionality that is accessible using well-known Internet standards and is independent of any operating system or programming language. The widespread adoption of Web services is enabled by a set of flexible and extensible XML-based standards including the Web Service Description Language (WSDL) [1], which is the widely used specification to describe Web services. Web services are widely expected to simplify the design of distributed applications that are amenable to automated discovery, composition, and invocation. The use of XML [2] facilitates in moving towards loosely-coupled applications that provide greater interoperability in distributed heterogeneous environments. However, the current XML-based specifications provide only syntactical descriptions of the functionality provided by Web services. Even though a wide variety of tools are available, the lack of semantics associated with Web service descriptions requires user intervention in the decision making process. Though an important motivation of Web services is to promote ease-of-use for application developers, the requirement that end users also be familiar with the design and some implementation details makes its usage difficult for end users. Our work addresses this problem by simplifying the user interaction with Web services. We have developed several algorithms and optimization techniques that match user queries to relevant operations in domain specific Web services. Our system presents a simple interface to accept user queries, similar to HTML based search engines, and maps the queries to appropriate Web services operations. We employ several query matching techniques including Semantic Web [3] and ontology technologies such as OWL [4], as well as tools such as WordNet [5], to retrieve contextual information from queries and determine the set of Web services operations relevant to the user query. The details of Web services specification and implementation are hidden from the user. For example, suppose a user wants to check the weather for a trip from Boston to Chicago. In our system, the user needs to enter the query "*weather for travel from Boston to Chicago.*"

Our system employs various matchmaking algorithms to understand the query and obtain a set of relevant operations of the Web service. Unlike other Web service implementations, the user does not have

[1] Supported in part by NSF grants IIS-0414981 and CNS-0454298

0-7695-3026-5/07 $25.00 © 2007 IEEE
DOI 10.1109/WI.2007.101

to fill detailed forms for each service. Our system takes into consideration previous Web service matchmaking results and utilizes them to improve performance for subsequent user queries in the same domain. Our system supports memoized optimization, which uses the knowledge of certain or entire parts of previously made queries, for the benefit of future queries**.** In this paper we present performance analysis for each module and quantify the exact gains in precision and recall due to the self-learning capability of the system. We also measure how precision changes with the increase in the number of elements and relations in the OWL ontologies.

2. Implementation Details

Figure 1 shows the components and control flow of our system. A brief overview of the modules in the system is provided in our previous work [6].

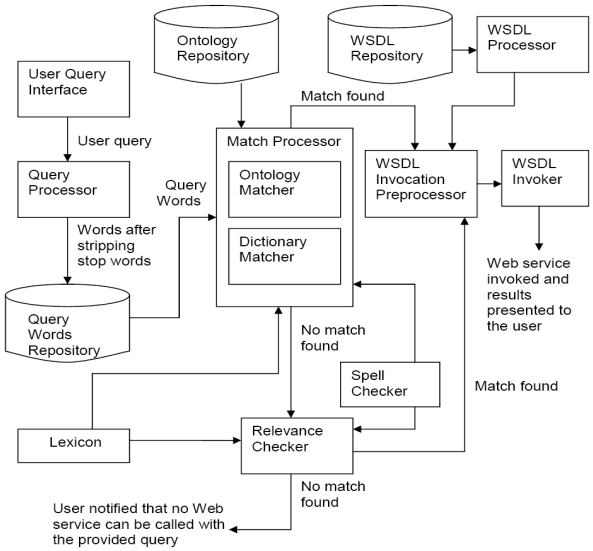

Figure 1. Overall architecture of the system

3. Automated Knowledge Acquisition

Within the Ontology Matcher module in our system, the Lexicon block is used and its features are employed to obtain better contextual information relevant to the client query. The Lexicon Block is built using the WordNet 2.0 Dictionary [5]. Our system uses JWNL 1.3 API [11] to access the WordNet Dictionary. Initially, the query words are matched using direct keyword matching with the Subject, Object and Predicate of each ontology statement. Irrespective of the matches found, we use the Lexicon block to employ synonym matching techniques. By taking into consideration different senses of a particular word, we can ensure that the selected ontology domain has the closest relevance to the client query string.

For synonym matching, four different search outcomes are possible.

- Neither the query word nor the synonym words are present in any of the ontology models.
- Some of the synonyms are present, but not the query word.
- The query word is present, but not its synonyms.
- Both the query word and its synonyms are present in the ontology model.

We collect these outcomes, shown in Figure 2, and use them to extend the ontology models, thus enriching the model. We have designed a learning module that stores the knowledge and information of a previously made query (the semantics of which are not in our ontology) to later queries for predicting more accurate results. If both the query word and its synonyms are not found, the ontology model does not get extended. The same condition applies when the query word and its synonyms are both found within the ontology model. However, the ontology file is extended when a synonym of a particular word yields a match. If a synonym of the query word is present in the ontology file, we infer that the query word very likely has contextual relevance to the ontology model.

Suppose we have a query "*temperature at Binghamton*" and we do not have the keyword "temperature" in our present ontology model. Further assume that from the Lexicon we can infer that "weather" is a synonym of temperature and "weather" is already present in the ontology model. It can then be inferred that "temperature" has a meaning that is semantically similar to "weather" and should be included in the ontology model. So we regenerate the weather ontology model and incorporate the keyword "temperature" in the ontology file. Each time any of the ontology models is updated, we create and read the new ontology model again so that the changes are incorporated. However, if a keyword from the user query string is present in the ontology model, every synonym of it does not qualify to be incorporated into the ontology model. For example, for the query string "*weather at Binghamton*", instead of "*temperature at Binghamton*", we get "endure" as a synonym for weather from Lexicon. Since "endure" is not present in the ontology file, we cannot extend the model because the word "endure" carries a different sense that is not relevant to the present context of "weather".

The self-learning mechanism, provided by our system, utilizes the knowledge of previously made

queries and enhances the efficiency of the system by a range of 20%-82% [6].

Query Word	Synonym	Result
0	0	Continue
0	1	Extend OWL file with query word
1	0	Continue, Synonyms may have different senses
1	1	Continue

Figure 2. Ontology Extension Table

4. Experimental Results

We conducted experiments on a Dell D620 with an Intel T2300 processor @ 1.66 GHz and 1 GB of RAM running Microsoft Windows XP.

Figure 3 denotes the amount of time taken by each module within our system. The results are averaged across 50 queries of varying sizes, randomly selected from 4 domains: Travel, Currency, Weather and Location. We generated the profiling data using TPTP 4.2.1 Eclipse plug-in and Eclipse SDK 3.2.2.

From Figure 3 it can be seen that the Query Processor, Relevance Checker, and Ontology Matcher modules consume the least amount of processing time. It is to be noted that the execution time of the Ontology Matcher does not take into account the time needed by the Jena toolkit [12] to load a new extended RDF model of the self-learning mechanism of the system. WSDL processing is a one-time operation; so the execution time of the WSDL processor does not have a major impact on the performance of the system. The major bottleneck in the overall process is the Lexicon block. It is used to find the synonyms, hyponyms, and hypernyms of client query words and the synsets[2] for extending the ontology models. Our analysis shows that a major portion of the time is spent in loading the JWNL API implementation for generating the synsets for each client query word.

We use the precision and recall measurements to study the accuracy of our system. We take into consideration a sample of 50 queries and evaluate the outputs. For measuring the overall accuracy of our system, we define *precision* as ratio of the number of relevant WSDL operations retrieved corresponding to a user query and the total number of WSDL operations returned by our system. We define *recall* as the ratio of the number of relevant WSDL operations retrieved and the total number of relevant WSDL operations for a user query present in the WSDL repository.

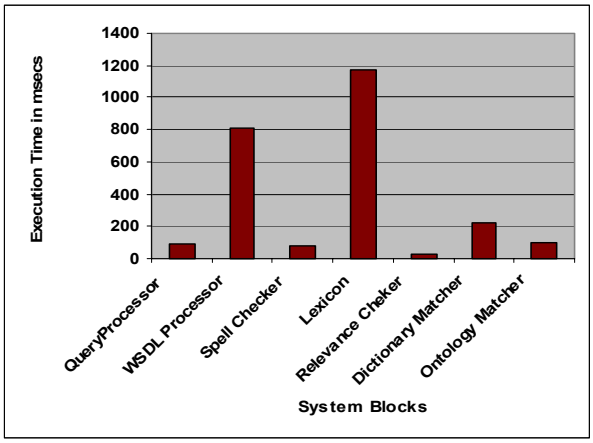

Figure 3. Execution time taken by each component of the system

Let D be the total number of WSDL operations present in the WSDL repository, Q_i be the query to be tested and R_{Q_i} be the set of WSDL operations in the WSDL repository that are relevant to the query Q_i. Since $R_{Q_i} \subset D$, therefore $D - R_{Q_i}$ is the set of WSDL operations that are irrelevant to the query under consideration. In other words, Q_i partitions D into two subsets, R_{Q_i} and $D - R_{Q_i}$. We define the contents of these sets and the query before running the precision/recall test. In our experiment, we refer T_{Q_i} to the WSDL operations returned by our system and G_{Q_i} to all relevant WSDL operations retrieved for query Q_i. Therefore, the sets hold to the relations $G_{Q_i} = T_{Q_i} \cap R_{Q_i}$ and $T_{Q_i} - G_{Q_i} = T_{Q_i} \cap (D - R_{Q_i})$. $T_{Q_i} - G_{Q_i}$ refers to the irrelevant WSDL operations retrieved for query Q_i. Thus, in our experiment, the precision is calculated as $P = |G_{Q_i}| / |T_{Q_i}|$ and recall as $R = |G_{Q_i}| / |R_{Q_i}|$.

Table 1, Table 2, and Table 3 refer to the precision and recall values across the different domains. In Table 1, precision-recall results of the domain-independent methods (Dictionary Matcher) are not impressive as semantics are not associated with keywords. The results in Table 1 quantify the upper limit of

[2] A synset (synonym set) represents a concept and contains a set of words; each of which is synonymous with the other words in the synset.

performance that we can get if just the algorithms for Dictionary Matcher and Lexicon are employed. In Table 2, we can see that both precision-recall values are significantly higher. The domain-dependent ontologies (Ontology Matcher) are more effective because of its use of semantic and domain-specific knowledge base. In Table 3, precision-recall results are slightly better than in Table 2 when we combine both domain-independent and domain-dependent methodologies. As the coverage of our ontological knowledge base is currently not exhaustive, the keywords that are not found in the OWL files by the Ontology Matcher are serviced by the Dictionary Matcher. This effectively improves the overall accuracy of the system.

Table 1. Performance of Domain Independent Methods.

Domain	Precision	Recall
Travel	73.3%	71.8%
Currency	67.9%	62.7%
Weather	83.4%	79.5%
Location	77.6%	74.7%
Average	75.6%	72.2%

Table 2. Performance of Domain Dependent Ontologies.

Domain	Precision	Recall
Travel	96.7%	93.8%
Currency	99.3%	91.6%
Weather	99.5%	98.2%
Location	98.8%	92.1%
Average	98.6%	93.9%

Table 3. Performance of Combined Methodologies.

Domain	Precision	Recall
Travel	97.2%	95.2%
Currency	99.5%	92.1%
Weather	100%	98.7%
Location	99.2%	94.9%
Average	99%	95.2%

We compare the performance between the domain-independent, domain-dependent, and combined methodologies in Figure 4. When the domain-dependent ontologies are not used, both precision and recall decrease, which indicates that the Ontology Matcher is vital to the accuracy of the system. If we remove the Ontology Matcher, both precision and recall drop by approximately 23%. The domain-independent method is a less significant measure pertaining to the accuracy of the system. Without the Dictionary Matcher, both precision and recall drop by approximately 1%.

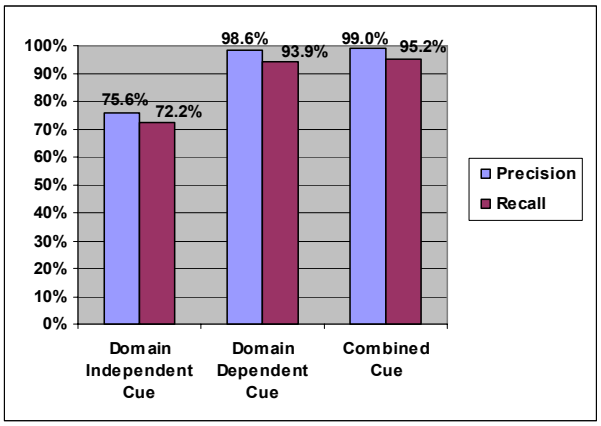

Figure 4. Evaluation of the different methodologies.

We conducted an experiment to determine how accurately our system behaves for a user query with an increase in the size of the ontological knowledge base. Based on the results of this experiment, we can draw the following conclusions:

a) With the addition of elements and relations in the OWL ontologies related to the user query, the Ontology Matcher always retrieves the relevant statements. So both precision and recall increase.

b) With the addition of elements and relations in the OWL ontologies unrelated to the user query, the Ontology Matcher never retrieves the irrelevant statements. So both precision and recall remain unchanged.

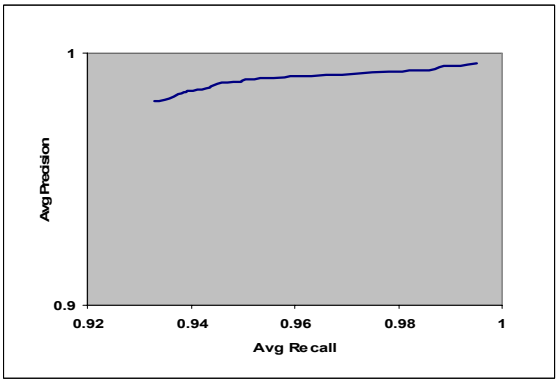

Figure 5. Precision-Recall averaged across four domains with the increase in the number of elements and relations in OWL ontologies.

Figure 5 pertains to category 'a' mentioned above, the only difference being that the OWL ontologies are automatically extended instead of manual addition of elements and relations. Figure 5 shows that with the increase in recall, precision also increases. This performance improvement is due to the self-learning mechanism incorporated in our system, which adds relevant semantic concepts to the ontology models. As a result, the number of relevant matches also increases with the number of statements retrieved from the ontological knowledge base.

5. Related Work

Patil et al. have developed MWSAF, a Web service annotation framework [7] that performs both element and structural level matching for Web services. The element level matching is bound on a combination of Porter Stemmer algorithm for root word selection, WordNet dictionary for synonyms, abbreviation directory to handle acronyms, and NGram algorithm for linguistic similarity of the names of two concepts. Sycara et al. have developed one of the earliest ontology-based semantic matchmaking engines, MatchMaker [8], which uses capability-based semantic match and various IR-based filters. Another related effort is Racer [9], which focuses solely on service capability-based semantic matches for application in e-commerce systems. Syeda-Mahmood et al. [10] explore the use of domain-independent and domain-specific ontologies for finding matching service descriptions. Domain-independent relationships are derived using an English thesaurus after tokenization and part-of-speech tagging, while domain-specific ontological similarities are derived by inferring semantic annotations associated with Web service descriptions. A combination of the matches due to the two cues is done to determine an overall semantic similarity score. Our work extends the work by Syeda-Mahmood et al. [10], but dynamically learning from previous matchmaking results, extending the ontological vocabulary, and applying the knowledge to subsequent queries.

6. Conclusions

This paper presents a system that matches user queries with Web services operations. The system uses lexical analysis, domain-independent matching techniques, domain-specific ontologies and a set of specialized algorithms and optimizations to match simple free-form queries to WSDL operations. Our system provides the ease-of-use of popular Web search engines, enhanced with the ability to combine and retrieve information related to user queries. We also provide a detailed accuracy and profiling study of our system. Experimental results demonstrate the viability of our approach in terms of simplicity, effectiveness, and performance, facilitating in query-based search and matchmaking of Web services.

7. References

[1] Web Services Description Language (WSDL), Version 2.0 Part I, W3C Working Draft, May 2005, *http://www.w3.org/TR/wsdl20/*.

[2] Tim Bray and Jean Paoli and C. M. Sperberg-McQueen and Eve Maler, "XML 1.0: Extensible Markup Language (XML) 1.0", Second Edition, *http://www.w3.org/TR/REC-xml*, October 2000.

[3] "Semantic Web" Web Page. Available: *http://www.w3.org/2001/sw*.

[4] "OWL Web Ontology Language Overview" Web Page. Available: *http://www.w3.org/TR/owl-features/*.

[5] G. A. Miller, "WordNet: A Lexical Database for the English Language" in *Comm. ACM 1983*.

[6] Chaitali Gupta, Rajdeep Bhowmik, Michael Head, Madhusudhan Govindaraju, Weiyi Meng, "A Query-based System for Automatic Invocation of Web Services", in the *Application Services and Industry Track, IEEE International Conference on Web Services (ICWS)*, Salt Lake City, Utah, July 2007.

[7] A. Patil et al., "METEOR-S Web Service Annotation Framework" in *Proc. WWW Conference, pp. 553-562, 2004*.

[8] K. Sycara et al., "Dynamic service match making among agents in open information environments" in *Journal of the ACM SIGMOD Record, 1999*.

[9] L. Li, I. Harrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology" in *Proc. WWW Conference, 2003*.

[10] Syeda-Mahmood et al., "Searching Service Repositories by Combining Semantic and Ontological Matching" in *Proc. of the IEEE International Conference on Web Services, 2005*.

[11] "JWNL 1.3" Web Page. Available: *http://jwordnet.sourceforge.net/*.

[12] "Jena - A Semantic Web Framework for Java" Web Page. Available: *http://jena.sourceforge.net.*