Semantic Structure Matching for Assessing Web-Service Similarity

Yiqiao Wang and Eleni Stroulia

Computer Science Department, University of Alberta, Edmonton, AB, T6G 2E8, Canada
{yiqiao,stroulia}@cs.ualberta.ca

Abstract. The web-services stack of standards is designed to support the reuse and interoperation of software components on the web. A critical step in the process of developing applications based on web services is service discovery, i.e., the identification of existing web services that can potentially be used in the context of a new web application. UDDI, the standard API for publishing web-services specifications, provides a simple browsing-by-business-category mechanism for developers to review and select published services. To support programmatic service discovery, we have developed a suite of methods that utilizes both the semantics of the identifiers of WSDL descriptions and the structure of their operations, messages and data types to assess the similarity of two WSDL files. Given only a textual description of the desired service, a semantic information-retrieval method can be used to identify and order the most similar service-description files. This step assesses the similarity of the provided description of the desired service with the available services. If a (potentially partial) specification of the desired service behavior is also available, this set of likely candidates can be further refined by a semantic structure-matching step assessing the structural similarity of the desired vs. the retrieved services and the semantic similarity of their identifier. In this paper, we describe and experimentally evaluate our suite of service-similarity assessment methods.

1 Introduction

The development of web-based applications in the service-oriented architecture style, as implied by the web-services stack of standards, relies on a set of related specifications, defining how reusable components should be specified (through the Web-Service Description Language – WSDL [15]), how they should be advertised so that they can be discovered and reused (through the Universal Description, Discovery, and Integration API – UDDI [12]), and how they should be invoked at run time (through the Simple Object Access Protocol API – SOAP [11]). A critical step in the process of reusing existing WSDL-specified components for building web-based applications is the discovery of potentially relevant components. UDDI servers are essentially catalogs of published WSDL specifications of reusable components. These catalogs are organized according to categories of business activities. Service providers advertise services by adding their WSDL specifications to the appropriate UDDI

directory category. Through a well-defined API, software developers can browse the UDDI catalog by category.

This category-based service-discovery method is clearly insufficient. It is quite informal and relies, to a great extent, on the shared common-sense understanding of the domain by publishers and consumers. It is the responsibility of the provider developer to publish the services in the appropriate UDDI category. The consumer developer must, in turn, browse the "right" category to discover the potentially relevant services. More importantly, this discovery process does not provide any support for selecting among competing alternative services that could potentially be reused; prioritization of the candidates is again the responsibility of the consumer.

In this paper, we discuss a set of WSDL similarity-assessment methods, which can be used, in conjunction with the current UDDI API, to support a more automated service-discovery process, by distinguishing among the potentially useful and the likely irrelevant services and by ordering the candidates in the first category according to their relevance to the task at hand. This method utilizes both the semantics of the identifiers of WSDL descriptions and the structures of their operations, messages and types to assess the similarity of two WSDL files. Given only a textual description of the desired service, a semantic information-retrieval method can be used to identify and order the most similar service-description files. This step assesses the similarity of the provided desired-service description, extended to include semantically similar words according to wordNet [16], with the available services. If a (potentially partial) specification of the desired service behavior is also available, this set of likely candidates can be further refined by a semantic structure-matching step assessing the structure and semantic similarity of the desired vs. the retrieved services.

The intuition underlying this method is that a plausible means of querying UDDI servers is "query by example", i.e., by providing a (potentially partial) specification of the desired service. The consumer developer may define various aspects of the desired service, such as descriptions in natural language, the namespaces of its data types and the input/output parameters of its operations, and the proposed method will return a set of candidate services with an estimate of their similarity to the provided example.

The remainder of the paper is organized as follows: section 2 discusses related work; section 3 explains in detail the design and implementation of our approach; section 4 discusses the results of our experimentation; section 5 outlines our plans for future work and concludes with a summary of our results to date.

2 Related Research

The problem of service discovery is similar to the well-studied problems of component retrieval and information retrieval. On one hand, a WSDL specification is the description of a "software component" including a description of its interface and a description of where the actual implementation exists and how it can be used. On the other, a WSDL specification usually includes a set of natural-language description of the service itself and comments on its elements. Thus, we looked at both these

research areas for applicable results to the service-discovery and similarity-assessment problem.

2.1 Component retrieval

In general, there are two categories of methods for component discovery: signature matching [8, 18] and specification matching [1, 19].

Polylith [8] proposed one of the earliest signature-matching methods for interface adaptation and interoperation. Through its NIMBLE language, coercion rules could be specified so that the parameters of the invoking module could be matched to the signature of the invoked module, including reordering, type mapping and parameter elimination. Zaremski and Wing [18] described exact and relaxed signature matching as a means for retrieving functions and modules from a software library.

Signature matching is an efficient means for component retrieval, for several reasons. Function signatures can be automatically generated from the function code. Furthermore, signature matching efficiently prunes down the functions and/or modules that do not match the query, so that more expensive and precise techniques can be used on the smaller set of remaining candidate components. However, signature matching considers only function types and ignores their behaviors; and two functions with the same signature can have completely opposite behaviors. Specification matching aims at addressing this problem by comparing software components based on formal descriptions of the semantics of their behaviors. However, because these specifications are developed independently from the module code, there is no guarantee that they correctly and completely reflect the component's behavior. Moreover, it is hard to motivate programmers to provide a formal specification for each component they write.

Zaremski and Wing [19] extended their signature-matching work with a specification-matching scheme.

WSDL [15], the Web-Services Definition Language, is an XML-based interface-definition language. It describes "services" as a set of operations implemented by a set of messages involving a given set of data types at a high level of abstraction. WSDL specifications of service-providing components are published in UDDI registries. UDDI [12] is designed as an online marketplace providing a standardized format for general business discovery. Developers can browse and query a UDDI registry using the UDDI API to identify businesses that offer services in a particular business category and/or services that are provided by a certain service provider.

WSDL service specifications do not include semantics. On the other hand, DAML-S [2] is a formal logic-based language that supports the specification of semantic information in RDF format. As part of the "semantic web" effort, it is intended as the means for specifying domain-specific semantics of ontologies. An extension of DAML-S supports service specification, including behavioral specifications of their operations; as a result, it enables discovery through specification matching, such as the method proposed in LARKS [4]. If indeed services were specified in DAML-S instead of WSDL it would be possible to formally prove that the requirements of the desired service and a discovered service do not conflict. However, there is no widespread adoption yet of DAML and DAML-S, and the high cost of formally

defining of provided and required services makes this adoption unlikely. This is the underlying motivation for this research: to provide some lightweight semantic comparison of syntactic specifications in WSDL, based on the syntactic structure of the specifications and the natural-language semantics of their identifiers, comments and descriptions.

2.2 Information retrieval

Traditional information-retrieval methods rely on textual descriptions of artifacts to assess their similarity and organize them in clusters or retrieve them in a "query-by-example" mode [3]. According to the vector-space model, documents and queries are represented as t-dimensional vectors, where t is the number of distinct words in the document; similarity assessment then becomes equivalent to vector-distance calculation.

WordNet [16] is a lexical database, inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Relationships between conceptions such as hyponym and hypernym relations are represented as semantic pointers linking between related concepts [5, 6]. WordNet has been used in numerous natural language processing applications, hoping to ameliorate traditional information-retrieval results [7, 9, 13] with limited success.

3 The Web-service Discovery Method

Our service-discovery method is aimed at enabling programmatic service discovery and integrates information- and component-retrieval ideas. The method assumes as input a (potentially partial) specification of the desired WSDL specification and a set of WSDL specifications of available services, such as the services advertised in UDDI. First, a traditional vector-space model information-retrieval step, enhanced with WordNet, retrieves the most similar services according to their WSDL service descriptions specified in natural language. Given the retrieved list of candidate services, a structure-matching algorithm, extended by a second WordNet method that calculates semantic distances between identifiers of WSDLs, further refines and assesses the quality of the candidate service set.

3.1 WordNet-Powered Vector-Space Model Information Retrieval

In traditional vector-space model, documents and queries are represented as T-dimensional vectors, where T is the total number of distinct words in the document collection after the preprocessing step. Preprocessing includes eliminating stop words (very commonly used words) and conflating related words to a common word stem. Each term in the vector is assigned a weight that reflects the importance of a word in the document. This value is proportional to the frequency a word appears in a document and inversely proportional to number of documents in which this word

appears [10, 13]. A common term importance indicator is *tf-idf* weighting the importance of a word *i* in document *j* is as follows:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/df_i) \quad (1)$$

In the above formula, tf_{ij} is the normalized term frequency across the entire document collection, and idf_i is the inverse document frequency of term *i*. *N* is total number of documents in the collection, and *Log* is used to dampen the effect relative to *tf*.

The WordNet-powered vector-space model extension involves the maintenance of three sub-vectors for each document and query: stems of original words in a document, stems of words' synonyms for all word senses, and stems of words' direct hypernyms, hyponyms and siblings for all word senses. All document terms' word senses are included, and therefore we bypass the problem of lacking effective automated word sense disambiguation techniques, frequently discussed in the literature.

The WSDL syntax allows textual descriptions for services, their types and operations, grouped under <documentation> tags. Given a natural language description of the desired service, we employ the WordNet-powered vector space model to retrieve published WSDL services that are most similar to the input description on the respective vectors. Corresponding sub-vectors from documents and queries are matched and we obtain three similarity scores accordingly. Different weights are assigned to sub-vector matching scores: matching scores of original word stems (first sub-vectors) are assigned twice the weight assigned to matching scores of synonyms (second sub-vectors), hypernyms, hyponyms, and siblings (third sub-vectors). A higher overall score indicates a closer similarity between the source and target specifications.

3.2 WSDL Structure Matching

A straight-forward extension of the signature-matching method to WSDL specifications involves the comparison of the operations' set offered by the services, which is based on the comparison of the structures of the operations' input and output messages, which, in turn, is based on the comparison of the data types of the objects communicated by these messages.

The overall process starts by comparing the data types involved in the two WSDL specifications. The result of this step is a matrix assessing the matching scores, i.e., the degree of similarity, of all pair-wise combinations of source and target data types. It is interesting to note here that the data types of web services specified in WSDL are XML elements; as such, they can potentially be highly complex structures.

The next step in the process is the matching of the service messages. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target messages. The degree to which two messages are similar is decided on the basis of how similar their parameter lists are, in terms of the data types they contain and their organization.

The third step of the process is the matching of the service operations. The result of this step is a matrix assessing the matching score of all pair-wise combinations of source and target operations. The degree to which two operations are similar is decided on the basis of how similar their input and output messages are, which has already been assessed in the previous level.

Finally, the overall score of how well the two services match is computed by identifying the pair-wise correspondence of their operations that maximizes the sum total of the matching scores of the individual pairs.

After all target WSDL specifications have been matched against the source WSDL specification, they are ordered according to their "overall matching scores": a higher score indicates a closer similarity between the source and target specifications. For each target specification, the algorithm also returns the mapping of its data types and operations to the corresponding data types and operations of the source specification as an "explanation" of its assigned match score. This algorithm is described in detail in [14].

3.3 Semantic WSDL Structure Matching

The WSDL structure-matching algorithm of section 3.2 aims at optimizing the mapping of the corresponding service structures. The semantic WSDL structure-matching algorithm is an extension to it: it also tries to find an optimal mapping between source and target service components based both on the similarity of their syntactic structures and also the semantic similarity between the identifiers of data types, operations and services to assess service similarities. The intuition behind it is that the chosen names of the types, operations, and services usually reflect the semantics of the underlying capabilities of the service.

The identifier-matching process is similar to that of the original WSDL structure matching. It starts by comparing the names of the data types (identifiers) involved in the two WSDL specifications. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target data-types. The next step in the process is the matching of the service operations. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target operations. The degree to which two operations are similar is decided on the semantic distance between operations' names and how similar their parameter lists are, in terms of the identifiers they contain. Finally, the overall score for how well the two services match is computed by matching the services' names and by identifying the pair-wise correspondence of their operations that maximizes the sum total of the matching scores of the individual pairs.

Figure 1 lists the algorithm `matchDocumentTerms` that explains the WordNet-based "cost structure" for assessing the similarity of two identifiers. If two words are identical or synonymous (regardless of words' senses), they are assigned a maximum score of 10 and 8 respectively. Otherwise, if two words are in a hierarchical semantic relation, i.e. they are hypernyms, hyponyms or siblings to each other we count the number of semantic links between these words along their shortest path in WordNet hierarchy. The identifier-similarity score between two such terms is calculated by dividing 6 by the number of links found between them. Thus, the term-similarity

score is a function of the terms' semantic distance in the WordNet hierarchy: terms that are farther away from each other have smaller similarity scores than terms that are located closer to each other in WordNet. Similar to the WordNet-based information-retrieval step, word senses are not disambiguated.

```
double matchDocumentTerms (term1, term2) {
  maxScore = 10;
  if (term1 is identical to term2)
    score = maxScore;
  else if (term1 and term2 are synonymous)
    score = 8;
  else if (term1 and term2 have hierarchical relations)
    score = 6 / number of hierarchical links;
  else score =0;
  return score; }
```

Fig. 1. Matching Document Terms Using WordNet.

In the end, the semantic structure-matching score between a web service S and a query service Q, $\mathrm{Sim}_{\text{semantic-structure-matching}}(S,Q)$, is a function of its structure matching score and its identifier matching score as follows:

$$\mathrm{Sim}_{\text{semantic-structure-matching}}(S,Q) = \mathrm{Sim}_{\text{structure-matching}}(S,Q) + \mathrm{Sim}_{\text{identifier-matching}}(S,Q) \quad (2)$$

In the above formula, $\mathrm{Sim}_{\text{structure-matching}}(S,Q)$ and $\mathrm{Sim}_{\text{identifier-matching}}(S,Q)$ are similarity scores calculated by the structure matching and the identifier matching method respectively. We assume that programmers follow Java-style naming conventions and use meaningful names for methods and data types. Under this assumption, all identifiers and names are broken into tokens by identifying delimiter characters such as underscores and capital letters.

4 Evaluation

To evaluate our service-discovery method as a whole and the effectiveness of its constituent elements, we had to obtain families of related specifications in order to evaluate the degree to which our algorithm can distinguish among them. We found such a collection published by XMethods [17]. The XMethods collection provided us with nineteen service descriptions from five categories: currency rate converter (three services), email address verifier (three services), stock quote finder (four services), weather information finder (four services), and DNA information searcher (five services).

In this section, we report on four sets of experiments: service discovery with WordNet-powered vector space model, discovery with structure matching, discovery with semantic structure matching, and discovery with WordNet-powered vector space model combined with semantic structure matching.

4.1 WordNet-Powered Vector-Space Model

In this experiment, we matched service descriptions specified in natural language of each service from each category (requests) against the text descriptions of all other services from all categories (candidates).

Table 1. WordNet-Powered Vector Space Model on the XMethods collection

Requests	Retrieved Matching Advertisements (Category: service name)
Currency rate converter Precision: 33% Recall: 100%	Currency: CurrencyExchangeService Currency: PwspNoCentrebankCurRates Currency: Currencyws Weather: TemperatureService DNA: TxSearch Stock: StockQuotes1 Weather: WeatherService Email: advancedemailcheckService Stock: StockQuotes (2)
DNA info Searcher Precision: 55% Recall: 100%	DNA: TxSearch DNA: Fasta DNA: ClustalW DNA: Blast Email: advancedemailcheckService Email: DOTSEmailValidate DNA: SRS Currency: pwspNoCentrebankCurRates Stock: StockQuotes (1)
Email Address Verifier Precision: 33% Recall: 100%	Email: AdvancedemailcheckService Email: DOTSEmailValidate Email: ValidateEmail Stock: MBSoapService(1) Stock: MBSoapService (2) DNA: Blast Stock: StockQuotes1 Stock: StockQuotes2 Weather: getCAWeatherService
Stock Quote Finder Precision: 44% Recall: 100%	Stock: MBSoapService(2) Stock: MBSoapService (1) Stock: StockQuotes1 Stock: StockQuote2 Weather: WeatherService Weather: TemperatureService Email: DOTSEmailValidate Weather: getCAWeatherService Weather: USWeather
Weather Info Finder Precision: 44% Recall: 100%	Weather: USWeather Weather: TemperatureService Stock: StockQuotes (1) Weather: WeatherService Stock: StockQuotes (2) Weather: getCAWeatherService Currency: CurrencyExchangeService

	Currency: Currencyws Stock: MBSoapService (2)

The similarity score between a given web service *S* and service requests from a given category *C* is the average of the similarity scores calculated between *S* and each request from category *C*. The candidate web services are ranked according to their similarity to the requests and the top 50% of services in the list are returned. We assume that if a web service ranks in the second half of the list, chances are that this web service is irrelevant to the request. Table 1 summarizes the results of this experiment.

We evaluate the effectiveness of our retrieval methods by calculating their precision and recall. "Precision is the proportion of retrieved documents that are relevant, and recall is the proportion of relevant documents that are retrieved" [13]. Average precision and recall for each test collection from each category of service requests are calculated and are listed in the first column of Table 1. Retrieved matching service advertisements are listed in column 2 of Table 1. They are sorted according to their similarity to requests from a given category. The WordNet-powered vector space model achieves a precision of 41.8% at 100% recall on average on this set of experiments.

4.2 Structure Matching

Experiments with structure matching were conducted in a similar manner: we matched the structure of each service from each category (requests) against the structures of all other services from all categories (candidates). Averages are calculated between service requests from each category and all candidate services. The candidate web services are ranked according to their similarity scores to the requests, and the top 70% of the list are returned. The results of this set of experiments are listed in Table 2.

Table 2. Structure Matching on the XMethods Collection

Requests	**Retrieved Matching Advertisements (Category: service name)**
Currency rate converter Precision: 14% Recall: 67%	Email: AdvancedemailcheckService Stock: StockQuote1 DNA: TxSearch, Blast, Fasta Stock: MBSoapService(1) Stock: MBSoapService (2) DNA: ClustalW, SRS Stock: StockQuote2 Email: ValidateEmail Currency: Currencyws Weather: USWeather Currency: pwspNoCentrebankCurRates
DNA info Searcher Precision: 36%	DNA: Fasta Stock: MBSoapService(1) Stock: MBSoapService (2) DNA: Blast

Recall: 100%	Currency: Currencyws DNA: ClustalW DNA: SRS DNA: TxSearch Email: DOTSEmailValidate Stock: StockQuote1, StockQuote2 Weather: USWeather Email: ValidateEmail Weather: WeatherService
Email Address Verifier Precision: 14% Recall: 67%	Email: DOTSEmailValidate Stock: StockQuote1 DNA: TxSearch DNA: Blast Currency: pwspNoCentrebankCurRates DNA: SRS Currency: Currencyws Stock: MBSoapService(1), Stock: MBSoapService(1) DNA: ClustalW, SRS, TxSearch Email: DOTSEmailValidate Stock: StockQuote1, StockQuote2 Weather: USWeather Email: ValidateEmail Weather: WeatherService
Stock Quote Finder Precision: 28% Recall: 100%	Email: DOTSEmailValidate DNA: Fasta, SRS, ClustalW, TxSearch Stock: MBSoapService(1) Stock: MBSoapService(2) Weather: USWeather DNA: Blast Stock: MBSoapService(1) Stock: MBSoapService(1) Currency: CurrencyExchangeService Currencyws, PwspNoCentrebankCurRates
Weather Info Finder Precision: 7% Recall: 25%	Email: DOTSEmailValidate Stock: MBSoapService(1) Currency: Currencyws Stock: MBSoapService(2) DNA: Blast, TxSearch Stock: MBSoapService(1) Stock: MBSoapService(2) DNA: ClustalW, Fasta, SRS Email: ValidateEmail Currency: CurrencyExchangeService Weather: USWeather

Average precision and recall are calculated for each set of queries, and on average, structure matching achieves a precision of 20% at 72% recall. Both precision and recall are considerably poor in this set of experiments because some related services have substantially different structures and some irrelevant services can often have higher matching scores because they have many spurious substructures that happen to match the query structure.

4.3 Semantic Structure Matching

In this experiment, we matched the services' structures and their chosen identifiers. The experiments were conducted in a similar manner as the experiments of sections 4.1 and 4.2 described above. Web services are ranked according to their similarity scores to the requests, and top 50% of web services on the lists are considered to be relevant to the requests and are returned to the users. The results of this set of experiments are listed in Table 3.

Semantic structure matching method achieves a precision of 35.2% at 81.8% recall on average. Please note that compare to performance of pure structure matching method, precision is improved by 15.2% from 20% and recalled is improved by 9.8% from 72%. Based on this experiment, we can infer that considering the implicit semantics of the WSDL identifiers is, in fact, enabling a more precise service matching.

Table 3. Semantic Structure Matching on the XMethods collection

Requests	**Retrieved Matching Advertisements (Category: service name)**
Currency rate converter Precision: 22% Recall: 67%	Email: DOTSEmailValidate Stock: StockQuotes (1) Weather: WeatherService Currency: Currencyws DNA: Blast, TxSearch, Fasta Stock: StockQuotes (2) Currency: pwspNoCentrebankCurRates
DNA info Searcher Precision: 55% Recall: 100%	DNA: Blast, Fasta, SRS, ClustalW, TxSearch Currency: Currencyws Stock: MBSoapService (1) Stock: MBSoapService (2) Email: DOTSEmailValidate
Email Address Verifier Precision: 22% Recall: 67%	Stock: StockQuotes (1) Email: DOTSEmailValidate Currency: pwspNoCentrebankCurRates Email: ValidateEmail Currency: Currencyws Weather: USWeather Stock: StockQuotes (2) Weather: WeatherService DNA: Blast
Stock Quote Finder Precision: 44% Recall: 100%	Email: DOTSEmailValidate Stock: MBSoapService (1) Stock: MBSoapService (2) Currency: Currencyws Stock: StockQuotes (2) Stock: StockQuotes (1) Currency: pwspNoCentrebankCurRates DNA: Blast DNA: Fasta
Weather Info Finder	Email: DOTSEmailValidate Stock: StockQuotes (1) Weather: USWeather

Precision: 33% Recall: 75%	Currency: Currencyws Weather: WeatherService Currency: pwspNoCentrebankCurRates Stock: StockQuotes (2) Email: ValidateEmail Weather: getCAWeatherService

4.4 WordNet-Powered Vector-Space Model and Semantic Structure Matching

Looking at the services retrieved with each query in the various experiments, we noticed that each method "picks" different types of similarity, which led us to hypothesize that their combination might be more effective than the best one of them. To investigate this hypothesis we conducted a fourth set of experiments where WordNet-powered vector space model and semantic structure matching method are combined. The WordNet-powered vector-space model was first used on all services as described in section 4.1 to obtain relevant web services compared to the query (top 50% of the services in the ranked list). Then, semantic structure matching was applied to the pruned list of candidates as described in section 4.3. The candidate services were matched and re-ranked, and the top 50% of the services in the list were returned. Therefore, after the two-step matching and refining, only the top 25% of all web services are returned as relevant services. The results of these experiments are shown in Table 4.

Table 4. WordNet-Powered Vector Space Model and Semantic Structure Matching on the XMethods Collection

Requests	**Retrieved Matching Advertisements (Category: service name)**
Currency rate converter Precision: 60% Recall: 100%	Currency: CurrencyExchangeService Currency: Currencyws Stock: StockQuotes (1) Currency: pwspNoCentrebankCurRates Weather: WeatherService
DNA info Searcher Precision: 100% Recall: 100%	DNA: Fasta DNA: ClustalW DNA: TxSearch DNA: Blast DNA: SRS
Email Address Verifier Precision: 60% Recall: 100%	Email: DOTSEmailValidate Email: ValidateEmail Stock: StockQuotes (1) Email: advancedemailcheckService Stock: StockQuotes (2)
Stock Quote Finder Precision: 80% Recall: 100%	Stock: MBSoapService (2) Stock: MBSoapService (1) Email: DOTSEmailValidate Stock: StockQuotes (1) Stock: StockQuotes (2)
Weather Info Finder	Weather: USWeather Stock: StockQuotes (1)

Precision: 60% Recall: 75%	Weather: TemperatureService Weather: WeatherService Currency: Currencyws

On average,this retrieval method that uses both the WordNet-powered vector space model and the semantic structure matching achieves a precision of 72% at 95% recall. Compared to the performance of WordNet-powered vector space model, precision is increased by 30.2% from 41.8% and recall dropped by 5% from 100%. Both precision and recall improved significantly compared to the results obtained by semantic structure matching method alone.

5 Conclusions and Future Work

In this paper, we described a web-service discovery method that combines two WordNet-based techniques with a structure-matching algorithm leveraging the structure of the XML-based service specification in WSDL. Currently developers can only browse UDDI registries and query the advertised services by business category. This is a very blunt and imprecise service-discovery mechanism.

Our web-service discovery method is inspired by traditional information retrieval methods, signature matching methods and many experiments conducted with WordNet for component retrieval. It is designed to calculate semantic and structural similarity between a desired service and a set of advertised services. WordNet-based methods do not attempt to resolve word senses; this problem has been proven difficult by current research, but fortunately it does not apply in the case of the WSDL descriptions, comments and identifiers, which are not likely to be complete grammatical sentences. WordNet is used as a "query expansion" mechanism: it includes semantically similar words retrieved from WordNet database for all documents and queries to ameliorate information retrieval results. The structure-matching algorithm respects the structural information of data types and is flexible enough to allow relaxed matching and matching between parameters that come in different orders in parameter lists. Our web service discovery method that combines WordNet-powered vector space model with semantic structure matching constitutes an important extension to the UDDI API, because it enables a substantially more precise service-discovery process.

We have conducted various experiments to evaluate the effectiveness of our retrieval system with very positive results. In the future, we plan to extend this algorithm to exploit the full WSDL syntax. Currently, we are not considering some of the syntax WSDL offers such as minOccurs, maxOccurs that indicate minimum and maximum occurrences of data types, and some other attributes of element tags. We also plan to experiment with larger sets of web services.

Acknowledgements

The authors wish to thank Tu Hoang for his help in developing parts of these algorithms. This research was supported by an IRIS grant.

References

1. I. Cho, J. McGregor, and L. Krause. "A protocol-based approach to specifying interoperability between objects". In Proceedings of the *26th Technology of Object-Oriented Languages and Systems* (TOOLS'26), 3–07 August 1998, Santa Barbara, CA, 84-96. IEEE Press.
2. The DARPA Agent Markup Language Homepage. http://www.daml.org/
3. C. Faloutsos, and D.W. Oard. "A survey of Information Retrieval and Filtering Methods, University of Maryland". Technical Report CS-TR-3514, August 1995.
4. K. Sycara, S. Widoff, M. Klusch and J. Lu. "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace". *Autonomous Agents and Multi-Agent Systems*, 5, 173–203, 2002.
5. G.A. Miller, R. Beckwith, C. Felbaum, D. Gross and K. Miller, "Introduction to WordNet: An On-line Lexical Database", International Journal of Lexicography, Vol. 3, No.4, 1990, 235-244.
6. G.A. Miller, "Nouns in WordNet: A Lexical Inheritance System", International Journal of Lexicography, Vol3, No.4, 1990, 245-264.
7. R. Mandala, T. Takenobu and T. Hozumi. "The Use of WordNet in Information Retrieval," in *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, Montreal, 1998, 31-37.
8. J. Purtilo and J.M. Atlee. "Module Reuse by Interface Adaptation". Software Practice and Experience, Vol. 21, No. 6, 1991, 539-556.
9. R. Richardson and A.F. Smeaton. "Using WordNet in a knowledge-based approach to information retrieval." Dublin City University School of Computer Applications Working Paper CA-0395.
10. G. Salton, A. Wong and C.S. Yang. "A vector-space model for information retrieval", In Journal of the American Society for Information Science, Vol. 18. November 1975, 13-620. ACM Press.
11. Simple Object Access Protocol (SOAP) http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
12. UDDI technical paper, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf
13. E. Voorhees. "Using WordNet for Text Retrieval", in C.Fellbaum (ed.), *WordNet: An Electronic Lexical Database* 1998, The MIT Press, Cambridge, MA. 1999, 285-303.
14. Y. Wang and E. Stroulia. "Flexible Interface Matching for Web-Service Discovery". In Proceedings of *4th International Conference on Web Information Systems Engineering*, December 10th - 12th, 2003 (to appear).
15. Web Services Description Language (WSDL) (WSDL) http://www.w3.org/TR/wsdl
16. WordNet http://www.cogsci.princeton.edu/~wn/
17. XMethods homepage. http://www.xmethods.com/
18. A. M. Zaremski and J. M. Wing. "Signature Matching: a Tool for Using Software Libraries". ACM Transactions on Software Engineering and Methodology, Vol. 4 No. 2, 146-170, Apr. 1995.

19. A. M. Zaremski and J. M. Wing. "Specifications Matching of Software Components". ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 4, 333-369, Oct. 1997.