Semantic Structure Matching for Assessing Web Service Similarity

Yiqiao Wang, Eleni Stroulia
University of Alberta

Presentation Outline

- Introduction and Motivation
- Related Work
- Web Service Similarity Assessment Methods
- Experiments and Evaluation
- Conclusion and Future Work

The Research Problem - Motivation

- The great opportunity:
  - Huge amounts of information and applications are available through the World Wide Web
  - Web-based applications constitute a substantial percentage of all developed applications
- The price of this availability:
  - Users have to decide:
    - which resources to use
    - how to interpret the information and services
    - how to combine them to accomplish their overall tasks

Current Web Services Standards

- The objective behind the design of the web-service stack of standards is reuse and interoperation of software components.
  - WSDL (Web Service Description Language)
    - How to specify reusable components
  - SOAP (Simple Object Access Protocol) API
    - How to invoke the services specified in WSDL
  - UDDI (Universal, Discovery, and Integration) API
    - How to advertise and discover WSDL services
- A critical step of reusing existing software components is the discovery of potentially relevant components.

Web Service Discovery through UDDI

- UDDI servers are catalogs of published WSDL specifications of reusable components.
  - Providers advertise services to the appropriate categories in UDDI.
  - Software developers can browse the UDDI catalog by category.
- This category-based service-discovery method is insufficient:
  - Service providers and developers must publish and browse the services in the appropriate UDDI category - shared understanding
  - Method does not provide support for selecting among competing alternative services that could potentially be reused.

The Semantic Web Solution

- Semantic web efforts support web-service discovery process by proposing a full-fledged ontology:
  - Defining domain-specific semantics and capabilities of web services
  - This definition process is costly
- We propose web service similarity assessing and discovery methods:
  - Light-weight natural-language based semantics with structure matching
  - Enable more precise discovery process at a low cost

Presentation Outline

- Introduction and Motivation
- Related Work
- Web Service Similarity Assessment Methods
- Experiments and Evaluation
- Conclusion and Future Work

Related Work - Component Retrieval

- Signature Matching
  - Polylith: one of the earliest signature matching methods, based on NIMBLE [PA91]
    - Coercion rules could be specified so that the parameters of the invoking module could be matched to the signature of the invoked module
  - Zaremski and Wing described exact and relaxed signature matching [ZW95].
- Specification Matching
  - Compares software components based on their functional behaviors.
  - Zaremski and Wing extended their signature-matching work with a specification-matching scheme [ZW97].

Related Work - Information Retrieval

- Traditional IR relies on textual descriptions of artifacts to assess their similarity
  - Vector Space Model
    - Each document is represented as a T-dimensional vector
    - Each term in the vector is assigned a weight reflecting its importance in the document
    - Similarity between two documents are assessed based on their representing vectors.

WordNet for Information Retrieval

- WordNet is a lexical database for the English language.
- Nouns, verbs, adjectives and adverbs are organized into sets
  - Each set represents one underlying lexical concept
- Relationships between two concepts X and Y include:
  - Synonym: concepts X and Y have similar meanings
  - Hypernym (Parent) and Hyponym (Child):
    - X is hypernym of Y → Y is hyponym of X → Y is "a kind of" X.
  - Sibling:
    - X and Y are siblings → X and Y have a common hypernym
- WordNet achieves limited success in ameliorating traditional information-retrieval results.

Presentation Outline

- Introduction and Motivation
- Related Work
- Web Service Similarity Assessment Methods
- Experiments and Evaluation
- Conclusion and Future Work

Web Service Discovery Methods

- The research question:
  - How can we accomplish "semantic" matching without the cost of semantically annotating WSDL specifications?
- The insight:
  - Service descriptions, their syntactic structures, and the chosen identifiers in WSDL specifications capture (some) semantics
- The intuition - Query by example
  - We provide textual descriptions and/or WSDL specification of the desired service to the suite of methods [WS03a, WS03b].

Example: Service Operations

- Desired example service getData's Operation

```
<portType name="getData">
     <operation name="getDataById">
          <documentation> search data type with a unique Id
          </documentation>
          <input message="getDataByIdRequest" />
          <output message="getDataByIdResponse" />
     </operation>
</portType>
```

- Candidate service getProduct's Operation

```
<portType name="getProduct">
    <operation name="getProductByNumber">
        <documentation> search product by id number
        </documentation>
        <input message="getProductByNumberRequest" />
        <output message="getProductByNumberResponse" />
    </operation>
</portType>
```

Service Messages - Request

- Operation getDataByID's Request Message

```
<message name="getDataByIdRequest">
    <documentation> method takes in a string as ID
    </documentation>
    <part name="id" type="string"/>
</message>
```

- Operation getProductsByNumber's Request Message

```
<message name="getProductByNumberRequest">
    <documentation> this method takes a number for
identification </documentation>
    <part name="number" type="int"/>
</message>
```

Service Messages - Response

- Operation getDataByID's Response Message

```
<message name="getDataByIdResponse">
     <documentation> method returns a product with
specified  ID number </documentation>
     <part name="data" type="DataType"/>
</message>
```

- Operation getProductsByNumber's Response Message

```
<message name="getProductByNumberResponse">
     <documentation> returns a product type with the
number   </documentation>
     <part name="product" type="productType"/>
</message>
```

Service Data Types

Service getData's type - DataType

```
<types>
  <schema>
    <complexType name="DataType">
      <all>
        <element name="id"
          type="string"/>
        <element name="category"
          type="string"/>
        <element name="items"
          type="Item">
      </all>
    </complexType>
    <complexType name="Item">
      <all>
        <element name="quantity"
          type="int"/>
        <element name="item"
          type="string"/>
      </all>
    </complexType>
  </schema>
</types>
```

Service getProduct's type - ProductType

```
<types>
  <schema>
    <complexType name="ProductType">
      <all>
        <element name="number"
          type= "int"/>
        <element name="description"
          type="string"/>
        <element name="price"
          type="float"/>
        <element name="part"
          type="ProductPart"/>
      </all>
    </complexType>
    <complexType name="ProductPart">
      <all>
          <element name="part"
          type="string"/>
       </all>
    </complexType>
  </schema>
</types>
```

Vector Space Model

- Documents and queries are represented as T-dimensional vectors
  - T is the total number of distinct words in a document
- Each term in the vector is assigned a weight:

$$w_{ij} = tf_{ij} \cdot idf_i = tf_{ij} \cdot \log_2 \left(\frac{N}{df_i} \right)$$

  - tf_{ij} : frequency of term i in document j
  - idf_i : the inverse document frequency of term i
- Similarity between a document vector, d, and a query vector, q, can be computed as the vector inner product:

$$Sim(d, q) = d \cdot q = \sum_{i=1}^{t} w_{id} \cdot w_{iq}$$

WordNet-Powered Vector Space Model

- Vector Space Model is extended with WordNet by including semantically similar words of service descriptions
  - Service descriptions' synonyms, direct hypernyms (parents), hyponyms (children), and siblings are retrieved from WordNet.
  - Three corresponding vectors are maintained:
    - Vector 1: Stems of original textual service descriptions
    - Vector 2: Stems of original description terms' synonyms
    - Vector 3: Stems of original description terms' direct hypernyms, hyponyms, and siblings (family of term).

WordNet-Powered Vector Space Model - Example

- Corresponding sub-vectors from the desired and the candidate services are matched using vector space model
  - We obtain three corresponding similarity scores.
  - Different weights are assigned to sub-vector matching scores:
  - Overall similarity score between two services is the sum of their corresponding sub-vector matching scores.
- Example
  - getData vs. getProduct, Overall similarity score is 5.2029
    - Original terms: 0.2192; Synonyms: 2.0875, Family: 0.3703
  - getData vs. currencyConverter, Overall similarity score is 0
    - Original terms: 0; Synonyms: 0, Family: 0

Structure Matching

- Matching service's data types
  - All pair-wise combinations of source and target data types are compared.
- Matching service's messages
  - All pair-wise combinations are compared.
- Matching service's operations
  - All pair-wise combinations are compared.
- Matching web services
  - The overall matching score of two services is the pair-wise correspondence of their operations that maximizes the sum of the matching scores of the individual pairs.

Matching Data Types - Properties

- ***Property 1***: Preference is given to the matches between data types with the same grouping organization of their elements.
  - Three organization styles: <all>, <sequence>, or <choice>
  - Bonus score of 10 is added to matches of data types with the same organization style
- ***Property 2***: If two data types have the same name and they are imported from the same namespace, they are identical data types and an exhaustive match is unnecessary.

Matching Data Types: Algorithm

```
int matchDataTypes (sourceList(m), targetList(n)) {
    matrix = construct a m⊗n matrix;
    for (int i=0; i<m; i++) {
       for (int j=0; j<n; j++) {
          sourceType = sourceList(i)
          targetType = targetList(j)
          if (both sourceType and targetType are primitive data types)
             matrix[i][j] = matchPrimitiveTypes (sourceType, targetType);
          else {
             if (both sourceType and targetType share the same name and namespace)
                matrix[i][j] = matchIdenticalTypes(sourceType, targetType);
             else {
                newSourceList = getCompositeDataElements(sourceType);
                newTargetList = getCompositeDataElements(targetType);
                matrix[i,j] =   matchDataTypes (newBaseList, newTargetList)
                              + organizationBonus(sourceType, targetType);
             }
          } } }
    find all possible matches between sourceList and targetList according to matrix;
return the score of the best matches;
}
```

Matching DataType & ProductType

DataType matches ProductType with a score of 45:

35 + 10 bonus for <all>=<all>

		ProductType			
		Number: int	Description : string	Price: float	ProductPart
DataType	Id: string	**5**	10	5	?→10
	Category: string	5	**10**	5	?→10
	Item	?→10	?→10	?→5	?→**20**

Item matches ProductPart with a score of 20:

10 + 10 bonus

		ProductPart
		Part: string
Item	Quantity: int	5
	Item: string	**10**

Matching Services getData and getProduct

- Matching Messages
  - Matching input messages: 5
  - Matching output messages: 45
- Matching Operations: 5+ 45 = 50
- Matching Services: 50

Identi?er Matcher

- Matching identifiers of data types
  - Best pair-wise combinations of source and target data type identifiers.
- Matching operations
  - The overall matching score of two operations is the sum of
    - Operation name matching score
    - Score of best pair-wise correspondence of their data type identifiers
- Matching web services
  - The overall matching score of two services is the sum of
    - Service name matching score
    - Score of best pair-wise correspondence of their operations

Matching Two Words - Algorithm

- Algorithm ***matchDocumentTerms*** assesses semantic distance between two document terms utilizing WordNet

```
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 between terms;
  else score = 0;
  return score;
}
```

Match Identi?ers - Example

DataType matches ProductType with a score of 32:

22 + 10 bonus for <all>=<all>

		ProductType			
		Number	Description	Price	ProductPart
DataType	Id	**3**	0	0	?→0
	Category	0	**3**	0	?→3
	Item	?→3	?→0	?→0	?→**16**

Item matches ProductPart with a score of 16:

6 + 10 bonus

		ProductPart
		Part
Item	Quantity	3 (terms are siblings)
	Item	**6** (terms are direct hypernyms)

Match Services - Example

- getData and getProduct's 'return' Operations match with a score of 41 (6 + 35)
  - Operations' names match with a score of 6
  - Operations' parameter lists match with a score of 35
- Services match with a score of 41 (0 + 41 + 8).
  - Service names match with a score of 0
  - Service request operations match with a score of 0
  - Service return operations match with a score of 41

Presentation Outline

- Introduction and Motivation
- Related Work
- Web Service Similarity Assessment Methods
- Experiments and Evaluation
- Conclusion and Future Work

Experiments and Evaluation

- Experiments used XMethods collection [XMethods]
  - Xmethods service collection: 19 services from 5 categories
    - Currency rate converter (3 services)
    - Email address verifier (3 services)
    - Stock quote finder (4 services),
    - Weather information finder (4 services)
    - DNA information searcher (5 services).

Experiments

- Each set of experiments are performed in the same manner:
  - Each service from each category (query) is matched against all other services from all categories (candidates).
  - The similarity score between a given web service ***S*** and service requests from a given category ***C*** is the average of similarity scores calculated between *S* and each request from category *C*.
  - Candidate web services are ranked according to their similarity to the requests.
  - Only services that are ranked higher than a given threshold are returned.

Experiments (Cont)

- *Precision* and *Recall* are used to evaluate our methods.
  - *Precision* is the proportion of retrieved documents that are relevant
  - *Recall* is the proportion of relevant documents that are retrieved
- Evaluating retrieval method's performance
  - *Precision* and *recall* for each test collection from each category of requests were calculated
  - The retrieval method's performance was evaluated using average *precision* and *recall* of these test collections from all service categories.

Experimental Results

- Discovery with WordNet-Powered Vector Space Model

Service Requests	Precision	Recall
Currency Rate Converter	33%	100%
DNA Info Searcher	55%	100%
Email Address Verifier	33%	100%
Stock Quote Finder	44%	100%
Weather Info Finder	44%	100%
Average Performance	41.8%	100%

Experimental Results (Cont)

- Discovery with Structure Matching

Service Requests	Precision	Recall
Currency Rate Converter	14%	67%
DNA Info Searcher	36%	100%
Email Address Verifier	14%	67%
Stock Quote Finder	28%	100%
Weather Info Finder	7%	25%
Average Performance	**20%**	**72%**

Experimental Results (Cont)

- Discovery with Semantic Structure Matching

Service Requests	Precision	Recall
Currency Rate Converter	22%	67%
DNA Info Searcher	55%	100%
Email Address Verifier	22%	67%
Stock Quote Finder	44%	100%
Weather Info Finder	33%	75%
Average Performance	**35.2%**	**51.8%**

Experimental Results (Cont)

- Discovery with WordNet-Powered Vector Space Model and Structure Matching

Service Requests	Precision	Recall
Currency Rate Converter	60%	100%
DNA Info Searcher	100%	100%
Email Address Verifier	60%	100%
Stock Quote Finder	80%	100%
Weather Info Finder	60%	75%
Average Performance	**72%**	**95%**

Presentation Outline

- Introduction and Motivation
- Related Work
- Web Service Similarity Assessment Methods
- Experiments and Evaluation
- Conclusion and Future Work

Conclusion

- Proposed methods constitute an important extension to the UDDI API
  - They enable a substantially more precise service-discovery process.
  - Similarity between services can be assessed for selecting among competing alternative services that could potentially be reused.
- We investigated the effectiveness of natural-language based semantics combined with structure matching at a lower cost compared to that of semantic web efforts.

Future Work

- Extend the WordNet-Powered Vector Space Method
  - To consider the locations of textual service descriptions extracted from WSDL specifications
- Combining the Structure and Identifier Matching methods
  - To enforce consistent source to target structural and identifier mappings.
- Eliminate family group words (words' parents, children, siblings) from the WordNet-Powered Vector Space Model
  - Too many family group words were included, many of which are not related to their original document terms.
- Explore the full syntax of XML Schema in WSDL
  - To consider attributes such as minOccurs, maxOccurs

References

- [PA91]. J. Purtilo and J. M. Atlee. "Module Reuse by Interface Adaptation". *Software Practice and Experience*, 21(6), June. 1991.
- [ZW95]. A. M. Zaremski and J. M. Wing. "Signature Matching: a Tool for Using Software Libraries". *ACM Transactions on Software Engineering and Methodology*, 4(2): 146-170, Apr. 1995.
- [ZW97] A. M. Zaremski and J. M. Wing. "Specifications Matching of Software Components". *ACM Transactions on Software Engineering and Methodology*, 6(4): 333-369, Oct. 1997.
- [XMethods]. XMethods http://www.xmethods.com/
- [WordNet]. WordNet http://www.cogsci.princeton.edu/~wn
- [WS03a]. Y. Wang, and E. Stroulia. "Flexible Interface Matching for Web-Service Discovery". *To be in the proceedings of Web Information Systems Engineering. Dec 2003.*
- [WS03b]. Y.Wang, and E. Stroulia. "Semantic Structure Matching for Assessing Web-Service Similarity". *To be in the proceedings of The First International Conference on Service Oriented Computing*, Dec 2003.

Questions and Answers

{yiqiao, stroulia}@cs.ualberta.ca