Suggestions for Galaxy Workflow Design Using Semantically Annotated Services

Alok Dhamanaskar, Michael E. Cotterell, Jessica C. Kissinger, and John Miller.

University of Georgia

Jie Zheng and Christian J. Stoeckert, Jr.

University of Pennsylvania

Presented by: Jie Zheng
Outline

1. Web Service Composition Issues
2. Semantic Annotation of Web Services
3. Semi-Automatic Workflow Composition: Service Suggestion Engine (SSE)
   • Input-Output Matching Algorithms
   • Objective Specification Compliance
   • Calculation of Concept Similarity
4. SSE Evaluation
   • Interfacing SSE with the Galaxy workflow editor
5. Summary and Future Work
Web Service Workflow Composition Issues

- Web services are developed by different contributors
- Not developed to work with one another

Web services are described using either a WSDL (Web service Description Language) document or a WADL (Web Application Development Language) document.
  - No standard naming conventions, *e.g.*, operation, input
  - Text descriptions are inherently ambiguous

Hard to identify whether inputs/outputs are compatible
Semantic Web Service frameworks

- **OWL-S**
  - Upper level ontology for Web services
  - Top-down approach
  - Difficult to model the large number of existing Web services

- **SAWSDL**
  - Bottom-up approach
  - Provides extension attributes for adding semantics to Web services: `sawSDL:modelReference`
  - Easy to semantically model existing Web services
  - No specific semantic model or ontology language required to use
  - W3C recommended web services semantic annotation mechanism
Bioinformatics Web Service Ontology (OBI-WS)

- OBI WS v 1.0 released
  - [http://purl.obolibrary.org/obo/obi/webService.owl](http://purl.obolibrary.org/obo/obi/webService.owl)
  - Bioportal: [http://bioportal.bioontology.org/ontologies/3119](http://bioportal.bioontology.org/ontologies/3119)
- Supports annotation of ~100 operations from 19 different web services including sequence analysis and utility web services
  - Sequence Similarity Web Services
    - WU-BLAST, NCBI-BLAST
  - Multiple Sequence Alignment Web services
    - Clustal W, T-coffee
  - Protein Functional Analysis Web services
    - SignalP
  - Phylogenetic Analysis Web services
    - Phylip
Service Suggestion Engine

• The Service Suggestion Engine (SSE) is a semi-automatic workflow composition system.

• **What it does:**
  - Facilitates the construction and extension of workflows by providing *suggestions* to the user for the next step.
  - It is capable of doing Forward Suggestions, Backward Suggestions and Bi-directional Suggestions.
  - SSE returns a ranked list of web service operations (from a candidate list of available operations) that could be used for the *next, previous or intermediate* step.
  - SSE ranking is based on Compatible score
Web Service Operations Compatible Scores Calculation

- The SSE scores the candidate operations depending on
  - How well the inputs of the candidate operation can be fed using the outputs of the operation(s) in the workflow.
    - **Input-Output Compatibility Score** $S_{\text{io}}$
  - How well the desired functionality if supplied by the user aligns with the “Objective specification” of the operation.
    - **Objective Specification Compliance Score** $S_{\text{obj}}$
SSE Sub-scores

<table>
<thead>
<tr>
<th>Sub-score</th>
<th>Weight</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{syn_{io}}$</td>
<td>$(1 - \sigma)(1 - \varphi)$</td>
<td>input-output syntactic sub-score</td>
</tr>
<tr>
<td>$S_{sem_{io}}$</td>
<td>$\sigma(1 - \varphi)$</td>
<td>input-output semantic sub-score</td>
</tr>
<tr>
<td>$S_{syn_{obj}}$</td>
<td>$(1 - \sigma)\varphi$</td>
<td>objective specification syntactic sub-score</td>
</tr>
<tr>
<td>$S_{sem_{obj}}$</td>
<td>$\sigma\varphi$</td>
<td>objective specification semantic sub-score</td>
</tr>
</tbody>
</table>

Table 2.1: Description of sub-scores calculated by SSE

$\sigma$: weight of objective specification  
$(1- \sigma)$: weight of input-output  
$\psi$: weight of semantic  
$(1- \psi)$: weight of syntactic

Sum of weights = 1

- When calculating these sub-scores, the system calls a “Concept similarity” module to find out how similar two concepts are.

Input-Output Compatibility

- Model inputs/outputs using a Directed Acyclic Graph (DAG)
- Transforms the I-O matching problem into a graph pattern matching problem
- SSE supports two algorithms to calculate $s_{io}$ sub-score
  - path based mapping algorithm
  - p-Homomorphism matching algorithm
Path Based Mapping Algorithm

Output of Web service 1

Input to Web service 2

Identify Paths

1. A – B – D
2. A – B – E
3. A – C

Identify Paths

1’ P – Q
2’ P – R – S

• The goal is to **match** the **input paths** of Web service 1 to **output paths** of Web service 2.
• So in this case we need to find best matching path for
  1’ P – Q
  2’ P – R – S
Path Based Mapping Algorithm

Finding best matching path for
2’ $P \rightarrow R \rightarrow S$

Comparing paths
- **2’ – 1** $P \rightarrow R \rightarrow S$
  $A \rightarrow B \rightarrow D$
- **2’ – 2** $P \rightarrow R \rightarrow S$
  $A \rightarrow B \rightarrow E$
- **2’ – 3** $P \rightarrow R \rightarrow S$
  $A \rightarrow C$

$w_1 \times \text{SyntacticSimilarity}(P, A) + w_2 \times \text{SemanticSimilarity}(P, A)$

$w_1 \times \text{SyntacticSimilarity}(R, B) + w_2 \times \text{SemanticSimilarity}(R, B)$

$w_1 \times \text{SyntacticSimilarity}(S, D) + w_2 \times \text{SemanticSimilarity}(S, D)$

**Best Matching Path:** max{ $\text{MatchPath}(2’ – 1)$, $\text{MatchPath}(2’ – 2)$, $\text{MatchPath}(2’ – 3)$ }
Path Based Mapping Algorithm

- The final Data mapping score would be weighted sum of best matching paths for all the paths in the input.
- The sub scores are calculated as follows:
  - $\text{SyntacticSimilarity}(P,A)$: is computed using various string matching algorithms
  - $\text{SemanticSimilarity} (P,A)$: is computed as Concept Similarity between concepts that are used to annotated both the node, $\text{ConceptSimilarity} (\text{Concept}_p, \text{Concept}_a)$
- The Concept similarity has been developed as an independent module that computes similarity score between two concepts in the ontology that considers
  - Textual Definitions in the Ontology
  - Logical Definitions in the Ontology (Properties and Restrictions)
  - The hierarchical position of the two concepts in the Ontology
P-Homomorphism
Input-Output Matching

• Path-Based matching decomposes the input-output DAGs into individual paths thus losing some structural Information.

• A homomorphism is a structure-preserving map between two algebraic structures (such as groups, rings, or vector spaces)

• Finding an exact Homomorphism mapping between inputs and output DAGs is remote
  • Important to consider similarity between the vertices when considering the mapping
P-Homomorphism
Input-Output Matching

Good Matches
A -> P
B -> R
C ->
D -> Q
E -> S

Threshold 0.7

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.8</td>
<td>0.2</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>0.4</td>
<td>0.4</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>C</td>
<td>0.2</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>D</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>E</td>
<td>0.3</td>
<td>0.5</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

\[(1 - \varphi)(\sigma \text{SemSim}(\text{concept}_u, \text{concept}_{u'}) + (1 - \sigma) \text{SynSim}(\text{label}_u, \text{label}_{u'}))\]
Objective Specification Compliance

- User can provide the desired functionality as:
  - keywords or
  - a concept in the ontology that he feels closely describes the functionality desired

- If semantics are available from the user as well as Annotation:
  - ConceptSimilarity( ObjectiveSpecification, DesiredFunctionality )

- In absence of semantics from the user:
  - SyntacticSimilarity( ConceptDefinition, Keywords )

- In absence of semantics from the user as well as annotation:
  - SyntacticSimilarity( OperationName, Keywords )
Workflow Management System

Two popular tools that provide a GUI for creating workflows

- **Galaxy**
  - easy to use, open-source, Web-based platform that provides multiple tools for bioinformatics data analysis.
  - provides an easy way to construct workflows using existing tools in a very simple fashion using a Yahoo pipes-based graphical designer
  - Previous work allows adding WebServices as tools to Galaxy

- **Taverna**
  - open source
  - is integrated with BioCatalogue, and supports the invocation of web services and their use in workflows.
Evaluation Scenario

• Find out more information about a protein sequence and its evolutionary relationships to other protein sequences.

• The user might be aware that he wants to
  • first search a database for similar sequences,
  • then perform multiple sequence alignment and
  • finally perform phylogenetic analysis to construct phylogenetic trees.

• Web services already exist for each of the above.

• We utilize semantically annotated versions of their descriptions for our example.
Evaluation Set up

• We have evaluated SSE for Suggestions provided for each step against a ranking by a human expert.

• When suggesting from 101 Web service Operations

Precision:  \[ P = \frac{(\text{Relevant Results}) \cap (\text{Retrieved Results})}{\text{Retrieved Results}} \]

Recall:  \[ R = \frac{(\text{Relevant Results}) \cap (\text{Retrieved Results})}{\text{Relevant Results}} \]

F-Measure:  \[ F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \frac{\text{precision}}{\text{precision}} + \text{recall}} \]
Evaluation: Adding Web Services to Galaxy

Step b: Select desired Web Service Tools / Operations (version 1.1.0)

1. The previous step was (No action required):
   - [Select All] [Unselect All]
   - [getResultTypes]
   - [run]
   - [getStatus]
   - [getResult]
   - [setParameterDetails]
   - [getParameters]

You can also select any previously performed Step a from the history to avoid repeated execution of Step a.

2. The tool specified by the Web service is:
   - [http://www.ebi.ac.uk/Tools/services/soap/clustalw2?wsdl]
   - Displays the tool (Web service) chosen in Step a. No action required.

3. Select functions of the Web service tool to add to galaxy:
   - [Select All] [Unselect All]
   - [getResultTypes]
   - [run]
   - [getStatus]
   - [getResult]
   - [setParameterDetails]
   - [getParameters]

Some tools have more than one function. It is important to select the appropriate functions.

[TIP] After the completing step b proceed to step c

NOTE: Step a, Step b and Step c merely add this tool to Galaxy. To use the tool, go to the Tools section on left and find the tool under Select Web Service Tools and Select Web Service Workflow Tools.
Evaluation: Workflow Construction

Step 1

- wublast.run
- sequence
- style
- program
- email
- database
- output (tabular)

Previous Step:

Desired Functionality:

Add a concept from an ontology in order to help the suggestion engine.

Make Suggestions

1. getResult
code: run
0.777 (DM: 0.777, FN: 0.000)
Installed: add to workflow

2. getResult
code: run
0.777 (DM: 0.777, FN: 0.000)
Installed: add to workflow

3. getResult
code: run
0.777 (DM: 0.777, FN: 0.000)
Available: RadiantWeb!

4. getResult
code: run
0.777 (DM: 0.777, FN: 0.000)
Available: RadiantWeb!

5. getResult
code: run
0.777 (DM: 0.777, FN: 0.000)
Available: RadiantWeb!

6. getResult
code: run
0.777 (DM: 0.777, FN: 0.000)
Available: RadiantWeb!
Evaluation: Extend Workflow
Evaluation: Forward Suggestions (Path Based)
Conclusions & Future Work

- The use case demonstrated,
  - that of choosing from 101 operations for a 9-step workflow, presents a challenging task for a user (without a tool support).
- Semantics from ontology can definitely help in different aspects of Web service Compositions.
- SSE can help the user considerably narrow down the choices for the next or previous step.
- The availability of a SSE as a Web service will facilitate easier integration with existing workflow composition tools.

**Future work**

- Other kind of web services, like secondary structure analysis, sequence annotation web services, etc.
- plug-in for other workflow editors, *e.g.*, Taverna

- Code and files available at:
  - http://mango.ctegd.uga.edu/jkissingLab/SWS/index.html
Acknowledgements

Alok Dhamanaskar: alokdhamanaskar@gmail.com

University of Georgia
  • Michael E. Cotterell
  • Jessica C. Kissinger
  • John Miller

University of Pennsylvania
  • Jie Zheng
  • Christian J. Stoeckert, Jr.

Funding: NIH R01 GM093132
Adding ontology annotation into WSDL / WADL files

```xml
<wSDL:definitions name="wublast" targetNamespace="http://soap.jdispatcher.ebi.ac.uk">
  <wSDL:documentation>
  WU-BLAST stands for Washington University Basic Local Alignment Search Tool. The emphasis of this tool is to find regions of sequence similarity with minimal loss of sensitivity. This will yield functional and evolutionary clues about the structure and function of your novel sequence. Dr Warren Gish at WU created a "gapped" version of BLAST allowing for gapped alignments and statistics.
  </wSDL:documentation>

  <wSDL:types>
    <xsd:schema attributeFormDefault="unqualified" elementFormDefault="unqualified" targetNamespace="http://soap.jdispatcher.ebi.ac.uk">
      <xsd:complexType name="InputParameters">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">Input parameters for the tool</xsd:documentation>
        </xsd:annotation>
      </xsd:complexType>
    </xsd:schema>

    <xsd:complexType name="InputParameters">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">Input parameters for the tool</xsd:documentation>
      </xsd:annotation>
    </xsd:complexType>

    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="exp" nullable="true" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            Expectation value threshold [Limits the number of scores and alignments reported based on the expectation value. This is the maximum expected to occur by chance.]
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>

  </wSDL:types>
</wSDL:definitions>
```

(expectation value)
RadiantWeb Annotation Tool

- Manually annotation is labor intensive and error prone
- RadiantWeb – suggest ontology terms, generate SAWSDL file