Suggestions for Galaxy Workflow Design Using Semantically Annotated Services

Alok Dhamanaskar [a], Michael E. Cotterell [a], Jie Zheng [d], Jessica C. Kissinger [a,b,c], Christian J. Stoeckert, Jr. [d] and John A. Miller [a,b]

[a] *Dept. of Computer Science*
[b] *Institute of Bioinformatics*
[c] *Center for Tropical and Emerging Global Diseases and Dept. of Genetics*
University of Georgia, Athens, GA 30602
[d] *Penn Center for Bioinformatics and Dept. of Genetics*
University of Pennsylvania, Philadelphia, PA 19104

Abstract.
The wide-scale development of ontologies in the bioinformatics domain facilitates their use in the creation of scientific workflows. To speed up the design of workflows, a Service Suggestion Engine is interfaced to the Galaxy Tool Integration and Workflow Platform. This enables users to ask for suggestions (e.g., what operation should go next) while designing workflows with the Galaxy user interface. The Service Suggest Engine utilizes semantic annotations to suggest appropriate Web service operations to plug into the workflow under design. The enriched Ontology for Biomedical Investigation (OBI) is used as a target for the annotations. The effectiveness of the suggestions provided is evaluated against a human consensus.

Keywords. Semantic Web Services, Ontology, Semantic Annotations, Workflow, Web Service Composition

1. Introduction

The bioinformatics domain is witnessing an exponential rise in available data as more efficient, cheaper and faster means of sequencing, transcript analysis, etc. are developed. Mining this vast amount of data to gain useful insights often requires the coordinated use of multiple bioinformatics analysis tools. An increasing number of such tools and software applications are being provided as Web services by the biological and biomedical communities. For example, Biocatalogue, a registry of biological Web services, currently has information on 2,278 Web services from 161 service providers [10]. To utilize these Web services effectively, there is a need to rapidly construct scientific workflows composed of Web services.

Galaxy [2] is an easy to use, open-source, Web-based platform that provides multiple tools for data analysis and bioinformatics research. Galaxy provides a platform to construct workflows using existing Galaxy tools in a very simple fashion using a Yahoo pipes-based graphical designer. In our previous work [29], we created a tool, Galaxy

Web Service Extensions[1], that permitted the addition of Web services as tools within Galaxy. This addition made Web service composition possible within the Galaxy framework. However, two important problems remain: 1) selection of the appropriate operations (tools) to achieve the desired workflow outcome and 2) connection of the appropriate input parameters with the right output parameters.

To understand the complexities involved with selecting two tools, or Web services, such that they are input-output compatible with each other, let us consider the output and input of the Web service operations in Figure 1. On the left is the output of Web Service 1 and on the right is the input to Web Service 2. The input to Web Service 2 can be perfectly fed by the output of Web Service 1, exp to `e-val` and SequenceId to `Sid`. As it can be quite difficult for a naive or non-specialist researcher to assign the correct mapping, there is a need for a system to assist the user.

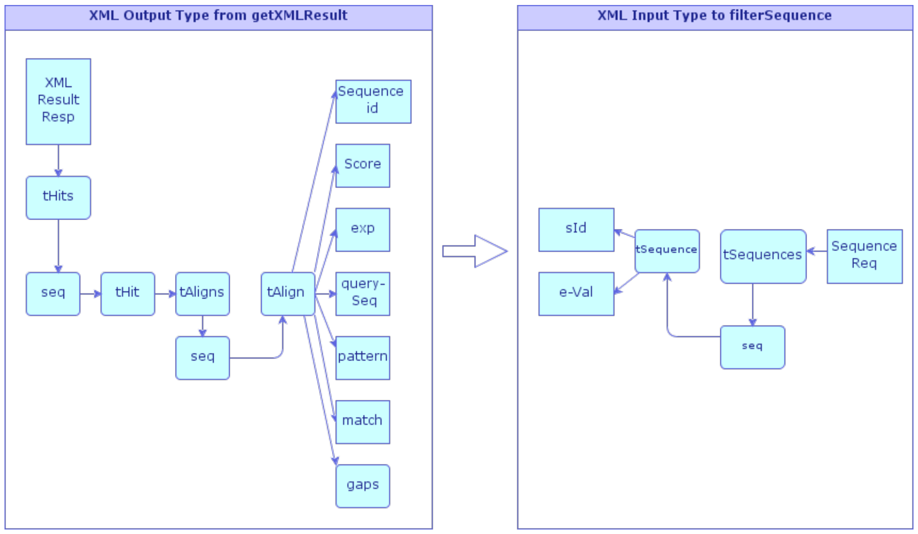


Figure 1. A representation of XML structures for the output and input of two different Web services

This paper extends our previous work and focuses on assisting the user by providing suggestions for the next possible Web service to use in the creation of bioinformatic and biomedical workflows that otherwise would need computer science as well as biological expertise to complete [31].

The result of our previous studies [30,31] was a Path-based data mediation algorithm, capable of providing service suggestions to help a user construct a desired workflow. The second work compared three data mediation algorithms (Leaf-based, Structure-based and Path-based) and the Path-based algorithm was usually found to work best among them. Here, we provide the following improvements and extensions:

- re-engineered code to improve calculation of metrics like Property Similarity,
- taking into account restrictions on concepts in the ontology when calculating Concept Similarity,

[1]Available at : `http://toolshed.g2.bx.psu.edu/repository/view_repository?id=94d0f039a25a883a`

- a better use of semantics to suggest the next operation and assist with connecting outputs to inputs,
- assist the user with possible input values and documentation about the different input-output parameters, and
- efficient interfacing with the existing Galaxy Workflow Editor to provide service suggestions within Galaxy.

SOAP [2] Web services are generally described by a Web Service Description Language (WSDL) document, which is a W3C Specification [3]. There is also a W3C Submission called Web Application Description Language (WADL) [4], a description language for REST[9] Web services or Web applications. Semantically annotating Web services involves adding references to terms in an Ontology for specific inputs, outputs and operations inside a WSDL/WADL file. W3C has recommended a specification, Semantic Annotation for WSDL (SAWSDL) [5], which defines a set of extension attributes for the WSDL language.

For a system to provide service suggestions that will help the user construct a desired workflow, the system should have a precise specification for what a tool does, the input it takes and the information it outputs. This generates the need to agree upon common vocabulary that would uniquely identify various aspects of the Web service or tool (i.e., the functionality of the operation performed, its inputs and outputs). Ontologies, explicit formal specifications of the terms in the domain and relations among them [11], are an ideal candidate to describe Web services (referred to as annotation of Web services) for a variety of reasons. Some ideal features of ontologies include:

- providing a rich modeling framework,
- enabling reuse of domain knowledge,
- facilitating formal community agreement,
- being Web accessible, and
- facilitates reasoning to ensure consistency.

The Ontology for Biomedical Investigations (OBI) [6], a member of the Open Biological and Biomedical Ontology (OBO) Library [28], is being developed to address the need for consistent descriptions of biological and clinical investigations, including data analysis. OBI includes terms that are applicable across various biological and technological domains. With the Basic Formal Ontology (BFO) [7] as an upper level ontology, OBI is interoperable with other OBO-compliant ontologies. OBI's existing structure makes it ideal for enrichment with concepts to support Web service annotations. We follow a systematic methodology for enriching OBI with terms to support Web service Annotation. This process involves the design of ontology analysis diagrams for Web services and their subsequent analysis to discover terms that need to be added to the ontology [12]. Analysis of the following Web services: `WUBLAST`, `NCBIBlast`, `PSIBlast`, `ClustalW2`, `TCoffee`, `WSDBFetch`, `WSConverter`, `Fasta`, `Muscle`, `WSFilterSequences` and `WSPhylip` has resulted in the identification of approximately 100 new ontological terms, which are cur-

[2]SOAP: `http://www.w3.org/TR/soap/`
[3]WSDL: `http://www.w3.org/TR/wsdl`
[4]WADL: `http://www.w3.org/Submission/wadl`
[5]SAWSDL: `http://www.w3.org/2002/ws/sawsdl/`
[6]The OBI Consortium: `http://purl.obolibrary.org/obo/obi`
[7]BFO: `http://www.ifomis.org/bfo`

rently pending approval by the OBI community. We are continuing to annotate additional Web services and tools in the bioinformatics domain. Since the terms we have proposed cover many of the fundamental concepts in Web services and bioinformatics analysis, we expect the number of additional terms required to annotate new services to decrease.

Once a Web service is semantically annotated, our algorithm calculates scores (data mediation and functionality) for candidate Web services and returns a ranked list of Web service operations or tools that can succeed the current operation. In section 2, we describe the details of the Service Suggestion Engine (SSE) and the recent improvements made relative to our previous effort. We discuss the interfacing of the Service Suggestion Engine with the Galaxy Workflow tool in section 3. In section 4, we present an evaluation of the SSE in terms of effectiveness of the suggestions. Sections 5 and 6 discuss related work and conclusions, respectively.

2. Service Suggestion Engine

The Service Suggestion Engine facilitates the process of constructing and extending workflows by providing suggestions to the user for the next Web service operation. Suggestions are provided as a ranked list of Web service operations. The implementation of the algorithms expands upon our previous work [30]. It considers the set of operations currently in the workflow (`workflowOps`), the set of operations available for use in the workflow (`candidateOps`), and a desired functionality (`desiredOp`) which can be either the URI for some concept in an ontology or a string representing some operation. The score for each of the candidate operations is the weighted sum of their data mediation (dm) and functionality (fn) sub-scores, as seen in equation 1. The data mediation sub-score is intended to measure how well the inputs to a Web service operation can be provided by preceding Web service operations in the workflow either directly or through some form of data mediation (e.g., based on SAWSDL schema mapping specifications). The functionality sub-score is determined by how well a Web service operation matches the functional category or objective indicated by the user (e.g., Multiple Sequence Alignment).

$$S = w_1 \cdot S_{dm} + w_2 \cdot S_{fn} \tag{1}$$

In equation (1), the weights w_1 and w_2 will always sum to one. If no desired functionality is provided then $w_2 = 0$. Currently, these weights are set manually to be equally weighted, however, they can be optimized by machine learning algorithms. The data mediation and functionality sub-scores are detailed in the following two sub-sections.

2.1. Data Mediation Sub-score

The suggestion algorithm tries to find matches between the inputs and outputs of various Web service operations. The data mediation sub-score S_{dm} is the result of comparing the Input Output Directed Acyclic Graph (IODAG) representing the input of the candidate operation with IODAGs representing the outputs of selected operations in the workflow [30].

This comparison involves checking both the syntactic and semantic similarity between respective nodes of the IODAG, termed as concept similarity (CS). S_{dm} is a sum of these comparison sub-scores (CS), weighted as a geometric series starting with minimum weight for the root node. This method is a Path-based data mediation approach.

Concept similarity CS, as seen in equation 2, considers syntactic, coverage and property similarity. Syntactic similarity involves comparison of the labels and definitions associated with each of the two concepts. Coverage similarity indicates how the concepts are related to each other from their relative positions in the ontology. Property similarity measures the similarity between the properties of the concepts being compared.

$$CS = w_3 \cdot Syntactic_{sim} + w_4 \cdot Coverage_{sim} + w_5 \cdot Property_{sim} \tag{2}$$

ConceptSimilarity is developed as an independent component so that it can be used by other algorithms to facilitate Semantic Web service discovery [27].

2.2. Functionality Sub-score

A functionality sub-score S_{fn} is calculated when a desired functionality (functionality the operation is expected to provide) is provided by the user as the next step they would like to perform. A user can provide information as a simple string of text describing the desired operation or the user can choose from a list of concepts that denote objective specifications. If the desired functionality is provided in the form of a string then S_{fn} is based on the string metric results between the string and the labels associated with the concepts denoting their candidate operations. In this case, we use the Levenshtein distance [6] as the metric to calculate the difference between the two. If the desired operation is provided in the form of a concept URI, then the functionality sub-score is based on the concept similarity (CS) score between the concepts denoting the `desiredOp` and the `candidateOp`.

2.3. Understanding Properties and Restrictions

An ontology, at its core, is a collection of concepts and the relationships between them. These relationships are modeled by properties and restrictions upon them. Hence, property similarity is an important part of concept similarity. A property restriction is a special kind of class description that describes an anonymous class, namely a class of all individuals that satisfy the restriction [8]. Restrictions impose constraints on the range of the property (value constraints) or the number of values the property can take (cardinality constraints). For example, if there is a value restriction on a property (`owl:allValuesFrom`), then the range of the property would be changed to what is specified by the restriction. Hence, if this information is not captured, it would lead to properties being scored incorrectly. Both value and cardinality constraints must be taken into account when calculating the property similarity score for two concepts.

Let P_{C_1} and P_{C_2} be the set of properties that concept 1 and concept 2 participate in, respectively. We calculate Matrix P given by

$$\mathrm{P} = [prop_{ij}]_{\{i=1..m,j=1..n\}} \ \ where\ m = |P_{C_1}| \text{ and } n = |P_{C_2}| \tag{3}$$

[8] `http://www.w3.org/TR/owl-ref/#Restriction`

The value for $prop_{ij}$ is the property match score between properties i and j as given by equation 4. The $Syntactic_{sim}$ between the properties is computed using a string metric that compares the labels and definitions of two properties. When calculating the $Range_{sim}$, the algorithm checks for value restrictions that may exist on the property for the concept under consideration. If an `owl:allValuesFrom` or `owl:someValuesFrom` restriction exists, ranges are updated accordingly. The $Cardinality_{sim}$ takes care of cardinality restrictions.

$$prop_{ij} = w_6 \cdot Syntactic_{sim} + w_7 \cdot Range_{sim} + w_8 \cdot Cardinality_{sim} \tag{4}$$

The matrix stores property match scores between every two properties that the concepts in consideration participate. An optimal assignment between the two sets of properties is found by the Hungarian algorithm [17] which gives the final property similarity score, $Property_{sim}$.

2.4. Improved Use of Semantics

The Service Suggestion Engine makes use of semantics available from ontological annotations to facilitate the construction and invocation of a workflow. The problem is not completely solved with just Suggesting the next operation, as the user also needs to know which output of the previous operation to connect to which input of the subsequent operation. As in Figure 1, each operation will typically have multiple inputs and outputs, with some having as many as fifteen inputs. The SSE can help the user with this. For instance, in Figure 1, the SSE indicates that `exp` can be connected to `e-Val` and `Sequenceid` to `sId`. The Service Suggestion Engine achieves this by keeping track of the highest scoring matched paths in the IODAG when calculating the data mediation score S_{dm}.

At each step in the workflow construction process, the algorithm also checks if the inputs to the newly added operation can be fed by the outputs of the preceding operation. To achieve this, the algorithm also compares IODAGs of the outputs of all previous operations with the IODAG of the input of the newly added operation. At each step all the inputs that cannot be fed from any of the previous outputs are tentatively categorized as Global inputs. For each identified Global input, the Suggestion Engine assists the user in two ways. First, it tries to suggest possible input values that can selected by the user. To accomplish this, the SSE makes use of the ontological structure and determines if the annotated concept has any direct sub-classes or individuals. Consider an actual scenario, the BLAST Web service has an input, 'blast program' which allows the user to select the type of BLAST program to execute. In such a case, the algorithm can provide suggestions such as blastp, blastn, blastx using concepts obtained from the ontology. Second, if the algorithm cannot provide any suggestions for possible values, it may still facilitate user comprehension of the parameter using the definitions included in the annotation properties of the ontology and the documentation included in the WSDL file.

Another example of the Suggestion Engine making best use of available semantics can be found in the WUBLAST and ClustalW Web Services. The suggestion engine knows that the `run` operation of `WUBLAST` can take only one sequence as input, while the `run` operation of `ClustalW` needs at least two sequences in order to perform a multiple sequence alignment. The Suggestion Engine is able to supply this information to the user. It makes use of the cardinality restrictions defined in the ontology as well as information specified in the WSDL document.

3. Interaction with Galaxy

Galaxy is a Web application, developed and maintained by researchers at The Institute for CyberScience at Penn State and Emory University. It was designed to facilitate data integration and the construction and execution of bioinformatic and biomedical workflows. As discussed in the introduction, it comes bundled with its own set of tools for use in the construction of such workflows, however, these tools are generally required to be installed locally as programs on the machine hosting the Galaxy application.

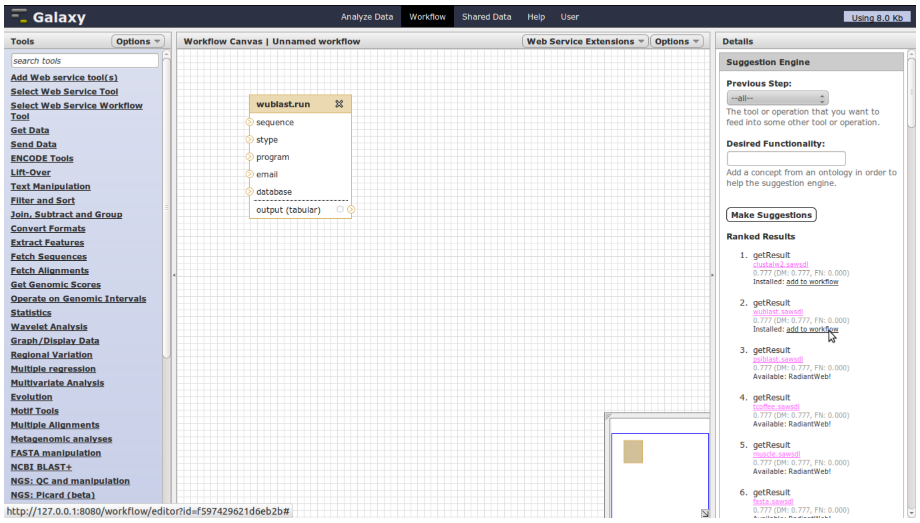
To facilitate the use of tools and resources located externally in the form of Web services, we created an extension to Galaxy that enables Web service operations to be added as tools in the Galaxy workflow editor. A user simply provides the URI to the desired, annotated, WSDL/WADL file and selects some or all of the operations that he or she desires to add. Additionally, we created another extension to Galaxy that extends the workflow editor to facilitate service suggestions via interaction with the Service Suggestion Engine. These tools were integrated into Galaxy using the following technologies: Java, JavaScript, JSON, JSONP, jQuery, Mako and Python. The Suggestion Engine described in this paper is hosted as a JSONP Web service so that it can easily be used by Galaxy and other tools.

To help make Web Service Composition easier, an interface addition to Galaxy that makes the Suggestion Engine Web service available inside the Galaxy workflow editor is provided. Users can request suggestions for Web service operations to be added after, before, or in the middle of the current workflow process, referred to as forward, backward and bi-directional suggestions, respectively[31]. By using the results provided by the Suggestion Engine, a human designer can easily cope with the input/output details of workflow composition and design.

3.1. Common Workflow

To illustrate the utility of semantically annotated Web services, a common workflow scenario is considered. A frequent use case encountered by biologists is that of discovering more information about a particular protein sequence and its evolutionary relationship to other protein sequences. Biologists often utilize several resources in a particular order to accomplish this task and thus it is an ideal candidate for a workflow. Web services already exist for each of the required resources and we utilize semantically annotated versions of each for our example. The input to the workflow is a user-supplied protein sequence. The workflow utilizes three popular bioinformatics programs: BLAST [1] for database searching and pair-wise sequence alignment, ClustalW [18] to perform multiple sequence alignment and Phylip [8] to construct phylogenetic trees.

Additionally, a few other Web services that perform format conversions, data retrieval, etc. are required. For the purposes of this paper, it is assumed that the required Web services have already been annotated and added as tools in Galaxy. The process of creating this workflow is described below.
The process begins with addition of the `run` operation of the `WUBLAST` Web service, which is annotated with objective specification “pairwise sequence alignment objective” from the OBI ontology. In order to determine the next step in the workflow, a Galaxy user needs to invoke the Suggestion Engine on the current workflow. This is accomplished by clicking on the “Web Service Tools” button provided at the top of the workflow editor and



Previous Step: --all--

The tool or operation that you want to feed into some other tool or operation.

Desired Functionality:

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

Ranked Results

1. getResult
   clustalw2.sawsdl
   0.777 (DM: 0.777, FN: 0.000)
   Installed: add to workflow
2. getResult
   wublast.sawsdl
   0.777 (DM: 0.777, FN: 0.000)
   Installed: add to workflow
3. getResult
   psiblast.sawsdl
   0.777 (DM: 0.777, FN: 0.000)
   Available: RadiantWeb!
4. getResult
   tcoffee.sawsdl
   0.777 (DM: 0.777, FN: 0.000)
   Available: RadiantWeb!
5. getResult
   muscle.sawsdl
   0.777 (DM: 0.777, FN: 0.000)
   Available: RadiantWeb!
6. getResult
   fasta.sawsdl
   0.777 (DM: 0.777, FN: 0.000)

Figure 2. Workflow Creation 1

selecting the Suggestion Engine from the list. Here, the user selects the `run` operation of `WUBLAST` as the previous step and clicks the "Make Suggestions" button. The Suggestion Engine returns a ranked list of possible services that can follow the previous step in the workflow. At the top of this list is the `getResults` operation provided by the `WU-BLAST` Web service. As this is the desired next step in the workflow, the user is able to click on the "Add to Workflow" link that is provided in order to place the operation onto the workflow canvas as a tool.

To complete the next step in the workflow, the user invokes the Suggestion Engine again, this time selecting `getResults` as the previous step in the workflow. This time, the ranked list that is returned includes the `filterByEvalScore` operation provided by `WSFilterSequences`. This particular operation filters the sequences returned by WUBLAST depending upon their e-value and score, which helps the user in narrowing down the number of sequences of interest before performing multiple sequence alignment. Once the sequences have been filtered, the user can invoke the Suggestion Engine again. This time, in addition to selecting the previous step, the desired functionality for the next step "multiple sequence alignment" is also specified. The ranked list of possible operations for the next step in the workflow includes the `run` operations of `ClustalW2`, `muscle` and `tCoffee` which are all multiple sequence alignments programs. As the `run` operation of `ClustalW2` appears at the top of the list and fulfills the desired functionality, the user can add it as the next step in the workflow. Just as with the previous case, the `getResults` operation provided by the `ClustalW2` Web service is suggested by the Suggestion Engine and added to the workflow by the user.

In order to achieve the final goal, a phylogenetic analysis (in this case utilizing a distance approach to phylogenetic estimation) the user needs to generate a distance matrix based on the multiple sequence alignment produced in the previous step. This time, with the user specifying the desired functionality as "protein distance matrix", the SSE returns the `protDist` operation offered by the `WSPhylip` Web service. Once this operation is added to the workflow the user wants to perform the last step of creating a distance-based tree using Phylip's Neighbor program. This time the user specifies the desired functionality as “phylip neighbor” and finds the operation `neighbor` from `WSPhylip`, which is then added to the workflow. The completed workflow is illustrated in figure 3

It should be noted that, throughout the process of workflow creation, the user could have specified the desired functionality using appropriate concepts from the ontology. The results of the rankings produced by the Suggestion Engine after each of the steps in the workflow is analyzed in the Evaluation section.

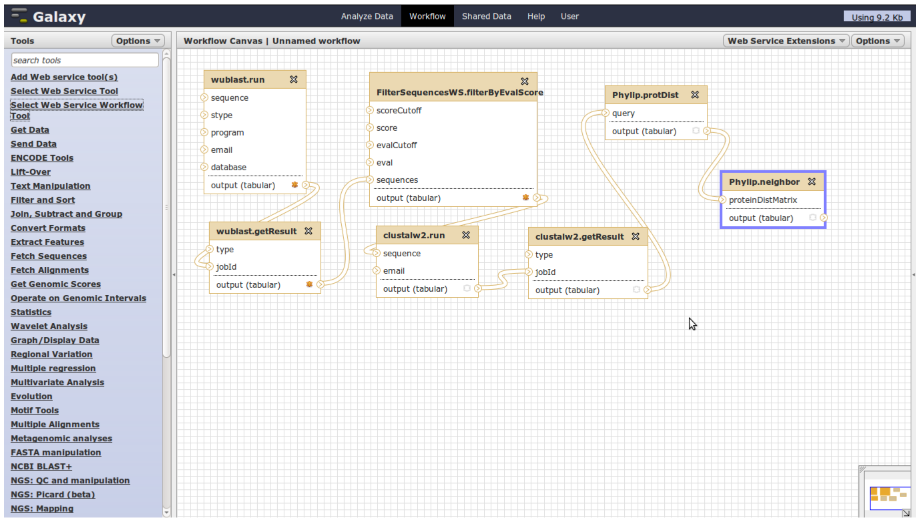


Figure 3. Completed Workflow

4. Evaluation

We have focussed our evaluation on a performance comparison of the Suggestion Engine relative to a consensus on ranking by Human Experts. The evaluation setup is comprised of 60 Web service operations from the following 11 Web services: `WUBLAST`, `NCBIBlast`, `PSIBlast`, `ClustalW2`, `TCoffee`, `WSDBFetch`, `WSConverter`, `Fasta`, `Muscle`, `WSFilterSequences` and `WSPhylip`. We have used the enriched Ontology for Biomedical Investigations (OBI), from our previous efforts [12] to annotate the Web services. All of the SAWSDL files and ontology used in this evaluation can be downloaded from `http://mango.ctegd.uga.edu/jkissingLab/SWS/Wsannotation/sawsdls.html`. A common bioinformatic workflow involving 7 steps (described above) is used as the basis for the evaluation. For the purpose of evaluation the algorithm in addition to returning a ranked list of operations, classifies them as high or low. The human consensus of the operations is also categorized as high or low. Since the operations classified as high are the ones that can ideally follow the current operation in the workflow, the performance of the Service Suggestion Engine is measured using precision and recall between the two sets. An ideal match indicates a reasonable choice for the next operation in order to advance the design a step further (although the workflow from section 3.1 specifies a unique operation for each step, the evaluation considers all reasonable next operations).

Precision (Eq: 5) and recall (Eq: 6) are a commonly used measure of quality of retrieved results (in our case the results are service operations) [20]. Precision ($\mathscr{P}$) is the fraction of retrieved results that are relevant, recall ($\mathscr{R}$) is the fraction of relevant results that are retrieved, and F-measure ($\mathscr{F}$) is the harmonic mean of precision and recall.

$$\mathscr{P} = \frac{(RelevantResults) \cap (RetrievedResults)}{RetrievedResults} \tag{5}$$

$$\mathscr{R} = \frac{(RelevantResults) \cap (RetrievedResults)}{RelevantResults} \tag{6}$$

$$\mathscr{F} = 2 \cdot \frac{\mathscr{P} \cdot \mathscr{R}}{\mathscr{P} + \mathscr{P}} \tag{7}$$

Figure 4 shows the precision and recall values for all the steps for two different cases. Case 1 represents Web services with no annotations and case 2 represents Web services with annotations on the input and output messages only. When no Annotations are present only a syntactic match between the inputs and outputs is considered and hence success depends only on the consistency in naming conventions between different Web services, which is rarely the case. Working with Web services with no annotations gives an average precision of 0.33 and recall of 0.33. With annotations on input and output messages only we got an average precision and recall of 0.62 and 0.87, respectively.

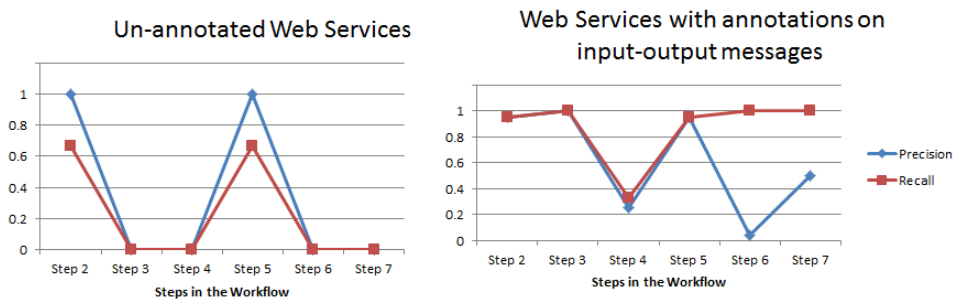


Figure 4. Precision and Recall for Un-annotated Web services and Web services with annotations on input and output messages

Figure 5 depicts the precision and recall values for Web services with annotations on input-output messages as well as the functionality of operations. Annotations with respect to the functionality of operations are important when doing semi-automatic workflow composition. This enrichment combines the input-output matching with the user's knowledge concerning the type of functionality that is desired. The graph on the left represents results obtained when the user supplies desired functionality as text giving precision and recall values as 0.64 and 0.98, respectively. The one on the right represents results obtained when the functionality is supplied as a concept in the ontology, giving 0.69 precision and 0.98 recall.

Figure 6 is a plot of average F-measure for all the steps for Web services with different levels of annotation. F-measure being the harmonic mean of precision and recall is a measure of overall effectiveness of the results.

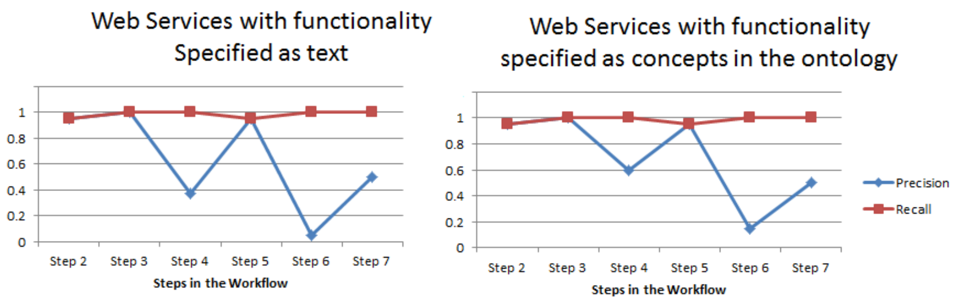


Figure 5. Precision and Recall for Web Services with Annotations on Input-Output Messages and Functionality

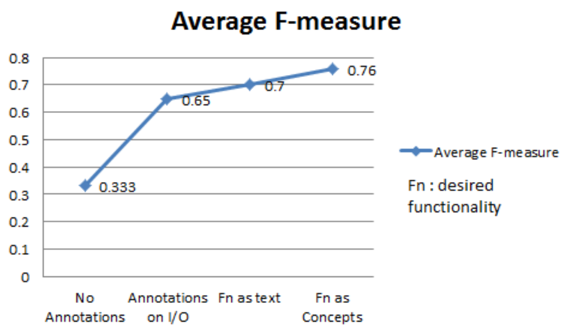


Figure 6. F-measure for different levels of annotation

When going from Un-annotated Web services to the Web services with annotations on input-output messages we observe an increase in F-measure of 0.32 (from 0.33 to 0.65), almost a 100% improvement. Adding annotations on the functionality of operations, shows a steady improvement in the F-measure. When the user supplies desired functionality as text the F-measure obtained is 0.70 and with functionality provided as a concept in the ontology, the F-measure is 0.76.

5. Related Work

In recent years, work has been done to advance the area of service composition, especially Web service composition (WSC). As early as 2002, McIlraith et al. [21] proposed the use of planning techniques for automatic composition of semantic Web services, however, this approach did not work well when workflow designers wished to influence the configuration of services during the composition process. In 2004, Kim et al. [15,16] proposed a Composition Analysis Tool (CAT) that provides feedback to composition designers as to whether a particular composition can be executed. As this work did not include user evaluations, statistical comparisons to compositions designed by domain experts were not performed.

In 2005, a survey by Rao et al. [22] details the three main approaches to WSC: manual, semi-automatic and fully-automated composition. Using a combination of these three techniques, an architecture for WSC that provides these different levels of automation was proposed. Later work by Charif-Djebbar et al. [4] proposed to further automate the service composition process using unplanned service-based dialogs among the agents who provide Web services. Cheng et al. [5] integrated Case-Based Reasoning (CBR), the process of solving new problems based on previous case studies, to compose Web services in a similar way. In all three papers, cases were made against fully-automated WSC due to the many complexities found in Web service environments. The work of Hull et al. [13] suggests that a semi-automatic composition process is preferable for service composition in general.

Schaffner et al. [24,25,23] also integrated a form of CBR in their study, proposing a semi-automated service composition approach based on mixed initiative features derived from an industrial case study. These features include filtering inappropriate services, checking composition validity, and suggesting partial plans. Pre-conditions and effects are used from the Web Service Modeling Ontology (WSMO)[9]. Evaluations were done on compositions for Business Process Management (BPM) scenarios using only composition validity as a metric. No statistical comparisons to compositions created by domain experts were provided (e.g., Precision and Recall). In 2008, DiBernardo et al. [7] proposed a composition client that ranks services in order to provide suggestions, however, the degree to which this aids designers in the WSC process is unknown as no comparison evaluations were made here either.

More recently, other frameworks for semi-automatic WSC have been proposed. In 2010, Khattak et al. [14] proposed a framework that operates mostly at the service level, however, due to the absence of an evaluation or actual implementation it is unclear how useful the suggestions provided by this approach are during the actual composition process. The work of Lécué [19] is similar to Schaffner et al. in that suggestions are provided primarily by first filtering out the services which cannot be composed. In later work by Canturk et al. [3], a similar approach is proposed that focuses on the discovery of Web services so that the WSC process can be started. An approach similar to the one presented in this paper is proposed by Schonfisch et al [26] that focuses on approximate subsumption of inputs/outputs in order to provide suggestions during the WSC process.

6. Conclusions and Future Work

This study was focussed on improving and extending the Service Suggestion Engine, based on our previous design effort for assisting users with workflow construction. Re-engineered code, improvements in calculation of metrics and consideration of restriction on concepts when calculating Concept similarity have yielded substantially improved performance. We have gone a step further in making use of available semantics, to help users find the next operation and build the workflow. This includes help with appropriate connection of outputs and inputs, providing lists of possible values the inputs can take and providing documentation for the input-output parameters that will help the user better understand and run the workflow.

[9]Web Service Modeling Ontology (WSMO): `http://www.w3.org/Submission/WSMO/`

Furthermore, work was performed on interfacing the Service Suggestion Engine with the Galaxy Tool Integration and Workflow Platform enabling invocation of SSE as a Web service from the Galaxy workflow editor. This also required enhancements to Galaxy's user interface to facilitate the user dialog. These enhancements facilitate the use of semantically annotated Web services in user-friendly format inside a Galaxy front end that provides a canvas for drag and drop construction of Web service operations into a workflow.

We are considering use of Schema Mapping [10](lifing and lowering), to further enhance data mediation between Web services using specified mappings. We would also like to reconsider the impact of adding pre-conditions and effects[31], since we expect that they will further increase the precision (pre-conditions and effect were dropped in the latest version for the sake of simplicity). Once we have considered schema mappings and pre-conditions & effects we plan to do an extensive evaluation that includes query times in addition to precision and recall for multiple workflow scenarios and additional Web services.

Acknowledgements

Funding for this study was provided in part by NIH R01 GM093132.

References

[1] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[2] D. Blankenberg, G. Von Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor. Galaxy: A Web-Based Genome Analysis Tool for Experimentalists. *Current Protocols in Molecular Biology*, 19(19.10):11–19, 2010.

[3] D. Canturk and P. Senkul. Using Semantic Information for Distributed Web Service Discovery. *International Journal of Web Science*, 1(1):21–35, 2011.

[4] Y. Charif-Djebbar and N. Sabouret. Dynamic Web Service Selection and Composition: An Approach Based on Agent Dialogues. *Proceedings of the 2006 International Conference on Service-Oriented Computing*, pages 515–521, 2006.

[5] R. Cheng, S. Su, F. Yang, and Y. Li. Using Case-based Reasoning to Support Web Service Composition. *Computational Science–ICCS 2006*, pages 87–94, 2006.

[6] F.J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[7] M. DiBernardo, R. Pottinger, and M. Wilkinson. Semi-Automatic Web Service Composition for the Life Sciences using the BioMoby Semantic Web Framework. *Journal of Biomedical Informatics*, 41(5):837–847, 2008.

[8] J. Felsenstein. *PHYLIP (Phylogeny Inference Package), version 3.5 c.* Joseph Felsenstein., 1993.

[9] R.T. Fielding and R.N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[10] C.A. Goble, K. Belhajjame, F. Tanoh, J. Bhagat, K. Wolstencroft, R. Stevens, E. Nzuobontane, H. McWilliam, T. Laurent, and R. Lopez. BioCatalogue: A Curated Web Service Registry for the Life Science Community. April 2009.

[11] T.R. Gruber et al. Toward Principles for the Design of Ontologies used for Knowledge Sharing. *International Journal of Human Computer Studies*, 43(5):907–928, 1995.

[10] `http://www.w3.org/2002/ws/sawsdl/spec/#schemaMapping`

[12] C. Guttula, A. Dhamanaskar, R. Wang, J.A. Miller, J.C. Kissinger, J. Zheng, and C.J. Stoeckert Jr. Enriching the Ontology for Biomedical Investigations (OBI) to Improve its Suitability for Web Service Annotations. pages 246–248, 2011.

[13] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-Services: A Look Behind the Curtain. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–14. ACM, 2003.

[14] A.M. Khattak, Z. Pervez, A.M.J. Sarkar, and Y.K. Lee. Service Level Semantic Interoperability. In *Proceedings of the 2010 International Conference on Applications and the Internet*, pages 387–390. IEEE, 2010.

[15] J. Kim, Y. Gil, and M. Spraragen. A Knowledge-based Approach to Interactive Workflow Composition. In *Proceedings of the 14th International Conference on Automatic Planning and Scheduling*, 2004.

[16] J. Kim, M. Spraragen, and Y. Gil. An Intelligent Assistant for Interactive Workflow Composition. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 125–131. ACM, 2004.

[17] H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[18] MA Larkin, G. Blackshields, NP Brown, R. Chenna, PA McGettigan, H. McWilliam, F. Valentin, IM Wallace, A. Wilm, R. Lopez, et al. ClustalW and ClustalX version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[19] F. Lécué. Combining Collaborative Filtering and Semantic Content-based Approaches to Recommend Web Services. In *The 2010 Internation Conference on Semantic Computing*, pages 200–205. IEEE, 2010.

[20] D.D. Lewis and W.A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 1994 Internationa Cconference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

[21] S. McIlraith and T.C. Son. Adapting Golog for Composition of Semantic Web Services. In *Proceedings of the 2002 International Conference on Principles of Knowledge Representation and Reasoning*, pages 482–496. Morgan Kaufmann Publishers; 1998, 2002.

[22] J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. *Semantic Web Services and Web Process Composition*, pages 43–54, 2005.

[23] J. Schaffner. Supporting the Modeling of Business Processes Using Semi-Automated Web Service Composition Techniques. Master's thesis, Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam, Potsdam, Germany, 2006.

[24] J. Schaffner, H. Meyer, and C. Tosun. A Semi-Automated Orchestration Tool for Service-Based Business Processes. *Service-Oriented Computing*, pages 50–61, 2007.

[25] J. Schaffner, H. Meyer, and M. Weske. A Formal Model for Mixed Initiative Service Composition. In *Proceedings of the 2007 International Conference on Services Computing*, pages 443–450. IEEE, 2007.

[26] J. Schönfisch12, W. Chen12, and H. Stuckenschmidt. A Purely Logic-Based Approach to Approximate Matching of Semantic Web Services. In *Proceedings of the 2009 International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web*, April 2009.

[27] A. Sheth, K. Verma, J. Miller, and P. Rajasekaran. Enhancing Web Service Descriptions using WSDL-S. *Research-Industry Technology Exchange at EclipseCon*, pages 1–2, March 2005.

[28] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, C.J. Mungall, et al. The OBO Foundry: Coordinated Evolution of Ontologies to Support Biomedical Data Integration. *Nature Biotechnology*, 25(11):1251–1255, 2007.

[29] R. Wang, D. Brewer, S. Shastri, S. Swayampakula, J.A. Miller, E.T. Kraemer, and J.C. Kissinger. Adapting the Galaxy Bioinformatics Tool to Support Semantic Web Service Composition. In *Proceedings of the 2009 World Conference on Services-I*, pages 283–290. IEEE, 2009.

[30] R. Wang, S. Ganjoo, J.A. Miller, and E.T. Kraemer. Ranking-Based Suggestion Algorithms for Semantic Web Service Composition. In *Services (SERVICES-1), 2010 6th World Congress on*, pages 606–613. IEEE, 2010.

[31] Wang, R. and Guttula, C. and Panahiazar, M. and Yousaf, H. and Miller, J.A. and Kraemer, E.T. and Kissinger, J.C. Web Service Composition using Service Suggestions. In *Proceedings of the 2011 IEEE World Congress on Services*, pages 482–489. IEEE, 2011.