ABSTRACT

In this paper, we recognize the shortcomings of the current search engines that do not index and search the Deep Web. We present requirements of a Deep Web Service search engine, that will lead to the query objects in the deep data sources. In order to realize the DWS search engine, we propose semantic metadata and annotation of Deep Web Services (DWS), a reasoning component to assess the relevance of DWS for searching the Deep Web contents, using likelihood of occurrence of data sources that contain the query terms, and present a method of ranking the DWSs. The Deep Web Service annotation considers not only the service descriptions like any Web services, but also has the frequency distribution, clustering and semantic prediction functions that may guide the search for DWSs.

Categories and Subject Descriptors
H.3 [Information Storage and Retrieval]: Search Process, Selection process, information Filtering.

General Terms
Search, Web Services, Access, Semantic Web

Keywords
Deep Web services, Semantic annotation, spatial and temporal annotation, WSML

1. INTRODUCTION

Businesses rely on data to generate meaningful information to make timely accurate decisions to stay competitive. Many business intelligence (BI) tools thus need to identify the appropriate data sources, extract the relevant data from the sources and integrate them to generate knowledge. With the Semantic Web, businesses also expect automated agents to identify the relevant Web sources for mining and searching its contents.

However, the current search engines do not crawl the Deep Web where the data is dynamically accessible through Web form interfaces or Web services. These Web interfaces and services are referred to as Deep Web Services (DWS).

Thus, there are needs to have a Deep Search Engine that may consider vast amounts of data often ignored by the search engines. The data contents may not be directly indexed for the Deep Web, since data are hidden and need to be extracted through iterative Web form interfaces. To overcome the difficulty of this direct indexing of the content, the Deep Web search engine may resort to two possible solutions. One is to sample the data contents returned from multiple sample queries of DWS [2]. The question for this approach is whether the sample data are actually representative of the data sources. How big does the sample need to be? Once the sample data are retrieved, how can they be used to help the search engine? The instances of the Deep data sources can be indexed, but other instances are not indexed. Another approach is to index the Deep Web services with annotations that may reflect the data source contents, and these annotations can be used by the search engine to discover appropriate DWSs that may contain the query terms. The Current Web services community uses UDDI or local directory services to discover appropriate Web services, and Web service search engines such as Woogle [6] use the UDDI to search Web services based on operations or input/output behaviors. The focus is to find a Web service, not the contents that a Web service may deliver.

We view DWSs as a gateway to the content in their data sources, and propose a semantic annotation of Deep Web Services that may reflect the content of the data source. These content-based semantic annotations allow a search engine to first locate the relevant DWSs. Once the content-relevant DWSs are identified, the search engine ranks these candidate DWSs according to the relevance to the query words, and presents the ranked list of DWSs as results. When one of these discovered DWSs is clicked the query word(s) may be used to retrieve the content of the data source.

The content of the Deep Web is represented with different coverage dimensions, such as thematic coverage dimension that shows the topic/thematic contents, geographic coverage dimension that shows the geographic distributions of the data items, and temporal coverage dimension that reflects the temporal distribution of the contents, as well as administrative dimension which states the content access, operation, and dissemination policies.

These content distribution dimensions are used to develop the weighted semantic annotation.

This paper is organized as follows: In section 2, we discuss our approaches for content-based annotation for DWSs using the sampled content instances as well as the information within DWS Web interfaces or Web service descriptions (WSDL). Section 3 shows how the weighted semantic annotations are used in search and ranking. In Section 4, we discuss related work, and in Section 5 we present our future activities to implement and show a design to conduct the performance analysis study, followed by conclusion.
2. SEMANTIC ANNOTATION OF DEEP WEB SERVICES

In the semantic annotation tool CREAM described in [1][3], the dynamic Web pages are annotated by the content provider (server side). Given this content provider’s annotation, the client can use its ontology to generate the mapping rules between the semantic class and the generic instances and attributes. The query processor makes use of the client ontology and the mapping rules to query the instances. However, as most of the annotation tools, it is heavily manual. The tool’s basic function is to drag and drop and manually match the instances on the Web page output to ontology classes, attributes or relations. Although it uses the notion of mapping a Web page instance to “Generic Instances” and stores the queries to represent a class of the instances, it is not clear whether the annotation tool actually helps the user to understand the underlying database concepts. If so, it requires more experimental studies to show the effectiveness of the approach in actual queries. Assuming a dynamic Web page is similar to a potentially generated Web page, this instance based annotation may well represent the content. However, it requires a cooperative content provider who is willing to annotate the content. In addition, the semantic annotation relies on ontology, but it is unclear whether it makes full use of the larger context information beyond the semantics.

The issue is how to enrich the Deep Web service descriptions to reflect the dynamic content. What quantities of the dynamic pages are needed in order to say that a deep web service sufficiently represents the contents in the Deep Web well enough? We agree that the contents of the dynamic Web pages need to be semantically annotated. In order to achieve this, instead of using a content provider’s manual annotations, we propose to use polling of many dynamic Web pages in order to assess the content coverage using different views (Web forms or Web services). Given the sampling of dynamic pages returned by the deep web services, we can build a distribution of the contents and confidence levels.

OWL-S [4] provides a semantic ontology for describing Web services for automatic discovery, invocation and execution. The profile tags provide means to publicize the service metadata such as name, provider contact information, textual description, service functionality information such as service category, service type and product, input, output, pre-condition and results. However, the Web forms used in the Deep Web are not described in this manner for automatic discovery and invocation. What is needed is to augment the annotation for the Web forms for the Deep Web data sources, not only with the profile, but also with some of the instance occurrences.

Figure 1 shows the process of semantic annotation of the Deep Web services. First, the crawler will sample the data sources N different times in the Deep Web source sampling step; then the query results are used for semantic clustering analysis, as well as frequency/distribution analysis of instances, and semantic projection analysis to predict the likelihood of the occurrence of an instance, with a Semantic Projection Function. The Semantic Projection Function maps the input vector \([x_1, x_2, ...]\) which can be a semantic class or instance information to the output vector \([y_1, y_2, ...]\) that indicates the output semantic class or instances. In the next stage, the mapping function, between classes and instances of input vectors and output vectors are translated into RDF tuples using an ontology. These ontology enriched tags also include the spatial and temporal coverage of the data sources given the sampling results. The clustering analysis will provide the semantic centrality of the sampled Deep data sources. The frequency information, Semantic projection functions, and the semantic RDF relationships are recorded in the annotation file that is linked to the Deep Web Services to be used in Deep Web search.

Figure 1 Semantic Annotation Process for Deep Web Services
The results of this semantic annotation of Deep Web Services will contain not only the similar service description information as in WSDL, but also the contents it may find through the service, which we call the DWS content extent, that contains the thematic extent information such as content words distribution (frequency of content words), as well as the key concepts or instances that can be found through this service. The annotation also contains spatial and temporal distribution of the contents. For example, the Deep Web service contains 70% of sampled data covering from 1960 to 2000, in North East American cities, and 10% of data in a different period, and a different spatial coverage.

An example of data sampled with the DWS may contain the following instances and their frequency information:

Example 1: Austrian Air 8; British Air 10; Lufthansa 15; Czech Air 4;

From the distribution of the data content, the semantic similarity of this data can be extracted; thus, the class/concept “Airline” is the data content, with these instances of airlines, and from the geographic coverage of these flights, the Deep Web content contains the geographic coverage of Europe. This semantic commonality information (i.e. concept of “airline”) and its instances with the frequency information, and the geographic coverage information all can be stored in the annotation document. From this, one can also predict that a likelihood of finding a data item “Al Italia” is high, while “Japan Air” may occur with low probability. The rich annotation of the DWS as shown in this example will allow reasoning about the Deep Web data contents. The annotation is used for indexing the DWSs for search. The following section describes our approach to searching Deep Web services.

3. SEARCHING FOR DEEP WEB SERVICES

As indicated in [12], a critical problem for enterprises is finding services that can be reused across entire business processes. Use of a functional description of the service with a discovery mechanism that can leverage the description is considered the key to increasing reuse in enterprises. However, most of service related research does not consider the Deep Web source contents in its search mechanism. The semantic annotations and semantic tags are used for describing the service operations, parameters, and other metadata. The semantics considered include functional semantics, semantics of exchanged data, non-functional semantics such as service level agreements or quality of service attributes, and execution semantics of run time behaviors.

Thus, our approach is similar to this kind of work on Web service search in leveraging the semantic annotation, however, it does not limit the semantics in these categories only, such as functional and parameter annotations, but it includes richer content-related annotations for the Deep Web services. Thus, the Deep Web search engine should look for the services that match the semantics of those four types, but also match the semantic annotation of the contents, content distributions such as frequency information, centrality information, and spatial and temporal coverage information, sampled and summarized to represent the Deep Web source. Thus, our proposed search engine, called the Semantic Deep Web Service search engine, is geared towards identifying the Deep Web services that may provide the contents of the search interest, rather than its functional parameter matching or behaviors.

The current Web service discovery is done using UDDI which is a directory of published Web Service profiles. This UDDI may be private, used among collaborating partners in a business to discover and to bind to their actual implementations of the services. In this environment, a search/discovery engine can be developed to go through the Web Services description in a UDDI and find the Web services relevant to the user’s request. The Wooogle search engine [6] is a specialized Web Service search engine that finds Web services similar to an existing Web service that matched a user's interest. It measures the similarity on the input/output parameters, operations as well as composability of Web services. However, this search engine utilizes the structural information of Web services, and does not address the issue of dynamic data or contents the Web services provide.

Swoogle [8][9] is a Semantic Web document search system that discovers the Semantic Web documents and indexes them for search and retrieval. The Semantic documents are defined as documents written in Semantic Web languages such as RDFs, OWL or DAML+OIL, which contain semantic ontologies that define terms and extend classes and semantic databases that assert the instances and their relations. These documents are considered RDF graphs with nodes. Each node is classified as a class, property or instance. In addition, the semantic annotation on labels, comments or version information is also recorded. The system also captures the relationships existing among Semantic Web documents to be used for search. These include imports(A,B), i.e. A imports all content of B; uses-term(A,B), i.e. A uses some terms defined by B without importing B; extends(A,B), i.e. A extends the definitions of terms defined by B; and asserts(A,B), i.e. A makes assertions about the individuals defined by B. This is a specialized system to search the Web semantic documents that have special structure and have semantics, as opposed to other keyword search engines that concentrate on the unstructured Web documents.

3.1 Semantic Annotation Crawler for Deep Web Services

At a high level when searching for content in the Deep Web, one could narrow down the Web Services that might apply by examining the Deep Web service annotation (written in modified OWL-S [7] or WSML [8] or SAWSDL[11][12]). In order for this kind of semantic annotation search, a semantic annotation crawler for Deep Web service is used. It collects the DWS annotation files and indexes for searching the related DWSs.

The data collected by a crawler should be organized for fast identification of search words or concepts in the Deep Web as well as for identification of Deep Web services that meet certain criteria (e.g. functional and non-functional criteria.)

We define the annotation index with different types of tuples:

i. The sampled source contents are represented with the semantic information: \(<c, URI(O), URI(S), f>\) where c is the instance, C is the concept/class information of the instance c, URI(O) is the URI link to the ontology used, URI(S) is the URI link to the Deep Web service for the semantic annotation, and f is the frequency of c. For all
concepts $c \in C$, we can define a concept frequency distribution $F(c)$ over a sample size.

ii. The RDF relationships among contents are represented as $<\text{rdf}, \text{URI(O)}, \text{URI(S)}, f>$ where $\text{rdf}$ is a triple (subject, property, object) to represent a relationship or assertion between instances in the Deep Web source data. Similar to $F(c)$, we can construct a frequency distribution $F(\text{rdf})$ for all sampled $\text{rdf}$’s using the frequency counts of individual $\text{rdf}$’s.

iii. The contextual information of the Source contents are represented with: $<\text{cn}, \text{ext}, \text{URI(O)}, \text{URI(S)}, f>$ where $\text{cn}$ is the spatial or temporal terms, $\text{ext}$ is the spatial coverage area or temporal duration, $\text{URI(O)}$ and $\text{URI(S)}$ are the links to the ontology and to the Deep Web Service, and $f$ is the probability of the contextual coverage by the deep source data. The contextual frequency distribution can be constructed based on the contextual frequency.

The semantic crawler also stores the information for the Deep Web services. In order to facilitate the search, the descriptors adopted in WSDL as well as WSML used for describing Web services are used in indexing:

- hasFunctionalProperties
- hasNonFunctionalProperties
- importsOntology
- hasCapability
- hasInterface
- hasPrecondition
- hasTriggerEvents

The challenge lies in how to identify and extract information from the Deep Web Services. Unlike the Web services where the standard language WSDL and the protocol of sharing the Web services through publishing in UDDI are readily available, the Deep Web services are not provided with these descriptors. Thus, the descriptors either should be specified by the providers, or should be semi-automatically extracted from the Web forms.

### 3.2 Search Engine for Deep Web Services

For this purposes, the Semantic Deep Web Search engine uses the following search process as shown in figure 2. It uses two stage search. First the semantic annotation of the Deep Web services is used to identify the Deep Web content related services. The identified services are further filtered using the Service related functionalities, if needed. The ranked services are returned to the user to explore and enter the search terms for dynamic content retrieval.

As seen in figure 2, a user enters a search term through a search interface. The term(s) is analyzed with its semantics using an ontology. They are entered for semantic matching and reasoning with the semantic annotation index database. Then the appropriate relevant semantic annotations are retrieved and ranked which are indexed to the referenced Deep Web services. The Deep Web services linked from the selected annotation index will be further analyzed for its relevance. At this time, its relevance is measured with the functionalities, and non-functional attributes, such as Quality of services, and the ranking is recalculated based on both the semantic annotation search results and the Deep Web Service related matching results. The final ranked list of the Deep Web services is returned for the user to explore and formulate the Deep Source query using one of the DWSs.

### 3.3 Semantic Similarity Search

The semantic Deep Web search engine described above and shown in figure 2 implies that the search resorts to semantic analysis and semantic matching, different from the database or XML query languages, or the popular search engines, which rely heavily on more exact string matching, i.e. more linguistic and structural matching. The semantic similarity matching should consider the similarity between concepts conveyed in the search terms and those in the semantic annotation. Even though the exact label or instance may not match, the class concepts or the sibling concepts match, in which case we consider this as “Semantic hit.”

In order to determine the semantic hit or similarity, we use semantic operators such as semantic equivalence, semantic overlap, semantic generality, or specificity operators. These operators are derived from more primitive operators such as “equalTo”, “instanceOf”, “parentOf”, “siblingTo”, “childOf”, “synonymOf” and existing properties and relationships of two matching objects. For instance, semantic equivalence may be measured with either the two terms match linguistically or structurally, or one term is a synonym of another term. Semantic overlap may depend on the sibling or overlap of properties or relationships. The semantic similarity function also looks at the overlap, semantic generality, or specificity operators. These semantic operators such as semantic equivalence, semantic overlap, structural matching, and semantic generality, or specificity operators.

In order to determine the semantic hit or similarity, we use semantic operators such as semantic equivalence, semantic overlap, semantic generality, or specificity operators. These operators are derived from more primitive operators such as “equalTo”, “instanceOf”, “parentOf”, “siblingTo”, “childOf”, “synonymOf” and existing properties and relationships of two matching objects. For instance, semantic equivalence may be measured with either the two terms match linguistically or structurally, or one term is a synonym of another term. Semantic overlap may depend on the sibling or overlap of properties or relationships. The semantic similarity function also looks at the overlap, semantic generality, or specificity operators. These semantic operators such as semantic equivalence, semantic overlap, structural matching, and semantic generality, or specificity operators.

### 3.4 Ranking Deep Web Services

The relevance of the Deep Web services can be measured using the semantic similarity of the thematic, spatio-temporal distribution of the contents, and the similarity matching results of the Deep Web services. Different weights are assigned depending on the query terms and context. If the query is mostly on the Deep Web content, then the content-related semantic annotation matching will have a higher weight than others. If the query is mostly about the service descriptors, e.g. quality of services or
provider types, then more weight is given on the service descriptor-based matching.

For each DWS, we will calculate the following semantic relevance of DWS to each query term \( t \):

\[
SS(DWS, t) = \sum SS(c, t) f(c) + \sum SS rdf * f(rdf) + \sum SS(cn) * f(cn)
\]

where \( SS \) denotes semantic similarity of search term to the semantic content in the DWS annotation, \( SS(c, t) \), similarity to the relationships the term involves, \( SS(rdf) \) and the spatio-temporal context of DWS to the search term \( SS(cn) \), each of which are weighted with the frequency distribution of each dimension.

4. SEMANTIC DEEP WEB SERVICE QUERY

The queries can be based on general keywords with optional constraints on service descriptors, such as Web service functionalities, or non-functional characteristics. For example, suppose one submits a query on “personal financial planning”. With Google or another search engine, one gets results for books, tools, articles, services, consultants, course listings and many more objects. Some results might even include web pages that have semantic web services but it is not clear which they are.

Clearly, it would be better if one could better describe what one is looking for using characteristics of the web services themselves. Search with augmented WSML specifications of Deep Web services allows one to search for specific aspects of the description. Thus, a query has the following format:

\[
[\text{KEYWORD}]^* \ [\text{WHERE} \ S \text{-descriptor OP value}]^* \ | \\
[\text{WHERE} \ SA \text{-descriptor SOP value}]^*
\]

It contains one or more keywords with optional constraints. The restrictive WHERE clause contains expressions on the Deep Web service criteria. Specifically, the service descriptor (S-descriptor) can contain characteristics of the Deep Web services such as \text{hasNonFunctionalProperties}, \text{importsOntology}, \text{hasCapability} \text{ or hasInterface}, and the operation OP is a logical or comparison operator (e.g. =, >=, contains, substring, etc.), and the values are the possible values for each descriptor type’s domain.

In addition, the WHERE clause may contain a semantic annotation expression to restrict the semantic context of the keywords. These semantic constraints use semantic operation SOP as we discussed in the semantic similarity matching in Section 3.3 (e.g. semantic Overlap, parentOf, etc.) Semantic annotation descriptor and value pairs include the ontology and its URI, frequency, and other criteria as discussed in section 3.1 for a Semantic annotation crawler.

Following are some examples of the possible queries using different restriction clauses.

Non-functional property query: A query using non-functional properties includes information about who contributed the service, its coverage spatially, temporally and jurisdictionally, the creator of the service, description, format, subject, title and type. While full of information might help in search, there are two problems. First, non-functional properties are optional and second, access to non-functional properties in WSML is not part of the logical language. Thus a search of nonFunctionalProperties may not lead to retrieval of all relevant services. However, services whose creators want them to be found will, over time, include the properties and offer an API to access them.

A user’s intent in searching for “personal financial planning” is to get current advice and information from well known experts. The advice could be in the form of information, tools, or interactive chatting any of which one might specify as a types that should be found in the format attribute. In addition, the semantic thematic coverage should be matched with “personal financial planning”.

Alternatively one might be looking for the latest research on financial planning. For example, one might be interested in ensuring that the temporal period covered is within the last year and the media type is text. This can be achieved using restriction on the semantic annotation attribute of temporal coverage [22].

Imported Ontology Query: Like nonFunctionalProperties, ontologies might be imported as part of the webService class as well as part of many different attributes of the web service. The difficulty here is deciding which ontology might be relevant. For example, the “financial exchange framework” ontology could be searched for within the web service definition.

Web Service Capabilities query: The capabilities of a web service are declared and may contain descriptions of shared variables, preconditions, postconditions and effects. If a particular effect is desired, it can be used as part of the search. However, this might be too specific for a deep web search where many different items might be desirable. Instead, the web service capabilities descriptions are mainly analyzed to determine whether a one stage or a two stage search might be workable.

For an implementation of a prototype, we consider existing RDF query language SPARQL [14] which is considered a query language for XML-based RDFS. Since the semantic annotation and the Deep Web service descriptions are using RDFS and augmented semantic standard languages, the query language SPARQL can be adopted for rapid prototyping. The query: to search services that return “airlines with destination city in countries in Europe” This query may be translated into the following pseudo-SPARQL statement:

\[
\text{PREFIX abc: <http://.../AirlineOntology#> .} \\
\text{SELECT ?airlines ?city ?country} \\
\text{WHERE {} } \\
\text{ ?x abc:cityname ?city ;} \\
\text{ abc:isInContinent abc:Europe} \\
\text{ abc:isDestinationOf ?y.} \\
\text{ ?y abc:airlinename ?airlines ;} \\
\]

4.1 Deep Web Service Search Interface

The query result would be a ranked list of web services according to the degree of similarity in semantic annotation of its sample content and their descriptions. Since the similarity search may not be exact matching and also the contents for matching are not complete, further exploration may be needed. To help this exploration, the thematic map with content distribution may be shown to the user to decide to invoke the service, or the Deep Web Service interface parameters may be displayed for the user.
to determine if the service can be used for composing with some other service. Our method to achieve this search result exploration is using the content clouds just like the tag clouds frequently used in Web 2.0 to visualize the tags generated in social networks [17, 15]. More frequently mentioned thematic content is shown with larger icon than less frequently occurring content words.

In addition, the exploration of the ranked list of returned Deep Web services can be visualized on the map, using the spatial coverage by the Deep Web source content, and also the results can be visualized according to the temporal distribution [16].

5. RELATED WORK

There are many efforts to describe the content of Web sites, including the aforementioned specification languages such as WSDL, OWL-S, WSML, or WSMO, SAWSDL. For example, the Internet Content Description Language (ICDL) [13] is a language to describe the content of a Web site such the Web crawler can easily identify the site content by a web crawler. It can list the ontology concepts, HTML structure or content organization templates, and URLs used in the website. ICDL can be used to describe the concepts used in the Deep Web pages, but it does not include the metadata proposed in this paper such as frequency or spatial/temporal coverage and the sampling information. These specification languages should be extended in order to describe the Deep source content information and the Deep Web service.

In [6], a similarity search for Web services is proposed. The similarity of two Web services is based on similarity of several dimensions, namely, the textual description similarity, Input/output parameter similarity, operations similarity, as well as similarity on composable operations. They employed the clustering of semantics of parameters of Web services to achieve the conceptual similarity (called conceptual cohesion), instead of limiting it to the linguistic similarity. Based on these clusters of concepts the Web services can be measured for similarity in their Input/Output parameters or operations. However, the work is strictly limited to the Web service similarity, rather than the search of the data sources that can be provided by the Web Services. Its main concern is not about searching the data that can be dynamically generated from the Deep Web. This work also does not deal with the Deep Web Services (also called Source Query Interfaces [2]).

The Semantic Web search engine called Swoogle [8] extracts metadata for discovered documents, and computes relations between documents. The information retrieval system indexes these discovered documents and uses either N-Gram or URLs as keywords to find relevant documents and to compute the similarity among a set of documents.

In [18], the Deep Web content extraction wrapper is generated from the Web form results to reconstruct the table and the table column (attributes) names are labeled using heuristics of recognizing the form element labels and result labels for data elements. However, it does not address the semantic issues.

The multi-annotation approach in [19] makes use of different information to label web data elements, such as table label, query terms, frequency of terms, search interface schema values, in-text prefix, and common sense annotators. The annotation wrapper is generated from these. These multiple annotators can be weighted and aggregated, and it utilizes not just the HTML or XML tree structures as in DeLa [18]. Other related studies provide approaches on query translations for integrated search for the Web Data Interfaces [20, 21]. These methods use the similarity matching of attributes of the Deep Web Services (interfaces).

6. CONCLUSIONS AND FUTURE DIRECTION

Deep Web sources that are not searchable through the conventional crawlers. To address this challenge, we proposed the semantic annotation of the Deep Web services based on the semantic content of the data source using sampling methods. Then the semantic Deep Web service crawler’s components are discussed. The semantic annotation and web service descriptor based search engine has been discussed. However, as mentioned above, there are many challenges to Deep Web access.

First, the Deep Web services which are mostly form-based interfaces need to be described using semantic specifications. The parameters or the service behaviors as well as non-functional properties need to be extracted or inferred. In addition, the sample of Deep Web source contents have to be collected by the semantic annotation crawler, and semantic representations of these contents have to be stored and indexed. If the sample is not representative of the Deep Web, the search engine will miss many relevant DWSs.

We need to represent content relationship information in the annotation of the DWSs. This requires extracting and sampling the data to see the relationships among instances.

The search engine should use a method to filter many semantically relevant services to identify most relevant ones.

Further investigation is necessary to determine whether SPARQL supports the non-exact similarity matching discussed in section 3.3, and whether the operator sets contain those proposed in Section 3.3 and section 4.

7. REFERENCES


