Journal of Universal Computer Science, vol. 17, no. 14 (2011), 2029-2047
submitted: 3/12/10, accepted: 28/9/11, appeared: 1/10/11 © *J.UCS*

Web Services Discovery in a Pay-As-You-Go Fashion

Ying Pan
(College of Computer and Information Engineering, Guangxi Teachers Education University, Nanning, P.R. China
panying6@mail2.sysu.edu.cn)

Yong Tang[1]
(School of Computer Science, South China Normal University, Guangzhou P.R. China
ytang@scnu.edu.cn)

Shu Li
(Dept. of Computer Science, Sun Yat-sen University, Guangzhou, P.R. China
mcsyls@mail2.sysu.edu.cn)

Abstract: Extensive effort has been brought forth to assist in web service discovery. In particular, classic Information Retrieval techniques are exploited to assess the similarity between two web services descriptions, while Semantic Web technologies are proposed to enhance semantic service descriptions. These approaches have greatly improved the quality and accuracy of service discovery. However, these works require hard up-front investment before offering powerful functionalities for service discovery, and they do not study how to discover web services in a pay-as-you-go fashion. In this paper, a framework based on dataspace techniques is proposed to discover web services in a pay-as-you-go fashion. In this framework, a loosely structured data model based on dataspace models is presented to describe web services and the relationships among them, and then keyword-based query is supported on top of this model by using the existing dataspace query language. To support similarity-based service discovery, dataspace techniques are extended to declare the similarity among web services, and a discovery algorithm is presented. In addition, a lightweight way adding semantics to the query processing is also shown in the paper. Finally, the differences between our work and previous works are discussed.

Key Words: Web service discovery, Dataspace, Web service similarity
Category: C.2.4, D.3.m, H.3.m

1 Introduction and motivation

Web services play an increasingly important role in Computer Supported Cooperative Work (CSCW), and one of the major challenges in web services research field is to discover proper services in an accurate and efficient way. In recent years, extensive effort has been brought forth to assist in web service discovery. Commonly, web services are described by WSDL and advertised in UDDI registries.

[1] Corresponding author

However, UDDI provides limited keyword-based search only on names and comments, as well as keys of businesses and services descriptions, which suggests that this search facility is not powerful enough. To address this problem, Semantic Web (SW) technologies (e.g., ontology) are proposed to enhance semantic service descriptions [Akkiraju et al. 2005, Martin et al. 2007] and support semantics-based service discovery [Bianchini et al. 2008, Liu et al. 2009, Klusch et al. 2009, Adamopoulou et al. 2007]. Ontologies allow users to share common knowledge among different resources, and make the integration of semantics information more easily. Utilizing the available ontologies and ontology reasoners, search engines can easily find the appropriate web services. At the same time, classic Information Retrieval (IR) techniques (e.g., similarity measures) are exploited to assess the similarity between two web services descriptions [Dong et al. 2004, Stroulia and Wang 2005, Wu and Wu 2005]. The basic ideas of these approaches are similar: the service descriptions and queries are converted into a common representation, and then the similarity functions are computed to find out those web services with the most similar representations. These approaches have greatly improved the accuracy of service discovery.

However, the previous works mentioned above support web services discovery in a pay-before-you-go fashion, that is, they require hard up-front investment before offering powerful functionalities to discover web services, and do not offer gradually enhanced functionalities according to the amount of effort investment. For example, before powerful functionalities can be provided, the user needs to describe web services into the strict data models (e.g., WSDL and OWL-S) which enforce a schema over the data. This up-front effort is high-cost, because it is a difficult work to require all data to be under the control of a single domain and to conform to a single schema. In addition, semantic-based approaches require full semantic integration of the data sources before the semantic query is provided, but it is difficult to understand the data and fully create these semantic mappings immediately.

Web services are advanced and adjusted over time, and thus the web service descriptions need to be changed accordingly. Therefore, the full, one-time integration technique is not feasible to manage such evolving data. The recent researches [Franklin et al. 2005, Salles 2008] show that heterogeneous and evolving data should be managed or integrated in an incrementally, pay-as-you-go fashion, that is, the management system firstly offers the simple services (e.g., search and query) on data without requiring the expensive up-front investment, and then gradually enhances services over time. We thus argue that web services management and discovery in a pay-as-you-go fashion is more suitable than that in a pay-before-you-go fashion. However, to the best of our knowledge, the previous works did not study how to discover web services in an incrementally, pay-as-you-go fashion.

Dataspace [Franklin et al. 2005], a new style of pay-as-you-go data management, addresses the challenges mentioned above. In this paper, we show how to discover web services in a pay-as-you-go fashion by extending and adapting the existing dataspace techniques. The main contributions of this paper can be summarized as follows:

1. We present a model Web Services Description Graph (WSDG) based on current dataspace data models. WSDG is a very loosely structured model, which represents web services and the relationships among them into a logical graph. Moreover, WSDG may be constructed in a pay-as-you-go, ongoing fashion.

2. We show the transformations between WSDG and other dataspace models, and support keyword-based discovery on top of WSDG.

3. We extend dataspace techniques to declare the similarity among web services. An algorithm for similarity-based service discovery on top of WSDG is also presented.

4. With an aim to improve the accuracy of query results, we add semantics to the query processing using dataspace techniques. This approach is much more lightweight, allowing users to add semantics over time in a pay-as-you-go fashion.

The remainder of this paper is organized as follows. The next section introduces the dataspace techniques which are relevant to our research. Section 3 discusses the framework and the key methods for web services discovery in a pay-as-you-go fashion. Section 4 evaluates our work, and Section 5 discusses the differences between our work and previous works. Finally, Section 6 concludes the paper and outlines the future work.

2 Dataspace techniques

2.1 Characteristics of dataspace

The concept of dataspace was proposed in 2005 SIGMOD [Franklin et al. 2005]. A dataspace contains all of the information relevant to a particular organization regardless of its format and location, and models any kind of relationships among individual data sources [Halevy et al. 2006].

Dataspace Management System (DSMS) manages all data of a particular organization, and provides the services, such as search and query, over dataspaces without requiring expensive semantic integration [Franklin et al. 2005]. DSMS differs from Database Management System (DBMS), in which the user needs to create a schema of the domain and populate the database with tuples before powerful services can be provided. This is to say, DBMS can not bring database style querying to the data which is not managed by database management systems. In sharp contrast to DBMS, DSMS provides services over all the diverse data regardless of its format and location. DSMS also differs from current information integration systems which require semantic integration of the sources before any benefit can be obtained. In sharp contrast to data integration systems, DSMS does not require full semantic integration before useful services can be provided, and it provides best-effort services in a pay-as-you-go fashion.

2.2 Personal dataspace management system iMeMex

In recent years, some Personal Dataspace Management Systems (PDSMS) have been developed, among which iMeMex [Dittrich 2006] is a famous platform for personal dataspace management, that embodys the key principles of pay-as-you-go fashion, such as ease of setup and incremental integration.

2.2.1 iDM

iMeMex represents all the heterogeneous and distributed mix of personal information into a single data model, called iDM [Dittrich and Salles 2006]. iDM describes all personal information (e.g., Word documents, XML, relational data, file content, folder hierarchies, email and data streams) by resource view graph. A resource view is defined as follows [Dittrich and Salles 2006].

Definition 1 (Resource View). A resource view V_i is a 4-tuple $(\eta_i, \tau_i, \chi_i, \gamma_i)$, where η_i is a string that represents the name of V_i. τ_i is a 2-tuple (W, T), W represents the attributes of V_i, and T represents their corresponding values. χ_i records the content of V_i. γ_i is a 2-tuple (S, Q) which represents the resource views related to V_i. S is a set of resource views, and Q is an ordered sequence of resource views.

The components of resource view can express structured, semi-structured and unstructured pieces of data, and these resource views are linked to each other for forming resource view graph. One important aspect of iDM is that it is lazily computed, i.e., all nodes and connections in the graph may be computed dynamically as deemed necessary. Therefore, iDM can support intensional data (e.g., data obtained by executing a query).

2.2.2 iQL

On top of iDM, the query language iQL [Dittrich and Salles 2006] is proposed to allow users to write intuitive keyword searches with structural restrictions.

The core of iQL is composed of keyword and path expressions. iQL is similar in spirit to NEXI [Trotman and Sigurbjörnsson 2005]. However, iQL includes features important for a PDSMS, such as support for updates and continuous queries.

In the following, we present some example queries.

Q_1:="weather",

Q_1 returns those resource views containing the keyword "weather".

Q_2:= [class="category"],

Q_2 returns those resource views having the attribute-value pair (Class, category).

2.2.3 Association trails

In iMeMex, association trail [Salles et al. 2010] is proposed to define the fine-grained relationships among individual instances in a dataspace intensionally, and its definition is shown as follow:

Definition 2 (Association Trail). A unidirectional (bidirectional) association trail is denoted as either

$$A := Q_L \xRightarrow{\theta(l,r)} Q_R \ \text{ or } \ A := Q_L \xLeftrightarrow{\theta(l,r)} Q_R,$$

where A is a label, θ is a predicate, and Q_L, Q_R are queries. The unidirectional association trail means that the query results $Q_L(G)$ are related to the query results $Q_R(G)$ according to θ, which takes as inputs one query result from Q_L and one from Q_R. Then, one virtual edge is added from left to right and labeled A, for each node pair given by $Q_L \bowtie_\theta Q_R$. The bidirectional association trail also means that the query results $Q_R(G)$ are related to the query results $Q_L(G)$ according to θ.

From the above definition, we can see that an association trail connects two groups of elements from the data sources with a join predicate. Moreover, the system can use many different processing strategies to answer queries over the intensional graph created by association trails. These strategies achieve a greatly improvement in processing cost over the approach of fully materializing the graph and then processing queries over it.

2.2.4 iTrails

In iMeMex, iTrail (semantic trail) [Salles et al. 2007] is proposed to map one query to another. Using iTrails technique, the lightweight integration hints, which are then exploited to improve the accuracy of query results, may be gradually provided to the system. iTrail is summarized in the following definition [Salles et al. 2007].

Definition 3 (iTrail). A unidirectional (bidirectional) iTrail is denoted as either:

$$\psi := Q_1 \longrightarrow Q_2 \quad \text{or} \quad \psi := Q_1 \longleftrightarrow Q_2.$$

The unidirectional iTrail means that Q_1 induces Q_2, i.e., whenever we query for Q_1, we should also query for Q_2. The bidirectional iTrail means that Q_2 also induces Q_1.

Users may define iTrails by themselves. They may also obtain a set of trail definitions by mining semi-automatically from content. For example, $\psi := car \longrightarrow auto$ could be automatically generated from Wordnet. Machine learning techniques can also be used to create keyword-to-keyword trails.

All the iTrails and association trails may be obtained (semi-)automatically, and users may extend the trail set in a pay-as-you-go fashion.

3 Discovering web services in a pay-as-you-go fashion

3.1 Overview of our framework

Our framework, which aims to discover web service in a pay-as-you-go fashion, is based on iMeMex system and dataspace techniques. Figure 1 illustrates the outline of our framework.

In this framework, WSDG is constructed (semi-)automatically by extracting relevant web services information (e.g., functionality descriptions, operations, inputs and outputs) from data sources, such as WSDL files, UDDI repositories and so on. These data sources are published by the service providers, who can also describe their services using WSDG directly.

On top of WSDG, simple keyword-based query is supported by iQL. In order to provide the powerful services, we extend the association trails technique to declare the similarity among web services and support similarity-based service discovery. We also use the iTrails technique to add semantics to the query processing. The service requester can find the right service by keyword-based query or similarity-based service discovery.

It is important to note that our framework provides best-effort services in a pay-as-you-go fashion. That is, the simple services (e.g., keyword search) are provided without requesting to fully materialize WSDG beforehand, and then more powerful services (e.g., similarity-based service discovery and semantic search) are provided gradually with time when more efforts are invested.

3.2 Describing web services using WSDG

3.2.1 WSDG

In recent years, some dataspace data models based on iDM are proposed for different scenes [Salles et al. 2007, Salles et al. 2010]. However, these models, in-

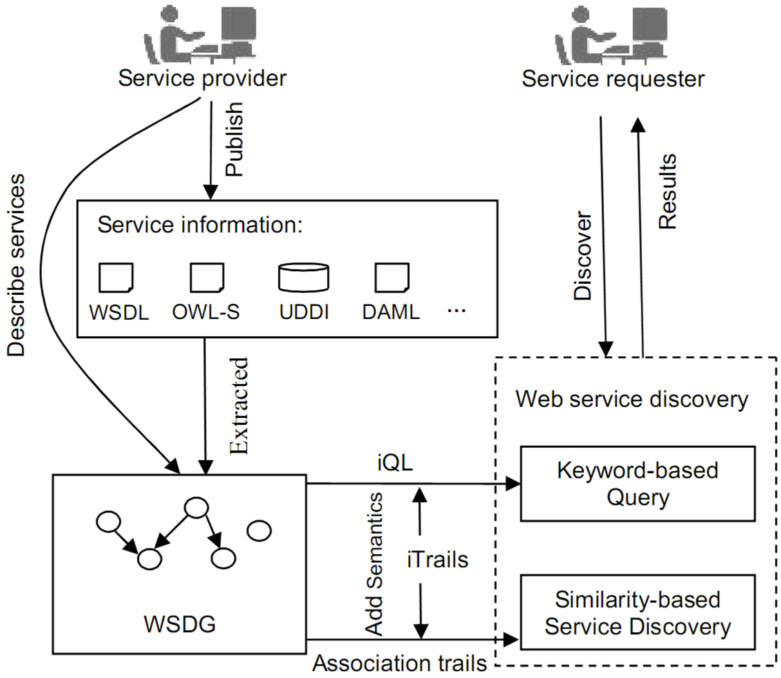

Figure 1: *Framework to discover services in a pay-as-you-go fashion*

cluding iDM, have not involved the issue of the representation of web services, and they are not suitable for web services field. For example, iDM and the Data Model in [Salles et al. 2007] both focus on describing a sequence of ordered relationship among data sources. However, the strength of relationship (such as the degree of similarity between web services) is more important than the order of relationship for service management and discovery. In order to describe web services concisely and intuitively, we adapt the model mentioned in [Salles et al. 2010] to Web Services Description Graph, which is summarized in the following definition.

Definition 4 (Web Services Description Graph, WSDG). The web services are described by a WSDG $G := (N, E)$, where N is a set of nodes. Each node N_i is a set of attribute-value pairs. Each value can be atomic, a bag of words or text content. E is a set of labeled, directed edges (N_i, N_j, L), where L is a label and $N_i, N_j \in N, i \neq j$.

It is easy to describe web services by WSDG, which represents web services as nodes and the relationships among web services as edges.

For example, a web service W_1 may be represented as one node N_{W1} (or several nodes) with a set of attribute-value pairs as follows:

{(Class, service), (Name, WeatherFetcher), (Operation, GetWeather), (Input, PostCode), (Output, [Temperature, WindChill, Humidity]), (TextualDescriptions, ...), ...}, where the value of "TextualDescriptions" is text content to describe the web service.

Similarly, a UDDI category C_1 can be represented as a node N_{C1} with a set of attribute-value pairs as follows:

{(Class, category), (Name, ...), ...}.

If W_1 is in the category C_1, then

E= {(N_{W1}, N_{C1}, BelongTo)}.

In general, E may contain explicit connections among nodes. When all the connections among nodes are not explicit, $E = \emptyset$.

It is worthwhile to note that, there are many XML-based representations for web services (e.g., WSDL, DAML and OWL-S), and our data model can represent these XML-based documents as follows: Each node in the document is represented as a corresponding node in WSDG, and the connections among nodes are given by E components. For example, the elements of WSDL document (e.g., services, operations, messages and data types) and their contents are represented by WSDG nodes and a set of attribute-value pairs, respectively. In addition, the relationships among elements are represented by edges with labels such as "hasoperations" and "hasmessages".

WSDG is a very loosely structured data model, which does not enforce a schema over the data, that is, the set of attributes of each node may be different, and the values of attributes may represent arbitrary unstructured content (e.g., the natural-language descriptions of the services).

3.2.2 iDM and WSDG are equivalent

We argue that the data model iDM and WSDG are equivalent. In fact, each node in iDM can be seen as a set of attribute-value pairs. We show the transformations between iDM and WSDG as follows:

1. iDM: name component $\eta_i \Leftrightarrow$ WSDG: (Name, η_i),
2. iDM: content component $\chi_i \Leftrightarrow$ WSDG: (Content, text),
3. iDM: attribute-value component $\tau_i = (W, T) \Leftrightarrow$ WSDG: (W, T),
4. iDM: edge component

$$\gamma_i = (S, Q) \Leftrightarrow \text{WSDG:} \begin{cases} \text{edges: } \{(N_i, N_{S1}, L_{S1}), \ldots, (N_i, N_{Sm}, L_{Sm})\} \\ \cup\{(N_i, N_{Q1}, L_{Q1}), \ldots, (N_i, N_{Qn}, L_{Qn})\} \\ \\ \text{attribute - value pair of } N_i: \\ (SubNode, [N_{Q1}, \ldots, N_{Qn}]), \end{cases}$$

where $N_{Sj} \in S\ (1 \leqslant j \leqslant m)$, $N_{Qk} \in Q\ (1 \leqslant k \leqslant n)$, and the value of $SubNode$ is a sequence of nodes.

Thus, the dataspace techniques in iMeMex system can support both iDM and WSDG. Specially, the language iQL which is used to query iDM could also be used to query WSDG.

3.3 Similarity-based web service discovery

3.3.1 Web service similarity

In iMeMex system, association trails are used to declare fine-grained, instance-level relationships in a dataspace. A core advantage of association trails is that they can model these relationships intensionally. In order to utilize the association trails technologies which follow the pay-as-you-go fashion, we consider similarity as intensional edge (relationship) in WSDG. To this purpose, we need to extend association trails to declare the similarity among web services by defining θ predicate as similarity function, and Q_L, Q_R to return the service nodes set whose similarity should be measured. We then extract the information of these service nodes to create word vectors. Here, web service similarity is defined as follow:

Definition 5 (Web Service Similarity). Given a data graph WSDG, web service similarity is a binary relation on the node pairs (l, r), which are given by:

$$Q_L \bowtie_t^{sim} Q_R = \{(l,r) | l \in Q_L(G), r \in Q_R(G), sim(l,r) \geq t\},$$

where Q_L, Q_R are queries which return those nodes having attribute-value pair (Class, service), $Q_L\,(G)$ and $Q_R\,(G)$ are the result of Q_L and Q_R, respectively. t is a similarity threshold, $t \in (0, 1]$, and $sim\,(l, r)$ is a similarity function.

By definition 5, we can get that web service similarity is one special type of association trail, which has the following features:

1. The expressiveness of Q_L, Q_R must include the query expressions [class= "service"], meaning Q_L and Q_R return the nodes which represent web services. Users can describe more details about the services what they want by adjusting Q_L and Q_R. For example, to describe the similarity between services which are provided respectively by Company A and Company B, Q_L and Q_R are defined as follows:

   Q_L:=[class="service" and provider="Company A"],

   Q_R:=[class="service" and provider="Company B"].

2. Users can define the different similarity functions to the individual similarities. The similarity functions could be computed by classic IR technique, such as Term Frequency/Inverse Document Frequency (TF/IDF), BM25 score and so on. For example, the operation similarity function $sim_{operation}(l, r)$ may be computed as follows: We create a bag of words from the operation descriptions (e.g., operation name, input and output terms), and use TF/IDF cosine similarity measure to compute the similarity of two such bags. The other similarity functions, such as the types similarity function $sim_{types}(l, r)$ and the messages similarity function $sim_{messages}(l, r)$, can be computed by the similar approach.

Note that, in our work, web services are represented as nodes with a set of attribute-value pairs in WSDG. Therefore, the similarity function may process those attributes on-the-fly to determine the word vectors for each node. We can also pre-process the nodes and store the resulting vectors along with them. This could be done by building an additional index structure.

3.3.2 Computing total similarity

Many similar factors (e.g., the similar input, output and text description) between two web services are helpful for users to evaluate similarity. Therefore, we introduce Total Similarity to describe all these similar factors. Each individual similarity is assigned a weight which is dependent on its relevance to the overall similarity. Weights may be specified by the user, determined by the system, or a combination of both.

The method of calculating total similarity is shown as follows:

Given a set of individual similarities predicates $S_* = \{sim_1, \ldots, sim_n\}$, and with corresponding weights $w_1, \ldots, w_n$, $w_i \in [0, 1]$, $\sum_{i=1}^{n} w_i = 1$. Then the total similarity value of node pair (l, r) is given by:

$$Tsim(l, r) = \sum_{i=1}^{n} sim_i(l, r) \times w_i.$$

It follows from Definition 5 that total similarity is also a binary relation on the node pairs (l, r), which are given by:

$$Q_L \bowtie_t^{Tsim} Q_R = \{(l, r) | l \in Q_L(G), r \in Q_R(G), Tsim(l, r) \geq t\}.$$

3.3.3 Similarity-based discovery algorithm

The results of similarity-based service discovery are all node pairs which satisfy $Q_L \bowtie_t^{sim} Q_R$. That is, given the node pairs from $Q_L(G) \times Q_R(G)$ and a similarity threshold t, only those node pairs which have the similarity value equal or greater than t will be included in the results set. In addition, the return results are ranked in a descending order of their similarities.

Some techniques were introduced to accelerate the processing of queries on the association-trail multigraph, such as query materialization and θ-join processing [Salles 2008, Salles et al. 2010]. These techniques are also suitable for similarity-based service discovery. In this section, we present an algorithm for service discovery based on total similarity. Service discovery based on individual similarity can be seen as a special type of service discovery based on total similarity, where $S_* = \{sim_1\}$.

Algorithm 1 Similarity-Based Web Service Discovery

Input: Left query results $Q_L(G)$; Right query results $Q_R(G)$; List of similarity predicates $sim_1, \ldots, sim_n$; Corresponding weights $w_1, \ldots, w_n$; A threshold t.

Output: Result set Res.

1: **For** each node pair $(l, r) \in Q_L(G) \times Q_R(G)$ **do**
2: Compute $Tsim(l, r)$
3: **If** $Tsim(l, r) \geq t$ **then** $Res := Res \cup (l, r)$
4: **End for**
5: Produce results Res that are ranked by a descending order of their similarity

The pseudo-code for similarity-based service discovery is given in Algorithm 1. For each node pair $(l, r) \in Q_L(G) \times Q_R(G)$, we calculate its total similarity $Tsim(l, r)$. If $Tsim(l, r) \geq t$, this node pair is included in the results set. Finally, the results are returned in the descending order of their similarities.

3.4 Adding semantics to query processing

iTrail may be used to encode schema information from different data sources, without requiring full integration of all sources from the start. In the following, we present how to add semantic to query processing through iTrails.

Example 1 . Combining iTrails with keyword query. The iTrail is assumed to be defined as follow:

$$\psi := car \longrightarrow auto.$$

ψ states that a query Q_1:="car" induces a query Q_2: ="auto". When users query for keywords "car" and "price", the search should include the results of the original query Q: ="car" and "price", but also the results of the query Q': ="auto" and "price".

Example 2 . Combining iTrails with similarity-based discovery. The iTrail ψ is assumed to be defined as Example 1. Given a web service similarity sim_1, which is a binary relation on the node pairs (l, r) given by:

$$Q_L \bowtie_t^{sim_1} Q_R = \{(l, r) | l \in Q_L(G), r \in Q_R(G), sim_1(l, r) \geq t\},$$

where Q_L:= [class= "service" and provider = "Company A"], Q_R:= ["car" and class= "service"].

According to the definition of ψ, the query Q_R:= ["car" and class = "service"] should include the results of the original query but also the results of the query Q:= ["auto" and class = "service"].

That is, sim_1 is a relation on the node pairs (l, r) which satisfy:

$$Q_L \bowtie_t^{sim_1} Q_R = \{(l, r) | l \in Q_L(G), r \in Q_R(G) \cup Q(G), sim_1(l, r) \geq t\} .$$

4 Evaluation

In this section, experiments have been performed: (1) to estimate how the performance of service discovery methods would be affected by Section 3.4's semantics addition method; (2) to evaluate the performance of similarity-based service discovery method based on individual/total similarity.

4.1 Experimental setup

All experiments were performed on top of iMeMex system which we have extended to support web service discovery, and the computer used for the experiments was a dual Intel(R) Atom(TM) CPU 1.66GHz with 1 GB of RAM. In order to evaluate our approaches, we acquired a set of service descriptions written in WSDL from SAWSDL-TC 3 [2], which is a public test collection to support the evaluation of the performance of SAWSDL semantic service matchmaking algorithms. Our test collection contains 480 services which covering 3 domains: travel, communication and economy. For each domain, we selected some queries (service requests) and used their relevance sets (true answers) for evaluation. Then WSDG was constructed automatically by extracting these service descriptions. The information about the experimental domains and services is shown in Table 1.

Table 1: Characteristics of the web services used in the evaluation

Domain	Total Size(MB)	# of Services
Travel	4.38	164
Communication	0.29	58
Economy	6.63	358
Total	11.3	580

We have assessed the experimental results using the IR metrics recall and precision. Recall computes the percentage of the number of relevant retrieved

[2] http://projects.semwebcentral.org/projects/sawsdl-tc/

documents for a query over the total number of relevant documents for that query. Precision computes the percentage of the number of relevant retrieved documents over the total number of retrieved documents.

4.2 Performance of semantics addition

We compare the keyword-based query approach without semantics addition to that with semantics addition by rewriting the queries utilizing iTrails technique. The iTrails used for evaluation are defined as follows:

$\psi_1 := car \longrightarrow auto,$
$\psi_2 := car \longrightarrow vehicle,$
$\psi_3 := book \longrightarrow novel,$
$\psi_4 := book \longrightarrow monograph,$
$\psi_5 := food \longrightarrow sandwich,$
$\psi_6 := food \longrightarrow coffee,$
$\psi_7 := food \longrightarrow bread,$
$\psi_8 := video \longrightarrow film,$
$\psi_9 := country \longrightarrow village,$
$\psi_{10} := geopolitical\ entity \longrightarrow geographical\ region.$

The average response times of keyword-based queries with/without semantic addition are shown in Table 2. The response times are obtained on a warm cache, i.e., each query is run several times until the deviation on the average response times becomes small. All queries response times are less than 0.2 seconds. Taken as a whole, the gap of response times between semantic addition and without semantic addition is very limited. We can conclude that our approaches do not add much overhead for semantic addition.

Table 2: Average response times of queries with/without semantic addition (measured in seconds)

Query ID	Expression	No iTrails	With iTrails
Q_1	car price	0.11	0.14
Q_2	book price	0.09	0.13
Q_3	prepared food price	0.07	0.12
Q_4	video media	0.06	0.07
Q_5	city country hotel	0.08	0.1
Q_6	geopolitical entity weather process	0.09	0.11

Figure 2 shows the performance (recall/precision) of semantics addition. As we may notice, the recall and precision of all queries are sharply improved by adding trails in a pay-as-you-go fashion. For example, the precision of Q_1 with

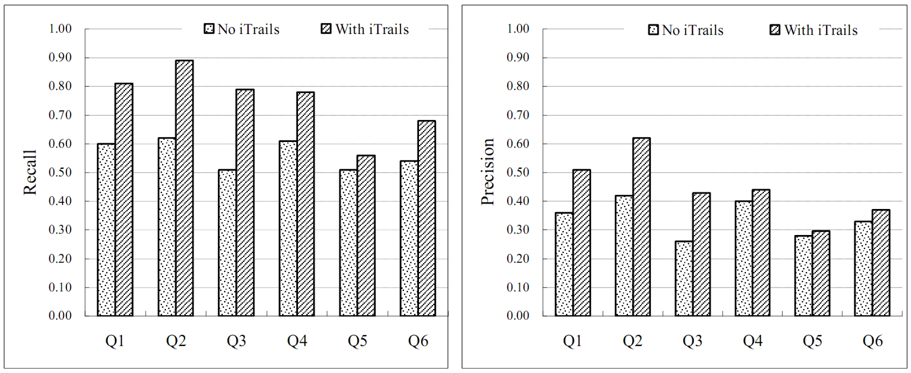

Figure 2: *Performance of keyword-based query with/without semantics addition*

iTrails (i.e., ψ_1 and ψ_2) is about 15% higher than that without iTrails, and the recall of Q_3 with iTrails (i.e., ψ_5, ψ_6 and ψ_7) is about 25% higher than that without iTrails. Moreover, both the recall and the precision of Q_2 with iTrails (i.e., ψ_3 and ψ_4) are about 20% higher than those without iTrails.

In summary, the experimental results show that our semantics addition method strongly improves the quality of query results when compared to the approach providing keyword and structural search which has no integration semantics of the data.

4.3 Performance of similarity-based service discovery

Experiments have been performed to evaluate the contribution of individual similarity by comparing service discovery method based on total similarity with that based on individual similarity.

Here, let Q_L:=[class="service"and node=N_r], Q_R:=[class="service"], t = 0.5, and $S_* = \{sim_{input}, sim_{messages}, sim_{types}\}$, with corresponding weights $W_{input} = 0.4$, $W_{messages} = 0.5$, and $W_{types} = 0.1$, where the request description is represented by the node N_r, i.e., $Q_L(G)$ include only one node N_r, besides, sim_{input}, $sim_{messages}$ and sim_{types} denote the similarity of inputs, messages and types, respectively. These similarities are computed following the approach mentioned in Section 3.3, the threshold and the weights are set manually based on the analysis of the results from different trials. In the future, we plan to use machine learning techniques (e.g., support vector machine) to learn weights and threshold.

Table 3 shows the response times for the queries evaluated. Most queries are executed in less than 0.5 seconds. The only exception is the total similarity $Tsim$.

Moreover, all the response times of discovery method based on total similarity are more than that based on individual similarity. Generally, for each pair of services descriptions, the number of similarity predicates (similarity functions) is usually less than 10. For example, there were 4 similarities predicates used in [Dong et al. 2004] and [Wu and Wu 2005] respectively. Therefore, there would be an upper limit on the response times of total similarity discovery method. In addition, we also plan to pre-process the nodes and store the word vectors along with them for providing even better response times in such situations.

Table 3: Average response times of similarity-based service discovery (measured in seconds)

Domain	sim_{input}	$sim_{messages}$	sim_{types}	$Tsim$
Travel	0.25	0.38	0.39	0.82
Communication	0.26	0.29	0.29	0.66
Economy	0.37	0.43	0.42	0.91

The precision and recall of similarity-based discovery method over three domains are shown in Figure 3. The results show that discovery method based on total similarity is better than that based on individual similarity. The reason is that total similarity provides additional evidence for service similarity assessment. For discovery method based on individual similarity, the precision and recall of $sim_{messages}$ is higher than that of the other similarities, and the precision and recall of sim_{types} is the lowest. That is because $sim_{messages}$ provides the strongest evidence while sim_{types} provides the weakest one. Note that, the values of weights and threshold are adjustable by the users for the characteristics of service descriptions, and the adjustment would affect the results of service discovery method. Thus, we draw the conclusion that our service discovery method based on total similarity is better with respect to service discovery based on individual similarity, in the case of the appropriate values of weights and threshold have been set.

5 Discussion

In this section, we compare our work with previous studies from three aspects: web service description & query, similarity-based service discovery and semantics addition. For each of these aspects, we focus on the pay-as-you-go features of our work.

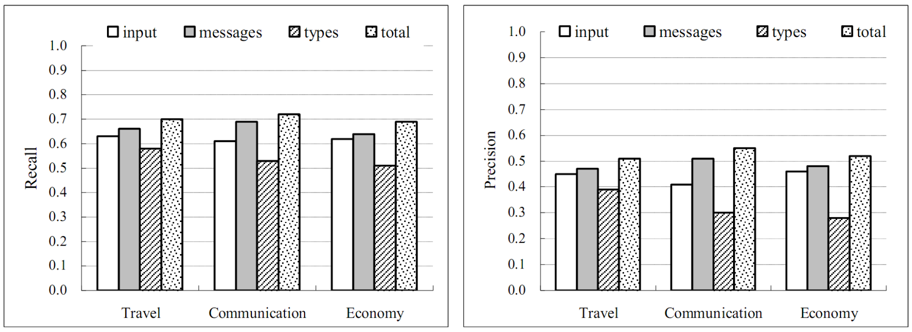

Figure 3: *Performance of individual/total similarity discovery*

5.1 Web service description and query

In order to discover web services effectively, extensive efforts have been focused on service description. For example, WSDL, an XML-based language, is used to describe the interface of web services, and semantic WSDL (i.e., WSDL-S) is a set of elements extending WSDL by using ontological concepts for enriching services with semantics [Akkiraju et al. 2005]. Besides this, OWL-S is an OWL ontology for describing semantic metadata about web services [Martin et al. 2007]. However, all of these service descriptions are based on a schema-first modeling strategy that need mapping data to a schema or a domain model (e.g., ontology) before querying may be carried out. These schema-first strategies make it difficult for integrating data in a pay-as-you-go fashion [Salles 2008]. In contrast, WSDG is a loosely structured data model which may be constructed in a pay-as-you-go fashion. That is, from the start, the simple information and very few (or inaccurate) relationships are depicted. With time, the system gets more familiar with the data, it depicts more accurate information and more complex connections in WSDG as deemed necessary. Moreover, in sharp contrast to these previous studies which are usually tied to having a physical representation of the whole data before querying, any of the components of the WSDG are not required to be materialized beforehand. It means that, all nodes and edges in WSDG may be computed lazily on demand.

5.2 Similarity-based service discovery

IR technologies have been proposed to measure the degree of similarity between services. In [Dong et al. 2004] authors proposed a clustering-based algorithm to combine multiple sources of evidence for determining similarity between a pair of web-service operations. In [Stroulia and Wang 2005] the methods were proposed to measure the similarity between two WSDL descriptions based on not only the semantics of their identifiers but also the structure of their operations, messages, and data types. In [Wu and Wu 2005], a web service conceptual model was presented, and the properties of service were divide into four categories. For each category, a similarity measure is given, and these similarity measures can be used together or individually. In our work, we combine association trails with IR technologies to define similarity and discover services. In contrast to the previous approaches, our similarity is defined as intensional edge in WSDG which can be computed lazily. In another words, the service nodes set and their information which are used to measure the degree of similarity can be obtained lazily, and thus we can compute similarity in a pay-as-you-go fashion.

5.3 Semantics addition

Extensive efforts have been devoted to combining web services and semantic web technologies. By utilizing ontologies to represent web services and background knowledge [Akkiraju et al. 2005, Martin et al. 2007, Cardoso et al. 2009], user can find appropriate services more efficiently. Moreover, there are also some studies on the semantics similarity-based service discovery. In [Bianchini et al. 2008] authors proposed a service discovery approach which combined an ontology-based deductive matchmaking and a similarity-based matchmaking derived from IR techniques. In [Klusch et al. 2009] OWLS-MX was presented to perform hybrid semantic service matching for OWL-S descriptions, and it complemented crisp logic-based semantic matching with syntactic similarity-based matching in case the former failed. In order to support an automated and veracity service discovery process, the authors of [Liu et al. 2009] presented a weighted ontology-based semantic similarity algorithm under the semantic web service framework. Unlike the previous semantics-based approaches which require significant efforts to declare data semantics through complex ontologies to enrich query processing, we add semantics by iTrail. Our approach is much more lightweight, allowing users to add semantics over time in a pay-as-you-go fashion. However, for the case of similarity-based service discovery, our work only focus on adding semantics into the query processing of Q_L and Q_R by iTrail, while not considering using semantics to compute the similarity function. That is, we currently can not fully realize semantics similarity-based service discovery in a pay-as-you-go fashion.

6 Conclusions and future work

Dataspace techniques, which emphasize the idea of pay-as-you-go data management, have received increasing attention in database and data integration communities. However, to the best of our knowledge, dataspace techniques have not gained much attention among researchers in the field of web service discovery until now. In this paper, we suggest a framework based on dataspace techniques to discover web services in a pay-as-you-go fashion. The framework is mainly based on three strategies: (a) presenting WSDG, which can be constructed gradually, to describe web services; (b) extending association trails techniques to support similarity-based service discovery on top of WSDG; and (c) using iTrails techniques, which can add semantics to the query processing gradually, to improve the quality and accuracy of service discovery.

In the future, we plan to integrate the technologies of ontology and Natural Language Processing for extracting web services information more effciently. Another future work is about adding semantics. To be specific, we are going to focus on how to fully realize semantics similarity-based web service discovery in a pay-as-you-go fashion.

Acknowledgements

This research is supported by the National Natural Science Foundation of China under Grant Nos. 60970044, 60673135, 60736020; the Natural Science Foundation of Guangdong Province of China under Grant No. 7003721.

References

[Adamopoulou et al. 2007] Adamopoulou, P., Sakkopoulos, E., Tsakalidis, A., Lytras, M.: "Web service selection based on QoS knowledge management"; Journal of Universal Computer Science, 13, 9 (2007), 1138-1156.

[Akkiraju et al. 2005] Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., Verma, K.: "Web service semantics-WSDL-S"; `http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/`.

[Bianchini et al. 2008] Bianchini, D., De Antonellis, V., Melchiori, M.: "Flexible semantic-based service matchmaking and discovery"; World Wide Web, 11, 2 (2008), 227-251.

[Cardoso et al. 2009] Cardoso, J., Sheth, A., Yu, L.: "Semantic Web services, processes and applications"; RECIIS, 3, 1 (2009), 85-88.

[Dittrich 2006] Dittrich, J.: "iMeMex: A platform for personal dataspace management"; SIGIR PIM Workshop, Citeseer, Washington (2006), 40-43.

[Dittrich and Salles 2006] Dittrich, J., Salles, M.: "iDM: a unified and versatile data model for personal dataspace management"; Proceedings of the 32nd international conference on very large data bases, VLDB Endowment, Seoul, Korea (2006), 367-378.

[Dong et al. 2004] Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: "Similarity search for web services"; Proceedings of the 13th international conference on very large data bases, VLDB Endowment, Toronto, Canada (2004), 372-383.

[Franklin et al. 2005] Franklin, M., Halevy, A., Maier, D.: "From databases to dataspaces: a new abstraction for information management"; ACM Sigmod Record, 34, 4 (2005), 27-33.

[Halevy et al. 2006] Halevy, A., Franklin, M., Maier, D.: "Principles of dataspace systems"; Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, ACM, Chicago, IL, USA (2006), 1-9.

[Klusch et al. 2009] Klusch, M., Fries, B., Sycara, K.: "OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services"; Web Semantics: Science, Services and Agents on the World Wide Web, 7, 2 (2009), 121-133.

[Liu et al. 2009] Liu, M., Shen, W., Hao, Q., Yan, J.: "An weighted ontology-based semantic similarity algorithm for web service"; Expert Systems with Applications, 36, 10 (2009), 12480-12490.

[Martin et al. 2007] Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D., Sirin, E., Srinivasan, N.: "Bringing semantics to web services with OWL-S"; World Wide Web, 10, 3 (2007), 243-277.

[Salles et al. 2010] Salles, M., Dittrich, J., Blunschi, L.: "Intensional associations in dataspaces"; IEEE 26th international conference on data engineering, IEEE, Long Beach, California, USA (2010), 984–987.

[Salles 2008] Salles, M. A. V.: "Pay-as-you-go Information Integration in Personal and Social Dataspaces"; Ph.D. thesis, ETH ZURICH (2008).

[Salles et al. 2007] Salles, V., Antonio, M., Dittrich, J., Karakashian, S., Girard, O., Blunschi, L.: "iTrails: Pay-as-you-go Information Integration in Dataspaces"; Proceedings of the 33rd international conference on very large data bases, VLDB Endowment, Vienna, Austria (2007), 663–674.

[Stroulia and Wang 2005] Stroulia, E., Wang, Y.: "Structural and semantic matching for assessing web-service similarity"; International Journal of Cooperative Information Systems, 14, 4 (2005), 407–437.

[Trotman and Sigurbjörnsson 2005] Trotman, A., Sigurbjörnsson, B.: "Narrowed extended xpath i (NEXI)"; Advances in XML Information Retrieval, 3493 (2005), 16–40.

[Wu and Wu 2005] Wu, J., Wu, Z.: "Similarity-based Web Service Matchmaking"; Proceedings of IEEE international conference on services computing, IEEE, Orlando, Florida, USA (2005), 287–294.