A Concept Analysis Approach for Guiding Users in Service Discovery

Bipin Upadhyaya, Foutse Khomh, Ying Zou
Department of Electrical and Computer Engineering
Queen's University
Kingston, Canada
{bipin.upadhyaya, foutse.khomh, ying.zou}@queensu.ca

Alex Lau, Joanna Ng
IBM Canada CAS Research
Markham, Ontario, Canada
{alexlau, jwng}@ca.ibm.com

Abstract—Web services are widely used as basic constructs to build complex distributed applications with fast speed and low cost. However, existing service discovery techniques provide users with poor results which require substantial human intervention to filter the services to locate the desired ones. In particular, users often have no prior knowledge of the functional description of the available services on the Web. The queries formulated by the users may not match well with the service descriptions of existing services. As a consequence, a user's query can result in a large number of returned services. In this paper, we propose an approach that derives the semantic concepts conveyed in the service descriptions and clusters the services based on the concepts. As a result, each concept is associated with a set of relevant services. To understand the semantic meanings of a user's query, we identify concepts behind the query and recommend related concepts associated with services. Our approach also guides users to formulate their queries. We conducted a case study and found that the average precision and recall of our approach for service discovery are respectively, 83% and 100%. We also performed a user study which shows that for 85% of time, a user reformulates their queries using the suggestion provided by our approach to improve the precision of the retrieved services.

Keywords- Web service discovery; search, concepts

I. INTRODUCTION

A Web service is a software component designed to support interoperation between machines over the Web [2]. Web services are increasingly used as basic constructs for rapidly developing low-cost distributed applications [6]. A service description document describes the interface of a Web service. A Web service can be implemented by various technologies, such as SOAP based Services and RESTful Services. SOAP based services are described in the Web Service Description Language (WSDL) [17] and communicated over Simple Object Application Protocol (SOAP). RESTful services are resource centric [14]. A resource is a remote accessible object of an application. Each URL in RESTful service represents a resource and can be invoked using standard HTTP methods[3]. RESTful services can be described by different languages, including Web Application Description Language (WADL) [18] and Resource Linking Language (ReLL) [13].

Typically, a service is discovered in four steps: 1) service providers advertise the capabilities of a service using service description documents (*e.g.,* WSDL and WADL) and register the service in a service repository; 2) an agent crawls different service repositories and stores the service description documents locally; 3) a service requester queries the agent to request for service providers that can best match the desired capabilities of services; and 4) the agent matches the request against the stored service descriptions and returns the matched result.

With the ever-increasing number of Web services published on the Internet (*e.g.,* Google has indexed 204,000[1] WSDL), it becomes challenging to find desired Web services. In particular, current approaches often return a large number of irrelevant Web services as the result of service discovery. It is time consuming for users to sift through the results to locate a desired Web service. Therefore, an effective service discovery mechanism is essential to help developers harness the benefits offered by Web services. The large body of research on service discovery can be summarized into three main categories: semantic Web approaches; Web search engines; and information retrieval (IR) approaches. Semantic Web approach is proposed to enhance service description documents by annotating the description document with domain specific ontologies. Web search engines (*e.g.,* Google[1] and Bing[2]) are used as new tools for discovering Web services [15]. Service descriptions usually reside in Web servers. Web search engines crawl and index the contents of servers and enable users to retrieve Web services. IR approaches [22, 23, 24] such as word sense disambiguation, stop-words removal, and stemming, have been used to extract relevant information conveyed in service description documents to index Web services. However, the existing approaches in service discovery suffer from the following two limitations:

Gap in the semantics of service descriptions and users' search queries. The precision of service discovery approaches is dependent on the understanding of the semantics of service description documents and user's query. The vocabulary adopted in service description is often used by developers in the software development domain. It can be very different from the ones used in users' queries. For example, a user may query for transportation services with vague description. However, service providers may describe services with more specific types such as, car, taxi, and bus. Without a good understanding of the semantics of service descriptions and queries, a service discovery approach is likely to retrieve a large number of irrelevant Web services or fail to return any Web service.

Limited support for query formulation. The available service descriptions act as a black box to users. A user has to conduct many trials to formulate an appropriate query to retrieve the desired services when an initial query fails. A user cannot specify searching criteria based on a particular requirement. For example, to reserve a two bed room hotel in downtown Toronto, users may either write a whole sentence

[1]http://www.google.com
[2]http://www.bing.com
[3]www.w3.org/Protocols/rfc2616/rfc2616.html

or specify a few keywords, like "reserve room". In both cases, users may receive too many results that need considerable amount of manual filtering. Therefore, it is critical to provide users with efficient ways to articulate service queries (*i.e.,* considering input and output of operations) to improve the precision of the service discovery.

To address the aforementioned limitations, we propose an approach that identifies the semantic meanings of service descriptions and users' queries as concepts. More specifically, a concept is a semantic notion or a keyword for describing a subject, e.g., "*traveling*", "*weather*" or "*taxi reservation*". In particular, we identify a set of concepts from the service description documents. We further index services in terms of their associated concepts and the relations among concepts using WordNet [8], a lexical database. Moreover, our approach assists users in formulating their queries using the similar vocabulary as the ones specified in service description documents. We use the commonsense knowledge (*e.g.,* checking reviews before watching a movie), encoded in ConceptNet [5] to link the concepts delivered in the service descriptions and the concepts extracted from a user's query. Commonsense knowledge is not available in a lexical database like WordNet. We provide a graphic tool that guides users to select the generated concepts from their queries, and allows users to combine different concepts to better describe their searching criteria.

The remainder of this paper is organized as follows. Section II discusses background knowledge on Web service description documents, WordNet and ConceptNet. Section III presents an overview of our approach. Section IV describes the prototype of our approach. Section V describes the case study and discusses results. Section VI discusses the related works. Finally, Section VII concludes the paper and explores future work.

II. BACKGROUND

A Web service description document is used by service providers to specify the interface and the capabilities of a Web service. Moreover, we use WordNet and ConceptNet to identify the semantic relations between concepts associated to Web services. In the following sub sections, we give a brief overview of service description documents, WordNet and ConceptNet.

A. Web Service Description Documents

WSDL is an XML-based language that describes the interface of Web services. A WSDL document specifies messages, operations, interfaces, bindings, and service endpoints. Figure 1 shows an excerpt of a WSDL document. The <*portType*> element in a WSDL lists one or more operations and the <*message*> element defines data elements corresponding to input, output and fault. Some of the service description documents link data types in a separate XML schema document. As illustrated in Figure 1(A), *getRate* is an operation which takes two inputs, *country1* and *country2*, and returns, *Result* as a float type.

WADL is an XML-based file format that provides a machine-readable description for HTTP-based Web applications. WADL describes the name of RESTful services, resource URLs, request parameters and response parameters. A URL usually consists of a scheme name, a domain name, a port number, a path for fetching a resource, a query string, and an optional fragment identifier. Figure 1 (B) shows an example URL for a resource, i.e., *http://search.yahooapis.com/NewsSearchService/V1/news Search*. It uses the HTTP method GET. The parameters for the request are *appid* and *query*. The parameter for the response is the status code along with an element (*i.e., yn:ResultSet*), and a media-type (*i.e., application/xml*) to define the schema for interpreting the results.

```
....
<message name="getRateResponse">
   <part name="country1" type="xsd:string />
   <part name="country1" type="xsd:string />
</message>
<message name="getRateResponse">
   <part name="Result" type="xsd:float"/>
</message>
<portType name="CurrencyExchangePortType">
   <operation name="getRate">
      <input message="tns:getRateRequest"/>
      <input message="tns:getRateRequest"/>
  </operation>
</portType>
.....
```

(A) WSDL (CurrencyExchange.wsdl)

```
...
 <resource path="newsSearch">
  <method name="GET" id="search">
   <request>
     <param name="appid" type="xsd:string" />
     <param name="query" type="xsd:string" />
      ...
   </request>
   <response status="200">
    <representation mediaType="application/
                    xml" element="yn:ResultSet"/>
   </response>
   ...
   </method>
  </resource>
...
```

(B) RESTful Service (Yahoo news search)

Figure 1: Excerpt of service description documents

B. WordNet and ConceptNet

WordNet is a lexical database to group concepts into sets of synonyms and connects concepts via semantic relations. WordNet defines four types of relations between concepts as follows:

- Hypernym represents "kind of" relation. For example, *car* is a hypernym of *vehicle*;
- Hyponym is the inverse of hypernym meaning that a concept is a super name of another. For example, *vehicle* is a hyponym of *car*;
- Holonym describes whole-part (*i.e., partOf*) relation. For example, *building* is a holonym of *window*;
- Meronym represents a part-whole relation. For example, *window* is a meronym of *building*.

ConceptNet [5] is a relational semantic network of different concepts that encompasses spatial, physical, social, temporal, and psychological aspects of everyday life. ConceptNet can be visualized as a graph of nodes and edges. A node represents a concept in the form of natural language fragments (*e.g., 'food', and 'grocery store'*). An edge describes semantic relations between two concepts. Each edge is labeled with a relation type (*e.g.,* "*IsA*" or "*LocatedAt*") and a score that describes the validity of the relation. There are 20 types of relations define to cover commonsense knowledge used in the real world [5]. ConceptNet provides a richer and pragmatic set of semantic relations between concepts.

III. OVERVIEW OF OUR APPROACH

Our approach consists of two major steps: 1) service indexing which extracts concepts from service description documents and cluster services using the concepts conveyed in the service description; and 2) service retrieval that

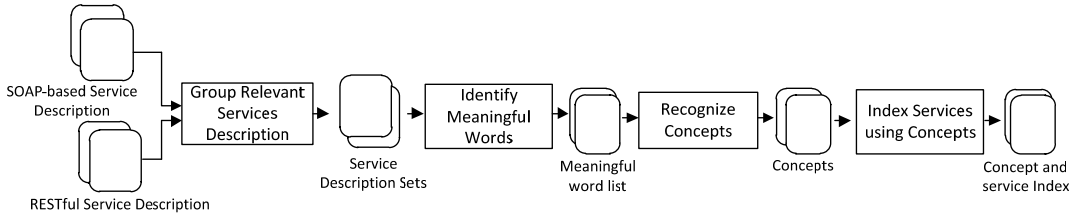

Figure 2: Overall steps for indexing services using the concepts extracted from service description documents

extracts concepts from a user's query, guides users to formulate queries and returns the services associated with the concepts. The service indexing is an offline procedure which is invoked once for each service. The service retrieval is an online step that is executed for each query. In the service retrieval step, we provide a mechanism to recommend concepts and allow a user to navigate through the concepts associated with services. In the following subsections, we discuss the two steps in more details.

```
<portType name="CurrencyExchangePortType">
    <operation name="getRate">
    <input message="tns:getRateRequest">
      <message name="getRateResponse">
          <part name="country1" type="xsd:string />
          <part name="country1" type="xsd:string />
      </message> </input>
    <output message="tns:getRateRequest"/>
      <message name="getRateResponse">
          <part name="Result" type="xsd:float"/>
      </message> </output>
</operation></portType>
```

Figure 3: Rearranged service description of Figure 1(A)

A. Service Indexing

Figure 2 gives an overview of the steps involved in the service indexing process. We explain each step as follows:

Grouping Relevant Service Descriptions. The functionality is reflected in the operation descriptions in SOAP based Web services and the resource descriptions for RESTful services. We call an operation or a resource as a service. However, such descriptions are often scattered throughout a Web service description document. For example in Figure 1(A) XML fragments related to *getRate* is scattered in multiple elements, such as *portType* and *message* in the description document. Instead of analyzing the entire service description document, we rearrange tags in service description documents and assemble information related to a service (i.e., an operation or a resource) in one location. The information related to a service is called as a service description set. A service description documents can have multiple service description sets due to multiple operations or resources defined. For SOAP-based services, a service description set contains an operation, its input parameters, its output parameters, and the documentation corresponding to the operation, input and output. Similarly, for RESTful services, a service description set includes a resource, the URL of the resource, request, response, and the documentation corresponding to the resource, its request and its response. Figure 3 shows a service description set after the relevant tag rearrangement of the example shown in Figure 1(A).

Identifying Meaningful Words. To identify concepts from each service description set, we first extract the name of the service (*i.e.,* operation or resource), messages and parameters in a service description set. For example shown in Figure 1(A), the extracted names include *CurrencyExchangePortType, getRate, getRateResponse, getRateRequest, country1, Result*. The extracted names can be compound words. For example, two words are joined by the change of case (*e.g., findCity*); words are separated by underscore (*i.e.,* "_") or dash (*i.e.,* "-") (*e.g., find_city*), and words are added with a suffix (*e.g., city1*). We tokenize compound words into single words. We use WordNet to check if a single word is a valid English word and keep the valid words.

Some names used in service descriptions are not separated by special symbols (*e.g.,* "_", "-" or camel case). For example a RESTful service describes a resource using URL. The names are not valid English words either. To identify meaningful words, we perform an n-gram analysis which can be used to detect sub words within a word by extracting a contiguous sequence of n letters from a string. For example, the 3-gram analysis of a word, *improve*, extracts all the possible consecutive three letters in the string, *i.e., imp, pro, rov, ove*. We perform n-gram analysis starting from 3-grams (*i.e.,* i-grams and i=3) to check if the three consecutive letters in a string are a valid English word using WordNet. Then, we iteratively increase from i-grams to i+1- grams to recognize valid words and continue until i+1 is the maximum length of the string. For example, an n-gram analysis of the URL, *http://bookmooch.com/topic/{topic},* generates a vector, i.e., *{book, topic, mooch}*.

TABLE I: RULES FOR DECOMPOSING WORDS

Rule	Original Word	Tokenized Word
CaseChange	FindCity	Find, City
	addCity	add, City
Suffix with Number	country1	country
Underscore Separator	Find_city	Find, City
n-gram Analysis	http://bookmooch.com/topic	book, topic, mooch
Root words	reserves, reserved	reserve

Moreover, we reduce the words to the root words using the Porter stemmer [9]. For example, '*reserve', 'reserved', 'reserving'*, and *'reserves'* have the same stem '*reserve'*. All these words have the similar semantic meanings. Table I summarizes the rules used to decompose words to identify meaningful words.

Each service description set has one or more words extracted from the service (*i.e.,* operation or resource), inputs and outputs. We classify the words into three groups: service (*i.e.,* resource and operation) Keyword (*SK*)); input parameters (*i.e., IP*); and output parameters (*i.e., OP*). The

extracted words are stored in the format shown in Equation (1). A word list (i.e., *SWord*) stores each word (*i.e., W_i*) extracted from a service description set, the frequency (*i.e., freq*) of the word (W_i) appeared in a service description set. The type attribute separate the *SWord* in the three groups (*i.e., SK, IP and OP*).

$$\text{SWord} = \{(Wi, \text{freq}, \text{type})\} \quad (1)$$

Where SWord is a list of words with frequency and type; W_i is an element in the word list, SWord; freq is the frequency of W_i; and type ∈ {SK, IP, and OP}.

Recognizing Concepts. A word list (i.e., *SWord*) may contain a lot of words. However, not all the words reflect the functionalities of a service. In particular, the names used in a service often consist of verb and noun, such as *getCustomer* and *deleteCustomer*. As a result of the last step (*i.e.,* extracting meaningful words), *get*, *delete* and *customers* are valid words in the word list. However, the general operational words, such as *delete*, *update* and *get*, describe the manipulation on the data without indication of the functionality of a service which is related to customer services in the example. Our aim is to identify and filter general operational words from the words extracted from a service description set. We consider the remaining words as functional words, i.e., concepts.

$$wordSim(x, y) = 1 - \frac{mcp(cp)}{mcp(cp) + dcp\,(cp, root)} \quad (2)$$

where cp is the common parent of the two words x, y; root is the root of the WordNet ontology; mcp(cp) is the shortest path from either x or y to cp; and dcp(cp, root) is the length of the path from cp to root.

To separate words in the word list, *SWord*, into two clusters (*i.e.,* general operational words and functional specific words). We apply k-means algorithm [4], a clustering algorithm, on the word list. We set the number of clusters generated to 2 (*i.e.,* k=2). K-means algorithm is selected due to its simplicity and efficiency. The clustering algorithm groups similar words using the semantic similarity as shown in Equation (2). Concepts in WordNet can be connected by different types of relations such as hypernym, hyponym and holonym. The two concepts can be directly or indirectly connected through many intermediate relations and concepts. A path length is the number of intermediate concepts to traverse from one concept to another. The similarity between two concepts *x* and *y* is measured by the path length between concepts to reach their common parent in WordNet ontology. The value of the similarity metric shown in Equation (2) ranges from 0 to 1. 0 represents unrelated words and 1 signifies synonymous words.

To determine the cluster that contains the general operational words, we predefine an oracle of general operational words (*e.g., update*, *data*, *post*, *add*, and *create*) by manually examining a number of service description sets in different domains. The cluster semantically closer to this oracle is determined to contain operational words and discarded from the word list (*i.e., SWord*). For example, the operation *getRate* shown in Figure 3, we extract a set of words from the service description, i.e., *{get, country, currency, rate, exchange, request, result}*. We further divide the words into two clusters, i.e., *{get, request, result}* and *{country, rate, exchange, currency}*. The cluster *{get request, result}* is close to the predefined oracle and hence is discarded. The remaining cluster *{country, rate, exchange, currency}* contains the functional words, i.e., the concepts representing the functionality of the service.

$$R(x) = \left\{ \sum_{y \in C; y \neq x} WordSim(x, y)\ f(y) \right\} + f(x) \quad (3)$$

where R(x) denotes the rank of the concept x in the cluster C; WordSim(x, y) is the similarity between concept x and y; and f(x) is the frequency of the concept x.

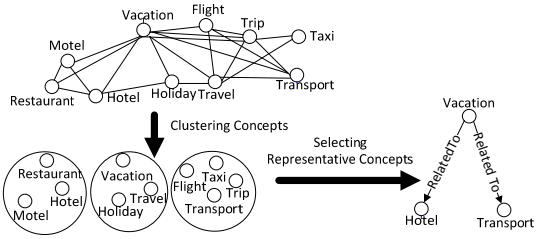

Figure 4: Process for creating a concept map

Indexing Services using Concepts. There is more than one concept describing the functionality of a service. Some of the concepts might be redundant. Other concepts can be less frequently used. We aim to identify a set of representative concepts that can capture the major functionality delivered by a service. We consider the most frequently used concepts as the representatives of the functionality. As aforementioned, the concepts in the refined word list (*i.e., SWord*) are divided in three groups (*i.e., SK, IP and OP*). For example, the concept set {*country, rate, exchange, currency*} contains {*exchange, rate, currency*} associated with the operation (*i.e., SK*) and {*country*} derived from the input (*i.e., IP*). We treat each group individually and rank the frequency of a concept based on the semantic similarity among the concepts within the group. Equation (3) is defined to compute the frequency of concepts in each group. The rank of a concept *x* (*i.e., R(x)*) is the sum of the frequency of concept *x* and the frequency of concept *y* prorated by the semantic similarity between the concepts *x* and *y*. The rank of concept *x* increases if the cluster has more concepts semantically similar to concept *x*. Ranking concepts signifies the frequency of a concept with respect to other concepts in a group. The computed rank is then normalized between 0 to 1 by dividing a rank of a concept with sum of all concepts rank in a group. 1 signifies the most dominant concept. For example, the similarity between *exchange and currency is 0.3; exchange and rate is 0.4; and rate and currency is 0.6*. Using these similarity values, we compute the rank of the concepts {*exchange, currency, rate*}, the concept rank as described in Equation (3) is {*exchange (0.3+0.4+1=1.7), currency (0.3+0.6+1=1.9), rate (0.6+0.4+1=2)*}. If we select two representative concepts, it will be *rate* and *currency*. The input has only one concept and its rank is {*country (1)*}. The number of representative concepts to use is a design decision. We manually examined a few service description sets and found that two concepts can effectively represent a group. Thus we use two

representative concepts for each group. The operation *getRate* in Figure 1(A) is indexed under concepts *currency* and *exchange* due to the semantic relation with the operation concept and also associated with concept *country* because of the relation with the input concept. Similarly the resource "*newsSearch*" in Figure 1(B) is indexed under concept *news*.

B. Service Retrieval

A user expresses queries using the natural language. Studies on users' Web search behaviors have shown that a large portion of Web search queries consist of only one to three keywords [21]. Short queries indicate that users often have difficulties in crafting queries to reflect the information needed. To understand user's intention and requirements, we extract concepts from users' queries and retrieve semantically related concepts from service sets (*i.e., SWord*). Users can select one or more concepts from the retrieved concepts from the service sets to formulate their query. This can provide some knowledge on the available services and help users articulate their specific needs. Hence we can provide services with the best matches. The service retrieval consists of two steps: 1) analyzing queries; and 2) grouping services.

Analyzing Users' Query. A user query can have two characteristics: 1) a few keywords containing key concepts; and 2) a long verbose query including "wh-" questions (who, what, when, where). In the former case, the concepts are directly provided by the user. In the latter case, we parse the user's query to analyze the structure of the sentences in the query and identify concepts from the sentences.

$$Rep(x) = \sum_{y \in C; y \neq x} WordSim(x, y) \quad (4)$$

where Rep(x) is the rank of the concept in a cluster; and WordSim(x, y) is the similarity between concepts x and y.

We create a concept map to help a user visually select concepts to formulate their query. A concept map is a set of concepts identified from a user's query and their relations. The concepts are represented as nodes. A relation between two concepts is derived from ConceptNet and illustrated as an edge linking two concepts. To create a concept map, we cross-reference concepts in a query with semantically related concepts in the service repository. For example if the concept "*vacation*" is present in a user's query, we retrieve concepts related to "*vacation*" from ConceptNet, provided that there are services associated with the retrieved concepts. For example the concept "*vacation*" is related to {*motel, hotel, restaurant, flight, trip, taxi, transport, travel, holiday*} as shown in Figure 4. There may be a large number of related concepts which can be overwhelming to users. We cluster related concepts by measuring their similarity as defined in Equation (2). Each cluster is represented by a representative concept. Equation (4) computes the rank of each concept with respect to other concepts in the cluster. A concept with the highest rank is considered as the representative candidate of the concept cluster. We represent the selected representative concepts and their relations as a concept map. Figure 4 shows the process of creating a concept map.

Table II shows an example of concept clustering and selection of a representative concept. For simplicity we use number of cluster as 3. A concept map should not contain a lot of recommendations that requires a considerable time for a user to explore. Miller [12] claims that there are 7 ± 2 slots available in human short-term memory. To guide a user to make instant decision, we list maximum seven concepts and their relations in a concept map.

TABLE II: EXAMPLES OF CONCEPT CLUSTERS AND REPRESENTATIVE CONCEPTS

Concept In Query	**Cluster of Related Concepts**	**Representative Concepts**
Holiday	{ car, taxi, vehicle}	{vehicle}
	{hotel, motel, hostel}	{hotel}
	{airplane, flight}	{flight}
Hotel	{car, taxi, vehicle}	{vehicle}
	{motel}	{motel}
	{ hostel}	{hostel}
Weather	{climate}	{climate}
	{wind}	{wind}
	{temperature}	{temperature}
Flight	{aero plane, plane}	{aero plane}
	{car, taxi, vehicle}	{vehicle}
	{holiday}	{holiday}
Computer	{monitor}	{monitor}
	{CPU, hard disk}	{CPU}
	{keyboard, mouse}	{keyboard}
Camera	{review}	{review}
	{feedback}	{feedback}
	{raking, rank}	{rank}
Entertainment	{movie, theater, film}	{movie}
	{casino, sport}	{sport}
	{ nightlife, dance}	{nightlife}

To allow a user to conduct a specific search in terms of the type of services (i.e., operation in SOAP-based Web services or resource in RESTful services), the input and the output, we define a simple query syntax for a user to specify the location of a concept (i.e., input, output and operation) in the query and combine more than one concepts using operators, such as, *and*, *or*, and *exclude*. For example, a search for a service that returns hotels by postal code can be specified as "*postal code ctype:IC & hotel cytpe:IC*", meaning that the user wants to find services with the *hotel* concept delivered in a service and the concept *postal code* as an input.

$$C(x,y) = \begin{cases} \dfrac{|IP(x) \cap IP(y)| * |OP(x) \cap OP(y)|}{|IP(x) \cup IP(y)| * |OP(x) \; U \; OP(y)|} \\ if \; OP(x) \cap OP(y) \neq \emptyset \wedge IP(x) \cap IP(y) \neq \emptyset \\ \dfrac{|IP(x) \cap IP(y)|}{|IP(x) \cup IP(y)|} \\ if \; IP(x) \cap IP(y) \neq \emptyset \wedge \; OP(x) \cap OP(y) = \emptyset \\ \dfrac{|OP(x) \cap OP(y)|}{|OP(x) \; \cup \; OP(y)|} \\ if \; IP(x) \cap IP(y) = \emptyset \; \wedge OP(x) \cap OP(y) \neq \emptyset \\ 0 \\ if \; IP(x) \cap IP(y) = \emptyset \; \wedge OP(x) \cap OP(y) = \emptyset \end{cases} \quad (5)$$

where C(x, y) is the ration of common concepts to the total concepts between service description set x and y; IP(x) is the

Input concept of the service description set and OP(x) is the output concept of the service description set x

Grouping Services. The returned result may contain multiple services. Grouping the returned services makes it easier for a user to find a specific service. Web services can share common attributes, such as the location of services, Quality of Service (QoS) and the number of shared common concepts. However, most of the services do not have the location and QoS information. In our work, we group services using the common concepts between operations or resources (*SK*), inputs (*IP*) and outputs (*OP*). More specifically, our approach groups the retrieved result into four categories with varying restrictive levels: (1) services that share common concepts in *SK, IP and OP*; (2) services that have common concepts in *SK and OP*; (3) services that contain common concepts in *SK and IP*; and (4) services that share common concepts in *SK*. Each group consists of the similar type of service description sets.

Equation (5) measures the similarity between two services. It is the ratio of the common concepts divided by the total number of concepts of the two services. The services in each group are ranked based on the sum of the rank of a concept in services and the rank of a query concept in the service description sets. We compute the rank of a concept extracted a query using Equation (3).

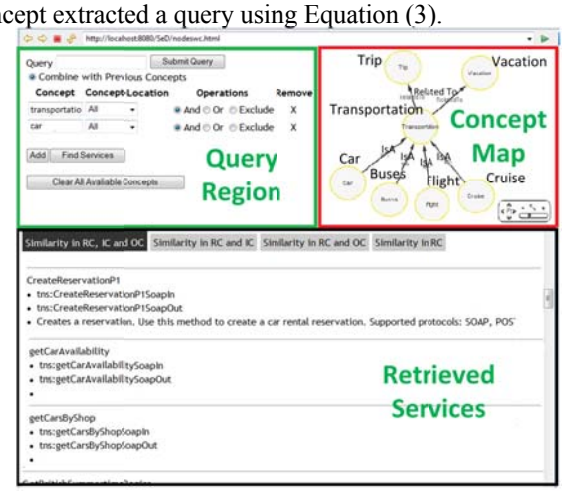

Figure 5: Annotated screenshot of our prototype tool

IV. PROTOTYPE IMPLEMENTATION

We have designed and developed a prototype of our proposed approach. Figure 5 shows the annotated screenshot of the user interface (UI) of our prototype. The user interface is divided into three areas: "Query Region" where a user can submit a query and select concepts and concept locations (i.e., input, output, or a service name); "Concept Map" shows the concepts extracted from a query as well as the related concepts to the query using ConceptNet; and the "Retrieved Service" area lists the retrieved services. As shown in Figure 5, a user submits a query (*i.e., "transportation in Kingston"*) in the "Query Region"; our prototype retrieves the related concepts shown in the "Concept Map" area. The Concept Map displays different types of transportation, such as car, flight and cruise. These concepts serve as an initial point to help a user to craft a more specific query. Once a user finds a relevant concept in the Concept Map, the prototype retrieves services related to the concept and lists them in the "Retrieved Services" area. To ease the user to go through the services, our prototype further divides the services into four categories from the strict matching of the concepts appearing in all three fields (*i.e.,* the input, the output and the name of a service) to a looser matching which matches one concept in one of the fields (*i.e.,* input, output or the name of the service).

V. CASE STUDY

We conduct a case study to evaluate the effectiveness of our approach. The objectives of the case study include: 1) evaluating if our approach can achieve high precision and recall to index services based on the concepts extracted from the service descriptions. We compared the precision and the recall of our approach with a baseline approach which indexes services using textual description (i.e., service documentation) of Web services; and 2) examining the precision of our approach for concept recommendation and query formulation through a user study.

TABLE III : DESCRIPTIVE STATISTICS OF OUR DATA SET

# WSDL documents	550
#WADL documents for RESTful Services	61
#Service Description Set (i.e., Services)	10,100
#Concept Identified	4100

Setup of the Case Study

Table III presents a descriptive statistics of our data set. We gathered 611 service description documents for SOAP based Web Services and Restful Services. Each Web service can contain several operations. Similarly, each RESTful service may include multiple resources. We consider an operation and a resource as a service. The services are such as WebserviceX[1], WebserviceList[2], and xMethods[3]. In total, we identified 4100 concepts from the 10,100 services.

To evaluate the effectiveness of our approach to extract concepts from service, we need to know the number of services relevant to a given concept among the 10,100 services. However, due to the limited resources, we cannot go through each of the services for a given concept. We choose six concepts with different popularity (*i.e.,* high, medium and low): holiday, weather, hotel, flight, computer and camera by manually examining the service description documents. The most popular concept, holiday, is associated with 200 similar service description sets. Flight, hotel and weather are a medium level of popularity and associated with 40, 60 and 40 services respectively and camera by manually examining the service description documents. The most popular concept, holiday, is associated with 200 similar service description sets. Flight, hotel and weather are a medium level of popularity and associated with 40, 60 and 40 services respectively. Camera and computer are the concepts with the lowest popularity and associated with 10 and 8 similar services, respectively. For the sake of performance comparison, we implemented a baseline approach used by the current UDDI registry. The baseline

TABLE IV: COMPARISON OF THE PRECISION, RECALL AND R-PRECISION OF OUR APPROACH WITH A BASELINE APPROACH

Concept in User's Query	Our Approach			Baseline Approach		
	Precision	Recall	R-precision	Precision	Recall	R-precision
Holiday	80%	100%	90%	67%	75%	50%
Hotel	78%	100%	90%	62%	70%	60%
Weather	93%	100%	100%	75%	95%	100%
Flight	82%	100%	90%	69%	85%	60%
Computer	85%	100%	100%	70%	83%	60%
Camera	83%	100%	100%	78%	81%	50%

approach groups services using textual description in service description documentation of Web services and allows users to find services. We setup a user study to evaluate the correctness of the concept recommendation and the assistances in the query formulation. We recruited eight participants to participate in our user study. For each participant, we gave eight different goals, such as booking a hotel room, booking a flight ticket and planning a holiday and ask them to use our prototype to find the services to fulfill the goals. Each participant has the software engineering background and had experience in using Web services.

$$P = \frac{\{relevant\ services\} \cap \{retrieved\ services\}}{\{retrieved\ services\}} \quad (5)$$

$$R = \frac{\{relevant\ services\} \cap \{retrieved\ services\}}{\{relevant\ services\}} \quad (6)$$

$$Rprecision = \frac{\{\#\ of\ relevant\ services\ in\ top\ R\}}{R} \quad (7)$$

Evaluation Criteria

We use precision and recall to measure the performance of our approach to group services using concepts. Precision measures the exactness of the retrieved results and is the ratio of the total number of services correctly retrieved by our approach to the total number of services retrieved as shown in Equation (6). Recall evaluates the completeness of the retrieved result [19]. As shown in Equation (6), recall, is the ratio of the total number of services correctly retrieved to the total number of services existed. The retrieved results could be very large. It could be a tedious job for a user to go through all of them. Therefore, we use R-precision to measure the precision of top R retrieved services [19]. For example, if there are 10 services relevant to the query within the data set and 7 of them are retrieved before the 11th (*i.e.,* R= 11) services, the R-precision is 70%. We choose to compute the R-precision of 10 retrieved result (*i.e.,* R=10) since Silverstein et al. [16] have shown that users mostly look at the first 10 results.

TABLE V: RESULTS OF CONCEPT RECOMMENDATION, QUERY FORMULATION AND PRECISION OF SERVICE RETRIEVAL FROM OUR USER STUDY

Average number of recommended concepts participants found useful	95%
Average number of times a participant reformulates the query	85%
Average precision of service retrieval	98%

To evaluate the correctness of concept recommendation and the assistance in query formulation in the service retrieval process, each participant determines the number of recommended concepts related to their query. Each participant also tracks if he or she reformulated their query using the recommended concepts. We calculate the precision of the retrieved result based on the user's query.

Case Study Results

Table IV shows the results of precision, recall and R-precision of our approach and the baseline approach. Our approach achieves a high recall for all the services and an average precision of 83%. The baseline approach has an average precision of 70% and a recall of 82%. The R-precision of top 10 retrieved results of our approach is 95% in comparison to 63% in the baseline approach. The high precision means that our approach lists relevant results in the beginning of retrieved services. Some of the services in the hotel domain also contain "trip" in their service description documentation; hence our approach indexes the service with both concepts. The recall of our approach is 100%, meaning that our approach can effectively extract concepts from service descriptions. The low recall of the baseline approach is due to the uses of textual description in service description documentation to group services. However, the documentation is not always available for many Web services. The baseline approach uses various terms to index the same services in different concepts. For example, *http://www.holidayguide.co.nz/webservices/frontdesk/holidayguide.asmx.xml* has the "*motel*" and "*holiday*" in the documentation whereas all operations are only related to motel reservation.

Table V shows the result of the number of recommend concepts that participants found useful, the number of times that participants formulate the query and the precision of the service retrieval in the user study. The user study shows that 95% of recommended concepts are related to the user's query and 85% of times a user reformulates the query using the concepts recommended. The average precision of the service retrieval is 98%. Retrieving services using a single concept gives a precision of 83% as shown in Table IV. When a user formulates a query with specific requirement, the precision increase to 98%. Thus, we found that query formulation helps participants to obtain a higher precision result and increases the precision of the service retrieval by 15%.

Threats to Validity

The main threat of our case study that could affect the generalization of the presented results relates to the number

of service description documents analyzed. We have analyzed 611 services from different domains. Nevertheless, further validation of our approach requires an analysis of a larger set of service description documents. The user study is based on eight people; all of them have good knowledge on Web services. The validation of the concept recommendation and query formulation requires a study on diverse group of people.

VI. Related Work

A large body of research on service discovery can be summarized into two main categories: semantic Web approaches and information retrieval (IR) approaches for non-semantic Web services. Semantic Web approaches propose to enhance service descriptions by annotating service description with domain specific ontologies. Semantic Web services are discovered by high level match-making approaches [20]. However non-semantic Web services are more popular and supported by both the industry and development tools. In our approach, we target the discovery of non-semantic Web services.

IR approaches, such as word sense disambiguation, stop-words removal, and stemming, have been used to extract relevant information conveyed within service description documents to index Web services. Sajjanhar et al. [10] propose a SVD-Based algorithm to locate matched services for a given service. This algorithm uses characteristics of singular value decomposition to find relationships among services. However, it only considers service description documentation and cannot reveal the semantic relationship between Web services. Al-Masri et al. [22] proposed a web service crawler and indexed services based on service documentation. Their approach does not filter important concept from the services and result of service retrieval were not ranked. Dong et al. [3] develop a clustering algorithm to group parameter names of operations into semantically meaningful concepts. Then these concepts are used to measure similarity of operations. It relies on names of parameters. Instead, our approach index services based on the semantic concepts and help users formulate queries in order to bridge the semantic gap. Liu et al. [24] and Elgazzar et al. [23] use a text mining techniques to extract features from WSDL documents. These features are then used to cluster services together. The mentioned clustering approached helps to functionally similar services and totally forgets about the user requirement. Similar to their approach we cluster the service description documents. However our approach puts the user on the center of service discovery process helping to reduce the semantic gap by suggesting concepts and narrow down the requirements.

VII. Conclusion

The paper presents an approach to index services based on the concepts shared between services. The proposed approach guides users to formulate queries and group the retrieved results. Our approach can minimize the human effort to find a specific service and bridge the semantic gap between users and service providers by assisting the users in formulating queries and recommending services related to a user's query. We found that our approach is effective in grouping services with a precision of 83% and a recall of 100%. A user study shows that the concept recommendation and query formulation make the user queries more specific and increase the precision of the service retrieval to up to 15%. In future, we plan to conduct a larger case study using a large number of Web services.

References

[1] E. Al-Masri, Q. H. Mahmoud, "Investigating web services on the world wide web," International World Wide Web, pp. 795-804, 2008.

[2] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, "Web Services Architecture", World Wide Web Consortium, Note NOTE-ws-arch-20040211, February 2004.

[3] X. Dong, Halevy A., Madhavan J., Nemes E., and Zhang J. , Similarity Search for Web Services, 30th VLDB Conference, 2004

[4] A.K. Jain, R.C Dubes , Algorithms for clustering data. Prentice-Hall, Englewood Cliffs, 1988.

[5] H. Liu, P. Singh, ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal, To Appear. Volume 22.

[6] M. P. Papazoglou and D. Georgakopoulos. Service Oriented Computing. Communication of ACM, Volume 46, Number 10, October 2003. pp. 25 – 28.

[7] C. D. Manning, H. Schütze, "Foundations of Statistical Natural Language Processing, MIT Press: 1999.

[8] G. A. Miller, "WordNet: A Lexical Database for English", Communications of the ACM Vol. 38, No. 11: 39-41.

[9] M.F. Porter, "An algorithm for suffix stripping, Program", 14(3) pp 130−137, 1980

[10] A. Sajjanhar, J. Hou and Y. Zhang, Algorithm for web services matching, Lecture Notes in Computer Science, 2004.

[11] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. Proceedings of the 2003 Conf. on Empirical Methods in Natural Language Processing, pp 216-223, 2003.

[12] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," Psychol. Rev., vol. 63, pp. 81-97, 1956

[13] R. Alarcón and E. Wilde. "RESTler: crawling RESTful services." In Proceedings of the 19th international conference on WWW '10, pages 1051-1052, New York, USA, 2010. ACM.

[14] R.T. Fielding. "Architectural Styles and The Design of Network-based Software Architectures". PhD thesis, University of California, Irvine, 2000

[15] H. Song, D. Cheng, A. Messer, S. Kalasapur, Web Service discovery using general-purpose search engines, in: IEEE International Conference on Web Services (ICWS), 2007, pp. 265-271.

[16] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. SIGIR Forum, 1999.

[17] Web Service Definition Language (WSDL), www.w3.org/TR/wsdl, last accessed on 25th February, 2012

[18] Web Application Description Language, www.w3.org/Submission/wadl/, last accessed on 25th February, 2012

[19] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel, "Performance measures for information extraction," DARPA Broadcast News Workshop, Herndon, VA, February 1999.

[20] Matthias Klusch, Benedikt Fries, Katia Sycara, "Automated semantic web service discovery with owlsmx," International Conference on Autonomous Agents and Multi-Agent Systems, 2006.

[21] A. Spink, D. Wolfram, B. J. Jansen, and T. Saracevic. Searching the Web: The public and their queries. Journal of the American Society for Information Science and Technology, 52(3):226–234, 2001

[22] E. Al-Masri, Q.H. Mahmoud, "WSCE: A Crawler Engine for Large-Scale Discovery of Web Services", ICWS, 2007

[23] K. Elgazzar, A. E. Hassan, P. Martin, Clustering WSDL Documents to Bootstrap the Discovery of Web Services, ICWS, 2010

[24] Fangfang Liu, Yuliang Shi, Jie Yu, Tianhong Wang, Jingzhe Wu, Measuring Similarity of Web Services Based on WSDL, ICWS, 2010

[25] Q. Yu: Place Semantics into Context: Service Community Discovery from the WSDL Corpu, Paphos, Cyprus, ICSOC 2011

All URLs are last accessed on 20th October, 2012