Web Service Discovery: Dealing With Natural Language Requests and User Preferences

Philippe Larvet
Alcatel-Lucent France
Research and Innovation Centre
Route de Nozay
F-91461 MARCOUSSIS CEDEX
Email : Philippe.Larvet@alcatel-lucent.fr

Nomane Ould Ahmed M'bareck and Samir Tata
GET/INT, CNRS UMR SAMOVAR
9 rue Charles Fourier
91011 Evry Cedex France
Email : {Nomane.Ould_ahmed_mbarek,Samir.Tata}@int-evry.fr

Abstract— **Web services aim at providing standard means to promote interoperability and extensibility among software applications, as well as to allow them to be composed in order to perform more complex operations. An important step towards this goal is service description and discovery. In this paper we propose an approach to deal with this important step. Our approach consists in i) a technique for translating user's requests written in natural language into semantic and formal descriptions, ii) enables users to express their preferences on the parts of their requests, and iii) providing a matching algorithm that can handle semantic description and user preferences.**

I. INTRODUCTION

Web services provide a standard means to promote interoperability and extensibility among software applications, as well as to allow them to be composed in order to perform more complex operations. To do so, several approaches have been developed to describe services, publish these services into registries, discover them using matchmaking algorithms, bind and compose them. The key issue within this process is the way to better describe and discover web services in order to enhance their invocation and composition.

In this context, we propose to integrate and extend existing results to describe and better discover web services. In fact, the aim of this paper is to show how to discover a service fitting with a request expressed in the preferred language of the end-user: the natural language (NL).

One way to reach this objective is to be able to transform the original NL request into a formal need for a "requested service", expressed as a formal description of an in demand service. In addition, we propose to take into account requester's preferences in the process of service discovery. By requester's preferences we mean the degree of importance the requester assigns to the different elements of his query. The user could want to express in his query, for example, the fact he provides some inputs, requires - as a demand - some outputs, and he desires -as a simple wish - some others. In addition, we aim at providing matching mechanisms that focus not only on the inheritance relationship between the requests' concepts and the concepts used by services, but also to consider rhetorical relationships between them.

This paper is organized as follows. In Section II we summarize our approach and present requirements to consider natural language query processing as well as requester's preferences in service discovery are identified. Section III presents our method to process natural language queries in order to formalize user requests into SAWSDL descriptions. Formal semantic description of services, requests and preferences are presented in Section IV. Section V presents our matching algorithm that considers requester's preferences and not only inheritance relation but also rhetorical relations between ontology concepts. Section VI presents related work and approaches that inspired our work. Section VII concludes this paper and presents future work.

II. REQUIREMENTS

In the following we try to draw up some requirements that matching tools should take into account namely : translation of user's requests written in natural language into formal description like SAWSDL, consideration of inheritance and rhetoric relations between concepts during the matching process and finally dealing with the requester's preferences.

A. Natural Language Query Processing

One of the aims of this paper is to show how to discover a service fitting with a end-user NL-expressed request. One way to reach this objective is to transform the original NL request into a formal need for a "requested service", expressed as a formal description of a service. This formal description will be afterwards matched with the content of a service repository, with the help the algorithm we will present in Section V.

The result of the matching process will indicate whether the matching "discovered" service fits or not with the semantics of the original NL request. So, considering in one hand the service repository, containing formal descriptions of services expressed in SAWSDL [15] and, in the other hand, the matching algorithm working with SAWSDL descriptions, a logic way could be to transform the informations contained in the original NL request into a formal SAWSDL description. We will show this process in Section III.

B. Formal Semantic Description

We consider in this paper that Web services and service requests are defined using ontology-based descriptions. Indeed several works(e.g. [12], [4], [16]) have underlined many limitations of syntactical-based descriptions and needs of using ontology-based descriptions. Ontology is specification of a conceptualization of a knowledge and specified domain of interest. It controls the concepts and their properties and relations within a structure. Two types of concept relations are defined: inheritance and rhetoric. Inheritance is a directed relationship between two concepts (parent and child) in which the child concept inherits the properties of the parent concept. Only the properties which are different need to be described in the child concept. Rhetorical relation is a directed relationship between two concepts in which one concept has a property defined by another one.

Describing Web services using ontology will enhance service discovery. The "quality" of that discovery depends on the used algorithms and techniques. In order to improve the discovery, the matching algorithms have to take into account rhetorical and inheritance relations. In fact, a majority of existing algorithms only deal with inheritance. This allows getting a certain degree of similarity between a child concept and its parent, since the child inherits the properties of its parent. For example there is a certain degree of similarity, which has to be defined, between the Visa and Payment concepts. Matching algorithms should in addition take into account the rhetorical relation between concepts.

C. Dealing with the requester's preferences

The last but not least requirement we deal with in this paper concerns the requester's preferences that have to be associated with requests. Indeed, when a requester for Web Services formulates his request, he should get the means to specify that an element in his request is more important than another one. Moreover, matching algorithm must be able to take these preferences into account during the matching rather than during the ranking of matching results as it is done by many other approaches. For example, if more than one matching Web Service is found, WSMX [3] selects the most suitable one based on preferences provided by the service requester. Here we want to include preferences of the requester into the matching process.

III. Natural Language Query Processing

As exposed in Section II, the aim of the natural language (NL) query processing is to derive a formal description of a "need for a requested service" from a request expressed in the preferred language of the end-user: the natural language. The obtained formal description, expressed in SAWSDL, will be afterwards matched with the content of a service repository, with the help of the algorithm presented in Section V.

A. Natural Language Processing Strategy

Natural language processing could require a proper *knowledge base* describing semantically the service repository where the concepts are, internally or externally, linked to *lexical resources* pertaining to one or more languages (English, French, Italian, etc.) in order to allow the recognition of named entities within the text. Different approaches in literature propose the modeling of such a knowledge base at different detail levels and according to different strategies depending on the level of NL processing required: parse trees, dependency graphs, composed ontologies [8] or simply named entities recognition.

Key elements of this semantic repository are *goals*, *preconditions*, *effects* and *quality of service (QoS)* of operations exposed by the services, as well as input and output *semantic types of parameters* defined in their interfaces.

The principles of our process is to extract from the NL request the main need of the requested service, the inputs and outputs of this service, and to put these elements in correspondence with external adequate ontologies in order to generate a pertinent SAWSDL description of the requested service.

The proposed process can be divided into four steps: Extracting main need of service, Parameter extraction, Parameter priorization and Service identification.

Extracting main need of service consists in processing the NL request in order to extract the need of service. A good strategy is here to use a simple lexical/grammatical analysis of the NL request, in order to extract the main "request concepts" as the canonical form of pertinent used terms, after elimination of non-pertinent words. The pertinent kept concepts are afterwards searched in the ontologies associated to the semantic descriptions of services in order to help the service discovery mechanism.

Another possible, but more complex, strategy is to derive an ontological representation of the request. Different approaches can be used in order to achieve this goal: in [17] the user "query" is translated into an *ontological formula* that consists in a set of Description Logic predicates; in [17], [11], [6] the recognition of named entities offers *semantic coordinates* capturing the main concepts involved in the request and then a coarse grained analysis of the request (Subject-Verb-Object) is produced in order to intercept the logical relations between concepts.

From the textual request the semantic type of parameters as well as their values, if present, can be extracted; this task can be performed only if a proper recognizer, capable of extracting specific information, is used.

With the help of well-adapted ontologies, a priorization of the parameters is possible, in order to determine the user's preferences. For example, if the request contains " I want this service preferably in French but imperatively in video."

Service identification consists mainly in the identification of the service(s) that best match the user's needs expressed as the pertinent kept concepts extracted from the NL request in step 1. The matching algorithm described in Section V is used at this step.

B. Requested Service Specification

The specification of the requested service, coming from the request analysis, indicates the goal to be reached by the desired service. There are several different approaches in literature for the specification of goals. These include work on semantic web services and AI-planning. For example, Fujii and Suda in [7] use semantic graphs derived from natural language descriptions.

For the approach we chose to present in this paper, we assume that the service requester is an end-user, who represents the goal and the properties of the service he requires through the simplest way, via a NL request. For example, let us consider this sample of such end-user requests: *"A German version of a text"*.

A lexical analysis of this request is followed by a semantic analysis, in order to extract a set of requirements that can be expressed under the form of semantic descriptions of the in demand service.

The first result of the lexical analysis is a list of pertinent words, with their synonyms or canonical form if necessary (the non-pertinent words are ignored). The pertinent words are "German", "version" and "text". Using the ontology we deduce that "version" is synonym of "translation". In the second result, the semantic analysis module adds specific semantic information, pertinent for a given specific domain of application.

"A German version of a text", is "understood" by the analyzer as a "need" for a Translation service, and can be expressed under the form of the call of a Translate function with the parameters "text" and "German": *translate(text, German)*

As a consequence, through this simple request example, we assume that within the context of our approach, the specification of the required service, represented by a list of "needs of services"/concepts, can be finally understood as a couple $\{$*function, parameters*$\}$[1].

A couple of $\{$*function, parameters*$\}$ is transformed into a service request as follows. The name of the function is considered as the name of requested operation. The parameters are transformed into inputs of operation. The outputs of the operation depend on the request. In the following section, we show how we describe formally a service request.

IV. Formal Semantic Description

The contribution given in our approach consists in adding information to requests described in SAWSDL [15], thanks to *WSDL 2.0* [1] extensibility, in order to enable requester to specify his preferences and proposes matching algorithm that takes into account the additional information given in the previous step.

A. Preference ontology

One of the objectives of our approach is to enable users to specify their preferences on elements in requests. In this Section, we explain how preferences are expressed. In fact, we assume that users can either refers to an own preferences ontology or to a public one. In the later case we define a function that associates to each preferences ontology concept, a value between 0 and 1. For every preference ontology, we suppose that there are two basic concepts *Mandatory* and *Null*. *Mandatory* is the concept of the highest level of preference (i.e. preference = 1). *Null* is the concept with the lowest level of preference (preference = 0).

To simplify, the preference ontology presently used in our prototype contains only an enumeration class, called **level**. The class **level** has the property **value** which specifies a floating value associated to each **level**. **level** is one of the classes: *Mandatory*, *High*, *Medium*, *Low* and *Null*. Preference values associated to them are respectively 1, 0.75, 0.5, 0.25 and 0. This ontology could be extended to include other types of preferences.

B. SAWSDL extension for request description

Semantic Annotations for WSDL (SAWSDL) [15] defines how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. The extension attributes defined in this specification fit within the WSDL 2.0 extensibility framework [15]. To meet this end, SAWSDL defines a new specific namespace *sawsdl* and adds an extension attribute, named *modelReference*, to specify the association between WSDL components and concepts in some semantic model. In the following, we give the formal description using SAWSDL og the request *"A German version of a text"*.

```
<wsdl:operation name="translate" >
 <wsdl:input element="text"
   sawsdl: modelReference="http://w3/ontology/language#French"/>
 <wsdl:output element="text"
   sawsdl: modelReference="http://w3/ontology/language#German"/>
</wsdl:operation>
```

Following the example of WSDL extension, we extend SAWSDL to enable the specification of user's preferences. The extension is done by adding the attribute *preference*. It is used to associate an importance degree to WSDL operations, inputs, outputs, preconditions and effects.

In order to assign preferences to a simple type we use the attribute **preference** with name space *psawsdl*. Here for example the preference assigned to the input text is *High* i.e. it is important for the requester that the service translates French texts.

```
<wsdl:operation name="translate" >
 <wsdl:input element="text"
  sawsdl: modelReference="http://w3/ontology/language#French"
  psawsdl: preference="http://w3/ontology/preference#High"/>
 <wsdl:output element="text"
  sawsdl: modelReference="http://w3/ontology/language#German"
  psawsdl: preference="http://w3/ontology/preference#Medium"/>
</wsdl:operation>
```

V. Bringing Web services and users together

We present our matching algorithm that evaluates the similarity between user request and Web service advertisement both described in SAWSDL. This consists of matching two

[1]For the general case, the result is a list of couples $\{$*function, parameters*$\}$ where a composition process is needed to deal with the analyzed request.

SAWSDL files. This section is organized in three sub-sections. The first one details concept matching. The second shows how preferences impact this matching. And, the last subsection shows the matching algorithm.

A. Concepts matching

Prior to explaining the matching, let us give some definition. Assume that concept X is a direct or indirect subclass of concept Y. $PropNum(X)$ function denotes the number of properties of concept X and $path(X,Y)$ function denotes the set of classes (X and Y included) that are in the path connecting X to Y. $RhetoProp(X,Y)$ function determines the number of rhetorical relations between X and Y. It should be noted that concept and class mean the same thing here. We denote by X the concept in the request and by Y the concept in the advertisement service. $MatchCpt(X,Y)$ function matches a concept in a request against a concept in an advertisement. Now we give the exact definition of $MatchCpt(X,Y)$ function:

- $MatchCpt(X,Y) = MatchCpt(Y,X) = 1$; if X and Y are the same concept or declared as equivalent classes.
- X is subclass of Y and thus, X inherits all Y properties. Z is the first common super-class of X and Y. It is important to distinguish whether X and Y are outputs or not.
  - X and Y are outputs:
    $MatchCpt(Y,X) = 1$
    and
    $$MatchCpt(X,Y) = \frac{\sum_{n \in path(Y,Z)} PropNum(n)}{\sum_{n \in path(X,Z)} PropNum(n)}$$
  - X and Y are not outputs:
    $Match(X,Y) = 1$
    and $Match(Y,X) = \dfrac{\sum_{n \in path(Y,Z)} PropertyNum(n)}{\sum_{n \in path(X,Z)} PropertyNumebr(n)}$
- X and Y are two subclasses of class Z. Thus, they inherit all Z properties. The similarity between X and Y depends on the number of the common properties they inherit from Z and the rhetorical relation between X and Y and their own properties. In this case the similarity is symmetric.
  $rhProp = RhetoProp(X,Y)$
  and
  $$sum1 = \sum_{n \in path(X,Z)} PropNum(n) - rhProp$$
  and
  $$sum2 = \sum_{n \in path(Y,Z)} PropNum(n) - rhProp$$
  and
  $MatchCpt(X,Y) = MatchCpt(Y,X)$
  and
  $MatchCpt(X,Y) = \frac{PropNum(Z)+rhProp}{sum1+sum2-PropNum(Z)}$
- X and Y have a rhetoric relation between them and do not have a common super-class.
  $MatchCpt(X,Y) = MatchCpt(Y,X) = \frac{RhetoProp(X,Y)}{PropNum(X)+PropNum(Y)-2*RhetoProp(X,Y)}$
- for other cases the similarity between X and Y are null.
  $MatchCpt(X,Y) = MatchCpt(Y,X) = 0$

B. Impact of user's preferences on the matching

The second part of the algorithm consists of examining the preference included in a user's requester. Our algorithm uses $prefValue(X)$ function that returns the numeric preference value associated with element X. The function that calculates this value is called $MatchPref()$. It takes as parameters two elements (input,output) and returns the final matching associated with them (see Algorithm 1). The first parameter of $MatchCpt()$ is an element from the request and the second parameter is from the advertised service.

Algorithm 1 $MatchPref(rqt, adv)$

```
1:  $\alpha = MatchCpt(rqt, adv)$; $x = prefValue(rqt)$;
2:  if ($\alpha == 0$) then
3:      return $0$;
4:  else
5:      if ($x == 1$) then
6:          return $\alpha$;
7:      else
8:          return $(\alpha - 1)x + 1$;
9:      end if
10: end if
```

We assume that if the importance assigned to a request is mandatory (line 3 in Algorithm 1), the function keeps the exact value of the matching of concepts *rqt* and *adv*. When the preference assigned to the request is not *mandatory*, the algorithm increases the value of the matching (line 4). In fact, when there is an element in the request, which is not important for the user, we consider that this element can be replaced by another one that does not match exactly with it, since it is no too important for the user.

C. Computing the global matching result

In the previous two steps (Section V-A and V-B) we explained how to compute the matching of two concepts taking into account the preferences associated with the request. Once the matching of each pair of concepts is done, we compute the global matching as the average of these pair matching results. Line 26 in Algorithm 2 shows the computing of the global matching result of two operations.

The matching result between two operations is computed from the matching of concepts of inputs, outputs, preconditions, effects, and operations as well as the matching between user's contexts and service's capacities. The value associated with each of these elements in the final matching is computed from the result of concept matching (computed by $MatchCpt()$ function) and the preference associated with the request (computed by $prefValue()$ function).

Algorithm 2 defines the matching of two operations. To ease the reading, we do not deal with preconditions and effects

Algorithm 2 $MatchOp(rqtOp, advOp)$

```
1: $i_1 = Input(rqtOp)$;
2: $i_2 = Input(advOp)$;
3: $o_1 = Output(rqtOp)$;
4: $o_2 = Output(advOp)$;
5: $sumIn = 0$;
6: $numIn = 0$;
7: $sumOut = 0$;
8: $numOut = 0$;
9: $opMatch = 0$;
10: for ($e_1 \in Elts(i_1)\ and\ e_2 \in Elts(i_2)$) do
11:     $sumIn = sumIn + MatchPref(e_1, e_2)$;
12:     $numIn + +$;
13: end for
14: for ($e_1 \in Elts(o_1)\ and\ e_2 \in Elts(o_2)$) do
15:     $sumOut = sumOut + MatchPref(e_1, e_2)$;
16:     $numOut + +$;
17: end for
18: $inMatch = \frac{sumIn}{numIn}$; $outMatch = \frac{sumOut}{numOut}$
    return $\frac{inMatch+outmatch+opMatch}{3}$
```

matching and assume that elements of input/output are ordered and have the same number. The reader can easily deduce the general case.

VI. Related work

In [19], the authors introduce a corpus-based method to facilitate the matchmaking of WSDL files. This method uses the Web as an effective corpus and the matchmaking is done by computing the similarity between all pairs of elements from a request and WSDL advertisements. In order to calculate elements similarity they compute the similarity of words composing elements, which is done by taking into account the context in which words appear in documents. This approach has some drawbacks, some of them are addressed by the authors. First, by using the Web as corpus, one has the same corpus for different domains. In fact, for example the same word can be find twice in the same context with different meaning. Second, the system works correctly only for elements named in a single word. Third, the approach does not address any of our requirements given above in Section II

A step-by-step matching algorithm that can be used to determine the semantic similarity of two Web Services is introduced in [18]. The algorithm works for services described in OWL-S [14] and is composed of four steps. If one step fail, the other ones are not required. The first step consists of matching service *Profiles*. The second one is *input* matching. The third one consists of *output* matching. Finally if the result of these three former steps is not *Fail*, then the matching of the *non-functional properties* is done. Like [13], the algorithm defines five degrees of matching *Exact*, *PlugIn*, *Subsume*, *Intersection*, and *Fail*.The algorithm proposed in [18] does not take into account rhetoric relations between concepts during the matching as well as request preferences. The algorithm assume that the request is written on OWL-S and consequently it does not deal with the translation of the request from natural language into a formal description (our II-A requirement)

In [10] the authors propose an ontology service description language (*OSDL*) and an ontology service query language (*OSQL*). OSDL enables to annotate a WSDL file by mapping inputs, and outputs to ontology concepts. So it is similar to WSDL-S [2]. The *OSQL* is very similar to the *OSDL*. The difference is that *OSQL* describes the semantic of requests and *OSDL* describes the semantic of advertised services. This algorithm computes similarity between concepts by taking into account not only the inheritance relation but also the rhetoric relation between concepts. Two concepts *X* and *Y* are considered perfectly similar if (1) *X* = *Y*; they have the same information. Thus, *X* is similar to *Y*. (2) *X* is subclass of *Y*. In this case *X* is considered as similar to *Y* but not the other way around. (3) *X* is the property of *Y*. So *Y* is considered as perfectly similar to *X* but not the other way around. (4) *X* is the property of some class *Z*, while *Y* is a subclass of *Z*. Then *Y* is said perfectly similar to *X* but not the other way around.

Consequently this approach deals with the matching of rhetorical and inheritance relation and thus it address our second requirement (see II-B). However this approach does not address our first and third requirement (see our II-C and II-A requirements)

VII. Conclusion and perspectives

We presented in this paper a novel approach to enhance web service discovery based on, among others, requesters' preferences. First of all, we presented a technique for translating user's requests written in natural language into semantic and formal descriptions. Then we proposed a way to express preferences of request elements. Finally, we introduced a matching algorithm that takes into account these preferences in the discovery algorithm. Inheritance and rhetoric relations between concepts are used in the concepts matching. The result of that matching is modulated according to the request preferences.

Currently, we are developing a semantic-driven, capacity-aware, and preference-aware matchmaker that we will integrate into our CoopFlow framework [5]. This framework is already provided with a behavior-based matchmaker of composite Web services and service-oriented business processes and workflows [9].

References

[1] WSDL 2.0 Home Page. http://www.w3.org/TR/wsdl20/, 2006.

[2] WSDL-S Home Page. http://www.w3.org/Submission/WSDL-S/, 2005.

[3] WSMX Home Page. http://www.wsmx.org/, 2005.

[4] Boualem Benatallah, Mohand-Said Hacid, Christophe Rey, and Farouk Toumani. Request rewriting-based web service discovery. In *International Semantic Web Conference*, pages 242–257, Sanibel Island, Florida, USA, 2003.

[5] Issam Chebbi, Schahram Dustdar, and Samir Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data Knowl. Eng.*, 56(2):139–173, 2006.

[6] Kurt Englmeier, Javier Pereira, and Josiane Mothe. Choreography of web services based on natural language storybooks. In Mark S. Fox and Bruce Spencer, editors, *ICEC*, pages 132–138. ACM, 2006.

[7] Keita Fujii and Tatsuya Suda. Semantics-based dynamic service composition. *IEEE Journal on Selected Areas in Communications*, 23(12):2361–2372, December 2005.

[8] Johannes Heinecke and Farouk Toumani. A natural language mediation system for e-commerce applications: an ontology based approach. In *Workshop on Human Language Technology for the Semantic Web and Web Services, 2nd International Semantic Web Conference*, Sanibel Island, Florida, October 2003.

[9] Kais Klai, Nomane Ould Ahmed M'Bareck, and Samir Tata. Behavioral technique for workflow abstraction and matching. In Schahram Dustdar, José Luiz Fiadeiro, and Amit P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 477–483. Springer, 2006.

[10] Li Kuang, ShuiGuang Deng, Ying Li, Wei Shi, and Zhaohui Wu. Exploring semantic technologies in service matchmaking. In *Third European Conference on Web Services (ECOWS'05)*, pages 226–234, Växjö, Sweden, 2005.

[11] Javier Pereira Kurt Englmeier and Josiane Mothe. Choreography of web services based on natural language storybooks. In *Workshop on Information Integration on the Web in conj. with 15th international conference on World Wide Web, WWW 2006*, Edinburgh, Scotland, UK, May 2006.

[12] Ke Li, Kunal Verma, Ranjit Mulye, Reiman Rabbani, John A. Miller, and Amit P. Sheth. Designing semantic web processes: The WSDL-S approach. In Jorge Cardoso and Amit P. Sheth, editors, *Semantic Web Services, Processes and Applications*, pages 161–192. Springer, 2006.

[13] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th international conference on World Wide Web (WWW'03)*, pages 331–339, Budapest, Hungary, 2003. ACM Press.

[14] David L. Martin, Massimo Paolucci, Sheila A. McIlraith, Mark H. Burstein, Drew V. McDermott, Deborah L. McGuinness, Bijan Parsia, Terry R. Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katia P. Sycara. Bringing semantics to web services: The owl-s approach. In *Semantic Web Services and Web Process Composition Workshop*, volume 3387 of *Lecture Notes in Computer Science*, pages 26–42, San Diego, CA, USA, July 2004.

[15] Semantic Annotations for WSDL Home Page. http://www.w3.org/TR/sawsdl-guide/, 2006.

[16] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, pages 447–461, Stanford, CA, USA, 2001.

[17] Ian Wakeman, David Weir, Bill Keller, Julie Weeds, and Tim Owen. Composing grid services through natural language. In *Workshop on Ubiquitous Computing and e-Research (4th UK UbiNet Workshop)*, May 2005.

[18] Yonglei Yao, Sen Su, and Fangchun Yang. Service matching based on semantic descriptions. In *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW)*, page 126, Guadeloupe, French Caribbean, 2006.

[19] Ziming Zhuang, Prasenjit Mitra, and Anuj Jaiswal. Corpus-based web services matchmaking. In *Workshop on Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing*, pages 46–52, Pittsburgh, Pennsylvania, 2005.