International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 3, Issue 1 (Jan-Feb 2015), PP. 35-37

A LITERATURE SURVEY ON INFORMATION EXTRACTION BY PRIORITIZING CALLS

Kalaiselvi. K[1], Dr. P. Uma Maheswari Ph.D[2]
Department of Computer Science & Engineering,
Info Institute of Engineering,
Coimbatore, India.
[1]kalai.cbe@gmail.com, [2]dr.umasundar@gmail.com

Abstract- The Application Programming Interface restricts the types of queries that the Web service can answer. For instance, a Web service might provide a method that returns the books of a given author in fast manner, but it might not provide a method that returns the authors of a given book. If the user asks for the author of some specific book, then the Web service cannot be called – even though the underlying database might have the preferred piece of information, this scenario is called asymmetry. This asymmetry is particularly problematic if the service is used in a Web service orchestration system. In this survey, we propose to use on-the-fly information extraction (IE).IE used to collect values, and then the value can be used as parameter Bindings for the Web service. This survey shows how the information extraction can be integrated into a Web service orchestration system. The proposed approach is fully implemented in a prototype called Search Using Services and Information Extraction (SUSIE). Real-life data and services are used to demonstrate the practical viability and good performance of our approach.

Keywords- limited access patterns to relations, complete answers to queries, query stability.

I. INTRODUCTION

A. *Web Services*

Generally a Web service[9] is an interface that provides accesses to an integrated back-end database. There is an increasing number of Web services that provide a capital of information. There are Web services about books are isbndb.org, library thing. com, Amazon, Abe Books, and so on, and then about movies are api.internetvideoarchive.com, about music's are musicbrainz.org, lastfm.com, and about a large variety of other topics.

B. *How Web Services Work?*

For instance, the site musicbrainz.org offers a service for accessing its database about music albums. The Web service defines functions that can be called remotely musicbrainz.org offers the function getSongs, which takes a singer as input parameter and delivers the songs by that singer as output. If the user wants to know all songs by Leonard Cohen, she can call getSongs with input as Leonard Cohen. The output will contain the songs Hallelujah, Suzanne, etc.

C. *Web Services vs. Keyword Based search on web*

Web services play a vital part in the trend towards data centric applications on the Web. Unlike Web search engines, Web services deliver crisp answers to queries[10]. This allows the user to recover answers to a query without having to read through several result pages. Web services can also be used to answer precise conjunctive queries, which would require several searches on the Web and joins across them, if done manually with a search engine. The results of Web services are machine-readable, which allows query answering systems to provide to complex user demands by orchestrating the services. These are advantages that Web services offer over keyword based Web search.

D. *Binding Patterns of the Web Service Functions*

Web services allow querying remote databases. However, the queries have to follow the binding patterns of the Web service functions. Binding patterns [11][13] should be achieved by providing values for mandatory input parameters before the function can be called. In our example of musicbrainz, the function **get Songs** can only be called if a singer is provided with enough information. Thus, it is possible to ask for the songs of a given singer, but it is not always possible to ask for the singers of a given song.

E. *Web service asymmetry*

On the client side, this is a highly difficult limitation, in which the data may be available, but cannot be queried in the preferred way [12][14]. If the user wants to recognize, e.g., who sang Hallelujah, then the Web service cannot be used to answer this question, even though the database contains the preferred information. This control is not due to missing data, but a design choice of the Web service owner, who wants to prevent external users from extracting and downloading large fractions of its back-end database.

F. *Web Service Composition*

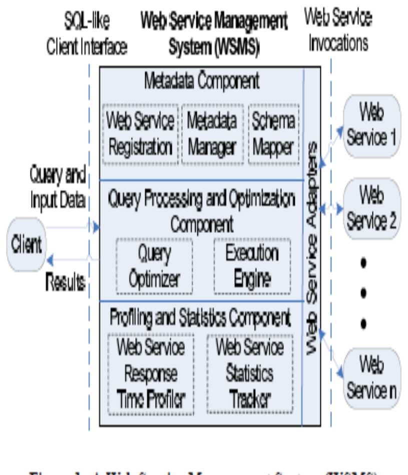

Figure 1: A Web Service Management System (WSMS)

Web Service Management System (WSMS) enables querying multiple web services in a transparent and integrated fashion. This paper [10] handles a first basic WSMS problem called query optimization for Select-Project-Join queries spanning multiple web services. The contribution of an algorithm for arranging a query's web service calls into a pipelined execution plan that optimally exploits parallelism among web services to minimize the query's total running time.

G. Using Web Services in Knowledge Base

The overall architecture specified in the paper [8] is illustrated in Figure 2.The system uses the existing YAGO ontology, which consists of 2 million entities and 20 million facts extracted from various encyclopedic Web sources. In addition, we extended the knowledge with a built-in collection of function definitions for the following Web services: Music Brainz, LastFM, LibraryThing, ISBNdb, AbeBooks, and IVA (Internet Video Archive). In this paper [8] author envisioned long-term usage, the function definitions would either be automatically acquired from a Web-service broker/repository or they could be semi-automatically generated by a tool.

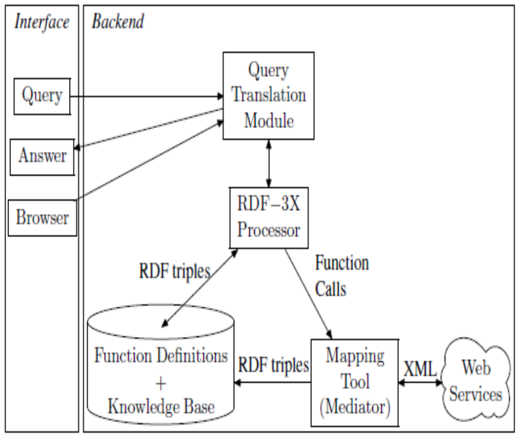

Figure 2:- System architecture of the ANGIE.

II. EXISTING SYSTEM

[1] In 2007 F. M. Suchanek, G. Kasneci, and G. Weikum, worked on "YAGO: A Core of Semantic Knowledge". This paper explains that, if the user asks for the singer of the song Hallelujah, then we can use a semantic knowledge base such as YAGO to get a list of singers. Then we call getSongs with every singer, and remember those for which the Web service returns Hallelujah. The first limitation in this system is runtime, with Web service calls taking up to 1 second to complete. Trying out thousands of singers that a knowledge base such as YAGO contains could easily take hours. The second limitation is the data provider itself, which most likely restricts violent querying from the same IP address.

[2]In 1995 A. Rajaraman, Y. Sagiv, and J.D.Ullman, worked on "Answering queries using templates with binding patterns".Rewriting the initial query into a set of queries to be executed over the given views considered in this scheme. The method show that for a conjunctive query over a global schema and a set of views over the same schema, shaping whether there exists a conjunctive query plan over the views that are equivalent to the original query in NP-hard.This rewriting strategy assumes that the views are complete (i.e., contain all the tuples in their definition).

The main contribution of this paper is obtaining the cheapest solution under certain reasonable cost metrics.

Thus, these works will produce pipelines of calls of an unbound length in order to obtain the maximal number of answer .This is infeasible and challenging task in the context of Web services. the assumption on views are unrealistic in our setting with Web services, where sources may overlap or complement each other but are usually incomplete.

[3] In 2004 S. Kambhampati, E. Lambrecht, U. Nambiar, Z. Nie, and G. Senthil, worked on "Optimizing recursive information gathering plans in this paper authors describe two optimization techniques that are specially tailored for information gathering. The first optimization technique is a greedy minimization algorithm that minimizes an information gathering plan by removing redundant and overlapping information sources without loss of completeness. Then these papers discuss a set of heuristics that guide the greedy minimization algorithm so as to remove costlier information sources first.

The main contribution of this paper is consider end-to-end the issues of redundancy elimination and optimization in recursive information gathering plan but it also reducing the number of accesses

[4]In 2006 W. Wu, A. Doan, and C. T. Yu, worked on "Webiq: Learning from the web to match deep-web query interfaces". This paper aims to match the schema of two Deep Web forms. This approach finds instances for Deep Web attributes by querying the surface Web and then validating the instances through original Web form.

The advantage of this method is avoiding unnecessary requests. Deep Web does not address the prioritization of inverse functions in execution plans is very big challenge.

[5]In 2008 A. Cal`ı and D. Martinenghi, worked on "Querying data under access limitations," This scheme represent an optimization technique for conjunctive queries that produces a query plan that:

(1) minimizes the number of accesses according to a strong notion of minimality; (2) excludes all sources that are not relevant for the query. The major role is decrease of the number of accesses in a large number of cases.

[6]In 2010 N. Preda, G. Kasneci, F. M. Suchanek, T. Neumann, W. Yuan, and G. Weikum, worked on "Active Knowledge : In this paper Dynamically Enriching RDF Knowledge Bases by Web Services (ANGIE)[8]," answers queries using views as surrogates for Web services, and it imposes an upper bound on the number of function calls.

This strategy is prioritizing function calls that are more likely to deliver answers is major role of this method but it also Reduces the number of sub-queries in a bottom-up evaluation.

[7]In 2011 M. Benedikt, G. Gottlob, and P. Senellart, worked on "Determining relevance of accesses at runtime, "In this process, authors relate dynamic relevance of an access to query containment under access limitations and characterize the complexity of this problem;

The advantage of this scheme is reducing the number of accesses. The major drawback in this paper is, the computation of maximal results is not considered also the guessing accesses are not eliminated.

III. PROPOSED SYSTEM

[20]The objective is to answer queries by using only function calls. The main goal of this paper, in contrast, is to compute the largest number of answers using the given budget of calls. Hence, we have to prioritize calls that are likely to lead to an answer. This is different from the problem of join ordering in previous work.

Information extraction (IE)[15][17] is concerned with extracting structured data from documents. This method should suffer from the inherent imprecision of the extraction process. Generally the extracted data is way too noisy to allow direct querying. This limitation should be overcome by SUSIE using Information extraction solely for finding candidate entities of interest and feeding these as inputs into Web service calls. Named Entity Recognition (NER) approaches aim to detect interesting entities in text documents. This scheme can be used to generate candidates for SUSIE.

The approach discussed in this paper matches noun phrases against the names of entities that are registered in a knowledge base a simple but effective technique that circumvents the noise in learning-based Named Entity Recognition(NER)[18][19] techniques. For SUSIE, we have developed judiciously customized methods along these lines. These are not limited to lists and tables, but detect arbitrary repetitive structures that could contain candidates.

Alternative IE[16] methods such as Wrapper Induction, fact extraction, or entity extraction could be also considered, but they are not practical in our scenario as they require training data and, thus, human supervision.

IE[20] has the following contribution

1. The proposed approach provide solution to the problem of Web service asymmetry, where input values for the Web services are extracted on-the-fly from Web pages found by keyword queries.
2. An alteration to the standard Data log evaluation procedure is that prioritizes inverse functions over infinite call chains.
3. An evaluation of our approach with the APIs of real Web services, improve the performance of a real-world query answering system.

Table 1:-Comparative Study on Existing vs. Proposed System

TECHNIQUE	**EXISTING**	**PROPOSED**
Scheme	Binding Pattern	Information Extraction
Query Type	Querying For Single Argument	Querying For Multiple Argument
Execution Plan	Ordered Join Plans	Ordered Sequences Of Calls
Aim	Maximal Number Of Query Answers	Maximal Contained Rewriting
Query Ordering	Join Ordering	Prioritize Calls

IV. CONCLUSION

This survey shows validity of SUSIE approach on real data sets. We believe that the good looks of our approach lies in the fruitful symbiosis of information extraction and Web services, which each mitigate the weaknesses of the other. Our current implementation uses naive information extraction algorithms that serve mainly as a proof of concept.

V. SCOPE AND FUTURE ENHANCEMENT

Future work will explore new algorithms that could step in and approaches that aims to automize the discovery of new Web services and their integration into the system.

REFERENCES

[1] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A Core of Semantic Knowledge," in WWW, 2007.

[2] Rajaraman, Y. Sagiv, and J. D. Ullman, "Answering queries using templates with binding patterns," in PODS, 1995.

[3] S. Kambhampati, E. Lambrecht, U. Nambiar, Z. Nie, and G. Senthil, "Optimizing recursive information gathering plans in EMERAC," J. Intell. Inf. Syst., 2004.

[4] W. Wu, A. Doan, and C. T. Yu, "Webiq: Learning from the web to match deep-web query interfaces," in ICDE, 2006.

[5] Cal`ı and D. Martinenghi, "Querying data under access limitations," in ICDE, 2008.

[6] N. Preda, G. Kasneci, F. M. Suchanek, T. Neumann, W. Yuan, and G. Weikum, "Active Knowledge: Dynamically Enriching RDF Knowledge Bases by Web Services. (ANGIE)," in SIGMOD, 2010.

[7] M. Benedikt, G. Gottlob, and P. Senellart, "Determining relevance of accesses atruntime," in PODS, 2011.

[8] N. Preda, G. Kasneci, F. M. Suchanek, T. Neumann, W. Yuan, and G. Weikum, "Active Knowledge : Dynamically Enriching RDF Knowledge Bases by Web Services. (ANGIE)," in SIGMOD, 2010.

[9] Deutsch, L. Sui, and V. Vianu, Specification and verification of data-driven web services," in PODS, 2004.

[10] U. Srivastava, K. Munagala, J. Widom, and R. Motwani, "Query optimization over web services," in VLDB, 2006.

[11] S. Thakkar, J. L. Ambite, and C. A. Knoblock, "Composing, optimizing,and executing plans for bioinformatics web services," VLDB J., 2005.

[12] K. Q. Pu, V. Hristidis, and N. Koudas, "Syntactic rule based approach to Web service composition," in ICDE, 2006.

[13] M. Benedikt, G. Gottlob, and P. Senellart, "Determining relevance of accesses at runtime," in PODS, 2011.

[14] M. Benedikt, P. Bourhis, and C. Ley, "Querying schemas with access restrictions," PVLDB, 2012.

[15] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open Information Extraction from the Web," in IJCAI, 2007.

[16] F. M. Suchanek, M. Sozio, and G. Weikum, "SOFIE: A Self-Organizing Framework for Information Extraction," in WWW, 2009.

[17] N. Kushmerick, "Wrapper induction for information extraction," Ph.D. dissertation, U.Washington, 1997.

[18] J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma, "2d conditional random fields for web information extraction," in ICML. ACM, 2005.

[19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in ICML. Morgan Kaufmann Publishers Inc., 2001.

[20] Nicoleta Preda1, Fabian Suchanek2, Wenjun Yuan3, Gerhard Weikum2," Susie: Search Using Servicesand Information Extraction" In Ieee Transactions On Knowledge And Data Engineering Year 2013.