Volume 5, Issue 2, February 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Web Services Using Inverse Function

Shetye A. N.*
M.E. Student
Comuter Engineering
Terna College of Engg. Nerul, India

Dhumal R. A.
Assistant Professor
Computer Engineering.
Terna College of Engg, Nerul, India

Abstract— ***There are many web services available in market which provides a particular service in a very efficient manner. The API of a Web service restricts the types of queries that the service can answer. For example, a Web service might provide a method that returns the songs of a given singer, but it might not provide a method that returns the singers of a given song. If the user asks for the singer of some specific song, then the Web service cannot be called – even though the underlying database might have all the desired of information. This asymmetry is particularly problematic if the service is used in a Web service orchestration system.***

The comparative study of web service analyzes various parameters which has an impact on performance of web service. Web service along with inverse function uses on-the-fly information extraction to collect values .That values can be used as parameter bindings for inverse function of the Web service. Again those extracted parameter values are validated against a web service. This paper has presented some facts with real-life data and services to demonstrate the practical viability and good performance of this approach of web service. So by following this approach, it is possible to achieve query optimization, recursive calls for query and to process union of conjunctive queries under integrity constraint with access limitation.

Keywords— Inverse Function, Asymmetry, Information Extraction, Web Service Orchestration, Query Optimization

I. INTRODUCTION

There are growing number of Web services that provide a wealth of information. There are Web services about books (isbndb.org, libraryhing..com, Amazon, AbeBooks),about movies (api.internetvideoarchive.com), about music (musicbrainz.org, lastfm.com), and about a large variety of other topics. Usually, a Web service is an interface that provides access to an encapsulated back-end database. For example, the site musicbrainz.org offers a Web service for accessing its database about music albums. Web services deliver crisp answers to queries. This allows the user to retrieve answers to a query without having to read through several result pages. The results of Web services are machine-readable, which allows query answering systems to cater to complex user demands by orchestrating the services. Web services allow querying remote databases. 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 Leonard Cohen as input. The output will contain the songs Suzanne, Hallelujah, etc.

The API of a Web service restricts the types of queries that the service can answer. For example, a Web service might provide a method that returns the songs of a given singer, but it might not provide a method that returns the singers of a given song. If the user asks for the singer of some specific song, then the Web service cannot be called – even though the underlying database might have the desired piece of information. This asymmetry is particularly problematic if the service is used in a Web service orchestration system. The queries provided to web service have to follow the binding pattern of the Web service functions. Thus the queries have to provide values for mandatory input parameters before the function can be called. For example of musicbrainz, the function getSongs () can only be called if a singer is provided. Thus, it is possible to ask for the songs of a given singer, but it is not possible to ask for the singers of a given song. If the user wants to know, e.g., who sang Hallejah, and then the Web service cannot be used to answer this question, even though its database contains the desired information. This restriction 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. On the client side, this is a highly inconvenient limitation, in which the data might be available, but cannot be queried in the desired way. It has been called as the problem of Web service asymmetry.

The asymmetric relations are by no means outlandish: It is legitimate, e.g., to ask for singers who won a certain prize. The only way to deal with such asymmetric web services is to try out all possible input values until the Web service delivers the desired output value. For example, if the user asks for the singer of the song Hallejah, then we can use a semantic knowledgebase to get a list of singers. Then we call getSongs with every singer, and remember those for which the Web service returns Hallelujah. Obviously, this approach quickly becomes infeasible. The first limitation is runtime, with Web service calls taking up to 1 second to complete. Trying out thousands of singers that a knowledge base contains could easily take hours. The second limitation is the data provider itself, which most likely restricts aggressive querying from the same IP address. If the asymmetric relations could be queried on a case by-case basis, without materializing the

entire function, then both the user and the provider would be helped. In addition, the web services user's limited rationality with regard to the knowledge about the web service model is depicted as a reason for their limited growth on the web service market. The users could assess a need for information on the demand side. Unfortunately that was not fulfilled by the communication policy of the web service vendors.

So there is a requirement of web service system which uses on-the-fly information extraction, to collect values that can be used as parameter bindings for the Web service. This system shows how this idea can be integrated into a Web service orchestration system. It uses Web-based information extraction (IE) on the fly to determine the right input values for the asymmetric Web services. It is the solution for reducing the number of accesses. Notions of minimal rewritings have been proposed however, the goal remains as the computation of maximal results .The use of information extraction to guess bindings for the input variables and then validate these bindings by the web service. Information Extraction is the process of retrieving structured information from unstructured text .Web Information Extraction (WIE) systems differ significantly from traditional systems in both methods and goals. Whereas traditional IE systems focus on squeezing as much juice as possible from small corpora, WIE systems focus on domain independent extraction from relatively simple sentences, and rely on the redundancy of the Web to provide large quantities of information.

II. LITERATURE REVIEW

A. Query Answering With Limited Access Capability

This approach shows how to optimize queries on relations with binding restrictions. Since relations require values of certain attributes to return data, we cannot answer a query in the traditional way of answering queries. a data model and a common query language that are designed to support the combining of information from many different sources.[3 In information-integration systems, sources may have diverse and limited query capabilities. To obtain maximum information from these restrictive sources to answer a query, one can access sources that are not specified in the query (i.e., outside of query sources) and compute answer to query .Suppose we want to compare the average prices of the books sold by amazon.com and barnesandnoble.com. Since both sources require each source query to specify at least an ISBN, an author, or a title, we cannot retrieve all their book records. Suppose we can access prenhall.com to retrieve all authors who have published books through the publisher. We can use this author list to query amazon.com and barnes and noble.com to get books and their prices and to compute the average prices. Note that the authors who publish books through Prentice Hall may also publish books through other publishers. We can use the author list of Prentice Hall as a seed list to retrieve book titles from the two bookstore sources, then use these titles to retrieve more authors, and so on so forth. We repeat the iteration until the author list remains stable, then average the prices of the books from the two sources. It also offers tools for generating automatically the components that are needed to build systems for integrating information. This approach is having a query-planning framework to answer queries in the presence of limited access patterns. In that framework, a query and source descriptions are translated to a recursive data log program. Then solve an optimization problems in this framework, including how to decide whether accessing off-query sources is necessary, how to choose useful sources for a query, and how to test query containment. They develop algorithms to solve these problems, and thus construct an efficient program to answer a query. [1-3]

B. Query Rewriting Using Recursive Query Plan to Provide Maximum Number Of Answers

In Recursive Query plan for data integration, a main concern is on policy refinement and the use of data integration as starting point for refinement specification. This will ensure that refined policies implement the target systems and do not violate predefined system constraints. Parallel, distributed refinement could lead to policies which conflict with each other as the refinement. There are many solution are available .The researchers study the problem of rewriting queries using views in the presence of access patterns, integrity constraints, disjunction, and negation. They try to provide asymptotically optimal algorithms for (1) finding minimally containing and (2) maximally contained rewritings respecting the access patterns and for (3) deciding whether an exact executable rewriting exists. They have shown that rewriting queries using views in this case reduces (a) to rewriting queries with access patterns and constraints without views and also (b) to rewriting queries using views under constraints without access patterns. They have shown how to solve (a) directly and how to reduce (b) to rewriting queries under constraints only (semantic optimization). [4-5]

C. Reducing number of Access

An optimization technique which can be applied at query plan generation, and which identifies the sources that is relevant for a query, thus avoiding useless accesses to non-relevant sources. It introduces an optimization technique, which can be applied during the query evaluation process. This technique takes the constants already extracted from the sources at a certain step of the evaluation of the Datalog program associated to a query.It reduces the problem of run-time query plan optimization for sources with limited capabilities in the presence of functional dependencies and full inclusion dependencies[6].The evaluation of the Datalog program is done as follows: starting from the initial bindings in the query, we can access all the sources we can, according to their binding patterns. With any new tuples obtained (if any), we can obtain new constants. From those tuples we can access the sources again, and will get new tuples. This continues, until we have no way of doing accesses with new bindings. At each step, the constants obtained so far are stored in the domain relations. The program extracts all tuples obtainable respecting the binding patterns, but there may be tuples in the sources that cannot be retrieved.

For the example, assume the sources have the extension shown in Figure 1. Starting from a1, the only constant in the query, when we access s1 getting the tuples (a1; b1) and (a1; b2). Now we have b1 and b2 with which to access s1 and s2;

from s2 we get (a2; b1), while from s2 we get nothing. With the new constant a2 we access s1 getting (a2; b3). Finally, we access s3 with b3 getting (a4; b3; c1) (with b3 we do not get any tuples from s2). At this point, we have populated the relations si^, from which we evaluate the query with the first two rules. The answer to q is therefore the tuples (c1). Tuples (a2; b4) and (a3; b1) could not be extracted from s2.

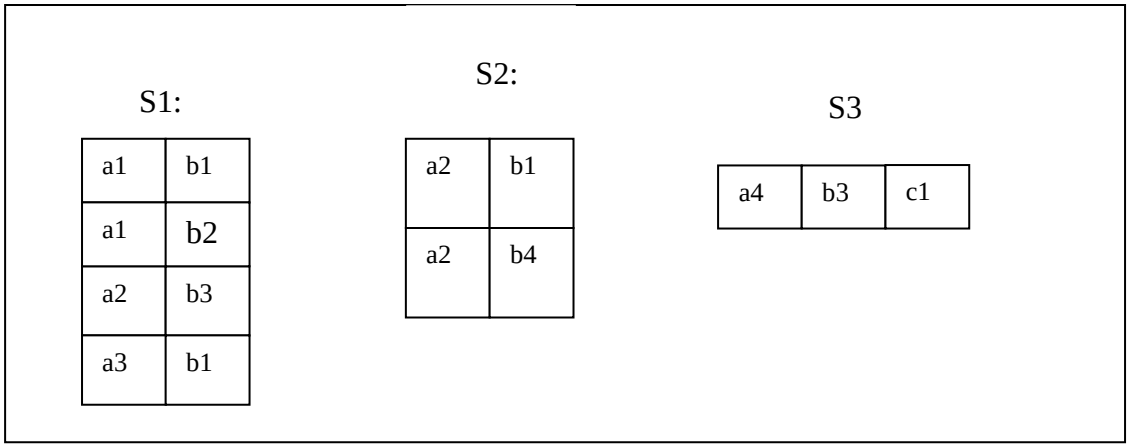

Figure 1: Extension of Sources

Full Inclusion Dependency Considerations:

Suppose we have the following sources: person(Code; Surname; City) and employee(Code; Surname; City) with employee is a subset of person. The attribute with domain City represents the city where the corresponding employee or person lives. We also have the functional dependency {Code ⟶ (City, Surname)} on both person and employee. Suppose as shown in figure below both person and employee have common information and we have attributes/initial constants as Mumbai and Delhi then we can get 5 more constants as 3 codes as 3,5,7 and 3 surname as shinde, mishra and sharma.Again by using these constants we can get ore constants by accessing a source person. By using the functional dependency if we bind code with one of these initial constants, then we will get the information that already obtained from person. Therefore access to employee is useless if we have accessed a person.

Code	Surname	City
1	sonar	mumbai
2	Sharma	Raypur
5	Shewade	Delhi
7	Mishra	Delhi
8	Jain	Pune

(A)

Code	Surname	City
1	Sonar	mumbai
5	Shewade	Delhi
7	Mishra	Delhi

(B)

Figure 2(A): An extension of Person and Employee Figure 2(B): Resultant Constants from Mumbai ,Delhi

D. CrawdSearcher

A .bozzon and M. Brambilla proposed "CrowdSearcher" a novel search paradigm that makes use crowds as first-class sources for the information seeking process. Since many people have tendency of asking for suggestions before selecting any way in their life .This tendency is hijacked by Web users, who are increasingly relying on social interaction to complete and validate the results of their search activities. This approach fulfil the gap between search system which operate on worldwide collected information which is stored in indexed form ,and social system, capable of making interaction with real people which include emotions, suggestions, opinions. Such personalized interaction occurs in social network aside of the search systems and processes. Then Search systems are superior machines to get world-wide information, the opinions collected within friends and expert/local communities from social internet sites and can ultimately determine our decisions. When such interaction is completed and users again try the use of search systems, they do it through new queries, loosely related to the previous search or to the social interaction. The proposed architecture is for integrating computerized search with human interaction, by showing how search systems can drive and capitalize social systems. [13]

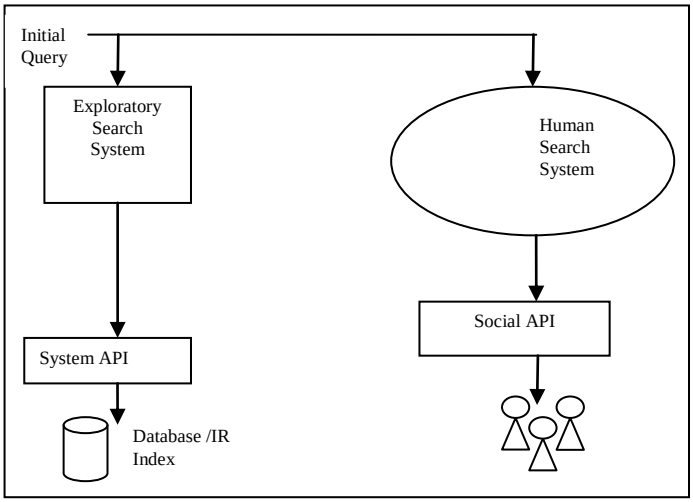

Figure 3: Overview of CrawdSearcher Approach

CrawdSearcher implements the queries depending upon human and social platforms. Steps involved are as follows:

1. The system instantiates query templates by importing information from search systems,
2. Sends queries to the social/crowd platform,
3. Gets a collection of responders involved, and gathers the results.

The **query master**, who has been selected in the crowd searching process, by being responsible (and possibly covering costs) of tasks which are spawn to the crowd and by offering friends and colleagues as responders. Important thing to remember is the social/crowd platform must import a query through a dedicated interface. Next, the platform and some dedicated users will get selected and then engaged by inviting them to reply to the question respond with their contribution. Finally, all replies are collected and manipulated for generating the final output model i.e used seamlessly integrated with the possible subsequent search.

III. PROPOSED SYSTEM

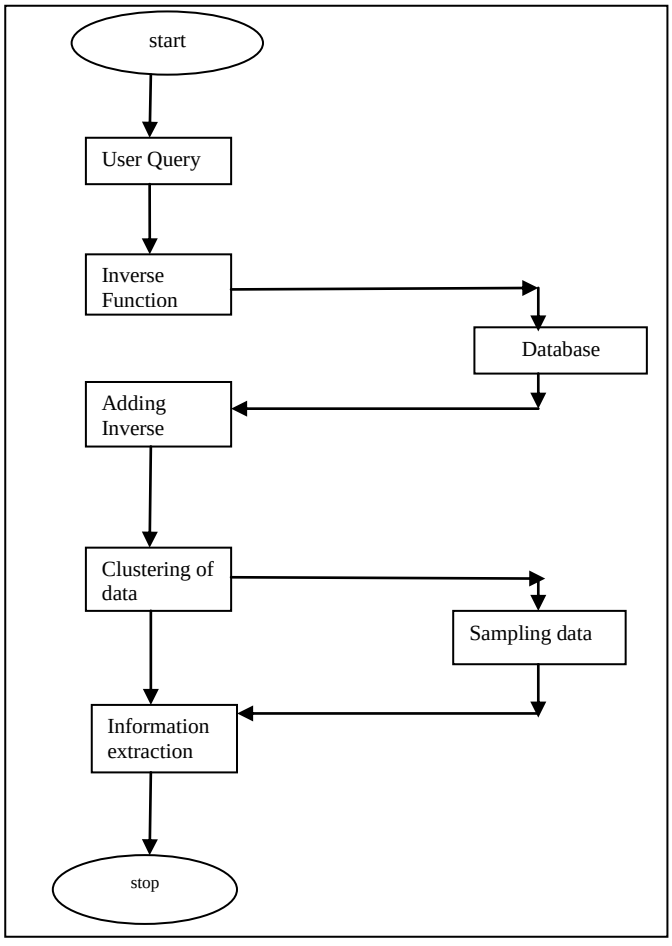

Figure4: Flow Diagram of Proposed System

A. Steps in Implementing and Adding Inverse Function in Web Service

Consider the function getSongs: $getSongs^{bff}(s,i,t) \leftarrow (sang(s,i), hasTitle(i,t))$. Here f means the attributes that we find and b means binding parameter/attribute in function definition. Information about singers and songs are likely to appear on the Web. Our proposed system provide inverse function of it as follows. Whenever $getSongs^{ffb}(s,i,t)$ is called, this system issues a keyword query of the form "Singer t" to a Web search engine(e.g., "Singer SonuNigam"). Then, the system will extract possible candidates for s from the result Web pages. Last, it will call the real Web service $getSongs^{bff}(s,i,t)$ with all candidates for s. If any of them succeeds and delivers the given t as output, then returns the values of s and i as an output of the inverse function $getSongs^{ffb}$. Thereby, the inverse function acts just like a real Web service function.

1) Finding suitable Web pages: In the example of getSongs, we know that the target type is Singer. We map each inverse function f* of f to a keyword query. This keyword query is a string of the form(tr ,x^{-}) , where tr is the target type and x^{-} are the bound variables of f*.To map $getSongs^{fbb}(s,i,t)$ to the keyword query "List of singers who sang t", ignoring the song id(i). Whenever the query evaluation calls the function f*(x1; : : : ; xn), this system issues the keyword query of f* to an Internet search engine and collects the top ten result Web pages

*2) Extracting Candidates: Extra*cting candidates from the Web pages, once the web pages have been retrieved, it remains to extract the candidate entities. Information extraction is a challenging endeavour, because it requires to extract only the required candidates. This system uses any of Information Extraction algorithms,String Matching Algorithm or Structured Extraction Algorithm. The String Matching Algorithm extracts only those entities that are already stored in a knowledge base (YAGO). The algorithm first loads all entities of the target type from the knowledge base. Then the algorithm runs through the Web pages and extracts all entities from the documents that appear in the knowledge base. This processing can be done in time O(n) in the best case and in time O(m _ n) in the worst case, where n is the total number of characters in the Web pages and m is the number of characters in the longest entity name.[8]The String Matching Algorithm has the disadvantage that it can only find entities that appear in the knowledge base. If we wish to overcome this limitation, then also consider structured data(tables ,list) extracted from web. This algorithm identifies structures of repetitive rows, where each row contains items that are separated by special strings or tags that re-appear in each row. Furthermore, the items in one column have to be of the same syntactic type (numbers, strings or dates). This procedure finds standard tables and standard lists as well as other types of repetitive structures. By comparing the elements of each column with the instances of the target type in knowledge base(YAGO),.More focus is on finding the column that constitutes most likely the answers to the query.[8-9]

3) Verifying the results for approval: The IE algorithms have delivered a set of candidate entities for the free argument x1 of f. Though Information extraction algorithm delivers desired candidates, it does not mean that the candidates would be correct: Web pages can contain many more entities of the target type than the desired ones. Therefore, all candidates have to be validated by the Web service to see whether they fulfil the conditions of the function definition. To do so, it will call the original Web service function f with all of these candidate entities, one after the other. This yields one, multiple, or no result tuples of the form(x_1,...., x_{ni})for each candidate entity x_1.If the values in the bound positions of the input values of the inverse function call f*(x1,.,.,xn), then the system delivers the tuples ($x_{1,.,.,,}$ x_{ni})as an output of the inverse function. Thereby, each candidate entity that has been extracted from the Web pages is verified by the Web service.

B. Features of Proposed System that overcomes limitation of previous approaches

1) Query Answering: The problem to be overcome in proposed system is of answering queries using views with limited access patterns. The proposed system enables to rewrite the initial query into a set of queries to be executed over the given views. Also it is possible that to find an answer for a conjunctive query over a global schema and a set of views over the same schema.

2) Information Extraction: Information extraction (IE) is concerned with extracting structured data from documents. IE methods suffer from the inherent imprecision of the extraction process. Usually, the extracted data is way too noisy to allow direct querying. The proposed system will overcome this limitation, by using IE 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. They can be used to generate candidates for this system. The approach that matches noun phrases against the names of entities that are registered in a knowledge base is also useful and simple but effective technique that circumvents the noise in learning-based NER techniques.

3) Web services: It has been observed that a considerable number of real world Web services allow asking for only one argument of a relationship, but not for the other. The proposed system use information extraction to guess bindings for the input variables and then validate these bindings by the Web service. Through this approach, a whole new class of queries has become tractable. Providing inverse functions alone is not enough. They also have to be prioritized accordingly. In proposed system data set is extracted and then validated against known knowledge base.

4) Extracting Candidates: Once the Web pages have been retrieved, it remains to extract the candidate entities. Information extraction is a challenging endeavour, because it often requires near-human understanding of the input documents. In this system it is somewhat simpler, because we are only interested in extracting the entities of a certain type from a set of Web. This is because all answers are checked by the Web service. verifying each result with the original Web service, the proposed system mitigates the central weakness of classical information extraction, its imperfect precision.

5) Appropriate Function Call: Our goal of web service is to answer queries by using only function calls. The difficulty in answering queries lies in the fact that the query is formulated in terms of the global schema, not in terms of the

functions. This is because the query expresses an information need, and not yet a constructive procedure. Since the user does not have access to the global database, we cannot execute the query directly on the tables of the global database. Rather, the algorithms automatically translate the query into query plans expressed in terms of the available Web service functions, respecting their binding pattern restrictions. The goal was to compute the maximal contained rewriting. Unfortunately, when the views have binding patterns, even the evaluation of conjunctive queries requires rewritings of unbound length. Thus, these works will produce pipelines of calls of an unbound length in order to obtain the maximal number of answers. This is infeasible in the context of Web Services. The proposed system considers a given budget of calls. This changes the goal of the evaluation. The goal is to compute the largest number of answers using the given budget of calls. Hence, it has to prioritize calls that are likely to lead to an answer. Means all calls of functions are prioritized according to query and available database.

Table 1: Comparison of various approaches with proposed system

Sr. No.	**Name Of Approach/System**	**Benefits**	**Limitation**
1	Query Answering	1. It offers data model and common query language which is used for combining information from many different sources.	1.There is a problem of how a mediator accesses sources to keep its cached data as fresh as possible, while minimizing the number of source accesses. It is an instance of a reformulation problem.
2	Query Rewriting	1. It solves the problem of rewriting queries using views in the presence of access pattern ,integrity constraints, disjunction, and negation	1. It doesn't solve the problem of asymmetric relation. 2. It doesn't provide inverse functions.
3	Reducing number of access	1. Provides global schema of data. 2. Presence of implications of dependencies enables to minimize number of accesses.	Problem of how to handle complicated query. No guarantee to obtain answer of all possible queries.
4	CrawdSearcher	1. It fills the gap between generalized search Systems and social networking site. 2. Provides architecture for integrating computerized search with human interaction. 3. Answer to query is improved as many suggestions, opinions are considered for that.	1. Not genuine way to get answer of query. 2. Doesn't provide solution for asymmetric relation. 3. Doesn't provide inverse function phenomenon.
5	Yago	1. Ontological control structure which has been used as data warehouse for data cleaning. 2. Rather than using information extraction it uses category pages from Wikipedia. 3. Uses IS-A hierarchy that helps to link two sources with near perfect accuracy. It is extendable by other sources by means of data gathered through information extraction.	1. Yago extract only 14 relation types such as subclass of, type, family-name-of from different sources of Wikipedia. 2. Yago doesn't provide inverse function.
6	DBPedia	1. It allows to ask sophisticated queries against dataset derived from Wikipedia. 2. It converts Wikipedia data into structured knowledge. It allows datasets to be either imported into third party application or can be accessed online using various database interfaces. 3. DBPedia datasets can be interlinked with with various other data sources which allows users to explore DBPedia dataset together with information from interlinked datasets.	1. Quality of DBPedia dataset needs to be improved. 2. Automation of data extraction process in order to synchronize it with changes in Wikipedia is not provided in DBPedia. 3. Atomatic consistency checks for Wikipedia content is not given in it.

IV. CONCLUSIONS

To improve the performance of web service, the proposed system uses on-the-fly information extraction to collect values that can be used as parameter bindings for the Web service. The API of a web service restricts the types of queries that can correctly answer it. This happens because of limited access pattern. Due to this, the available functions cannot answer the asked query even if the requested data present in database, because of asymmetry property of relation. The proposed system uses information extraction to guess binding for the input variables, and then validate these bindings by the web services. Also the care has been taken to prioritize inverse functions to limit the function calls.

ACKNOWLEDGMENT

I would like to express my sincere gratitude towards my guide, **Mrs. Rashmi Dhumal ,** for the help, guidance and encouragement, he provided during the Special topic seminar. This work would have not been possible without his valuable time, patience and motivation. I thank him for making my stint thoroughly pleasant and enriching. It was great learning and an honor being his student

REFERENCES

[1] Rajaraman, Y. Sagiv, and J. D. Ullman, "Answering queries using templates with binding patterns," in PODS, 1995.
[2] C. Li and E. Y. Chang, "Query planning with limited source capabilities," in ICDE, 2000.
[3] C. Li, "Computing complete answers to queries in the presence of limited access patterns," VLDB J., 2003.
[4] Cal`ı and D. Martinenghi, "Querying data under access limitations," in ICDE, 2008.
[5] Deutsch, B. Lud¨ascher, and A. Nash, "Rewriting queries using views with access patterns under integrity constraints," Theor. Comput. Sci., 2007.
[6] Cal`ı, D. Calvanese, and D. Martinenghi, "Dynamic query optimization under access limitations and dependencies," J. UCS, 2009.
[7] 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
[8] H. Elmeleegy, J. Madhavan, and A. Y. Halevy, "Harvesting relational tables from lists onthe web," PVLDB, 2009.
[9] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu, "Uncovering the relational web," in WebDB, 2008.
[10] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Y. Halevy, "Google's deep web crawl," PVLDB, vol. 1, no. 2, pp. 1241–1252, 2008.
[11] 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.
[12] N. Choi, I.-Y. Song, and H. Han, "A survey on ontology mapping,"SIGMOD Rec., 2006. [Online]. Available: http://doi.acm.org/10.1145/1168092.1168097.
[13] Bozzon, M. Brambilla, and S. Ceri, "Answering search queries with crowdsearcher," in WWW, 2012.

