Extracting and Reducing the Semantic Information Content of Web Documents to Support Semantic Document Retrieval

A. Illayarajaa[1] K. MohanKumar[2]
[1]Assistant Professor
[1, 2]Department of Computer Science and Engineering
[1, 2] Annai Mathammal Sheela Engineering College, Namakkal.

Abstract—Ranking and optimization of web service compositions represent challenging areas of research with significant implication for realization of the "Web of Services" vision. The semantic web, where the semantics information is indicated using machine-process able language such as the Web Ontology Language (OWL) " Semantic web service" use formal semantic description of web service functionality and enable automated reasoning over web service compositions. These semantic web services can then be automatically discovered, composed into more complex services, and executed. Automating web service composition through the use of semantic technologies calculating the semantic similarities between outputs and inputs of connected constituent services, and aggregate these values into a measure of semantics quality for the composition. It propose a novel and extensible model balancing the new dimensions of semantic quality (as a functional quality metric) with QoS metric, and using them together as a ranking and optimization criteria. It also demonstrates the utility of Genetic Algorithms to allow optimization within the context of a large number of services foreseen by the "Web of Service" vision. To reduce the semantics of the web documents then to support semantic document retrieval by using Network Ontology Language (NOL) and to improve QoS as a ranking and optimization.

Keywords: Data mining and warehousing, xml web application, Database

I. INTRODUCTION

A search engine is an information retrieval system designed to help find information stored on a computer system. Search engines help to minimize the time required to find information and the amount of information which must be consulted, akin to other techniques for managing information overload. The most popular form of a search engine is a Web search engine which searches for information on the public World Wide Web. Other kinds of search engines include enterprise search engines, which search on intranets, personal search engines, and mobile search engines.

Search engines provide an interface to a group of items that enables users to specify criteria about an item of interest and have the engine find the matching items within the group. In the most popular form of search, items are documents or web pages and the criteria are words or concepts that the documents may contain.

A Boolean search for an item within a group of items will either return the exact matching item or nothing. This is a rather orthodox search method where the equality between the desired item and the actual item must be exact. In application, it is sometimes far more beneficial and useful to incorporate a more lax measure of similarity between the desired item(s) and the items that exist in the group being searched.

The lists of items that meet the criteria specified by the query are typically sorted, or ranked, in some regard so as to place the most 'relevant' items first. Placing the most relevant items first reduces the time required by users to determine whether one or more of the resulting items are sufficiently similar to the query. It has become common knowledge through the use of Web search engines that the further down the list of matching items user browse, the less relevant the items become.

To provide a set of matching items quickly, a search engine will typically collect information, or metadata, about the group of items under consideration beforehand. For example, a library search engine may determine the author of each book automatically and add the author name to a description of each book. Users can then search for books by the author's name. Other metadata in this example might include the book title, the number of pages in the book, the date it was published, and so forth. The metadata collected about each item is typically stored on a computer in the form of an index. The index typically requires a smaller amount of computer storage and provides a way for the search engine to calculate the relevance, or similarity, between the query and the set of items.

Natural Language Search is the term used to describe web search engines that apply natural language processing of some form. Traditional search engines tend to use a non-linguistic model of language and the hypothesis is that NLS will provide better results - that is to say, results that more accurately and efficiently support a user's need.

A meta-search engine is a search engine that sends user requests to several other search engines and/or databases and returns the results from each one. Meta search enables users to enter search criteria once and access several search engines simultaneously. Since it is hard to catalogue the entire web, the idea is that by searching multiple search engines user can able to search more of the web in less time and do it with only one click.

The ease of use and high probability of finding the desired page(s) make Metasearch engines popular with those who are willing to weed through the lists of irrelevant 'matches'. Another use is to get at least some results when no result had been obtained with traditional search engines. Metasearch engines create what is known as a virtual database. They do not compile a physical database or

catalogue of the web. Instead, they take a user's request, pass it to several other heterogeneous databases and then compile the results in a homogeneous manner based on a specific algorithm.

No two meta-search engines are alike.

Some search only the most popular search engines while others also search lesser-known engines, newsgroups, and other databases. They also differ in how the results are presented and the quantity of engines that are used. Some will list results according to search engine or database. Others return results according to relevance, often concealing which search engine returned which results. This benefits the user by eliminating duplicate hits and grouping the most relevant ones at the top of the list.

Search engines frequently have different ways they expect requests submitted. For example, some search engines allow the usage of the word "AND" while others require "+" and others require only a space to combine words. The better metasearch engines try to synthesize requests appropriately when submitting them. Results can vary between metasearch engines based on a large number of variables. Still, even the most basic metasearch engine will allow more of the web to be searched at once than any one stand-alone search engine. On the other hand, the results are said to be less relevant, since a metasearch engine can't know the internal "alchemy" a search engine does on its result.

Semantic Search attempts to augment and improve traditional Research Searches by leveraging XML and RDF data from semantic networks to disambiguate semantic search queries and web text in order to increase relevancy of results.

In navigational search, the user is using the search engine as a navigation tool to navigate a particular intended document. Semantic Search is not applicable to navigational searches. In Research Search, the user provides the search engine with a phrase which is intended to denote an object about which the user is trying to gather/research information. There is no particular document which the user knows about that s/he is trying to get to. Rather, the user is trying to locate a number of documents which together will give him/her the information s/he is trying to find. Semantic Search lends itself well here. Rather than use ranking algorithms such as Google's Page Rank to predict relevancy, Semantic Search uses semantics or the science of meaning in language to produce highly relevant search results.

In order to understand what a user is searching for, Word sense disambiguation must occur. When a term is ambiguous, meaning it can have several meanings, the disambiguation process is started, thanks to which the most probable meaning is chosen from all those possible.

Such processes make use of other information present in a semantic analysis system and take into account the meanings of other words present in the sentence and in the rest of the text. The determination of every meaning, in substance, influences the disambiguation of the others, until a situation of maximum plausibility and coherence is reached for the sentence. All the fundamental information for the disambiguation process, that is all the knowledge used by the system, is represented in the form of a semantic network, organized on a conceptual basis.

In a structure of this type, every lexical concept coincides therefore with a semantic network node and is linked to others by specific semantic relationships in a hierarchical and hereditary structure. In this way, each concept is enriched with the characteristics and meaning of the nearby nodes.

Every node of the network groups a set of synonyms which represent the same lexical concept and can contain:

1) single lemmas ('seat', 'vacation'; 'work', 'quick'; 'quickly', 'more', etc.)
2) compounds ('non-stop', 'abat-jour', 'policeman')
3) Collocations ('credit card', 'university degree', 'treasury stock', 'go forward', etc.).

The semantic relationships, which identify the semantic relationships between the synsets, are the order principals for the organization of the semantic network concepts.

II. SYSTEM ANALYSIS AND DESIGN

A. Literature Review

Individuals on the Internet use aliases for various communication purposes. Aliases can be tailored to specific a scenario, which allows individuals to assume different aliases depending on the context of interaction [1]. For example, many online users utilize aliases as pseudonyms in order to protect their true identity, such that one alias is used for web forum postings and another for e-mail correspondence. Determining when multiple aliases correspond to the same entity, or alias detection, is useful to a variety of both legitimate and illegitimate applications. Regardless of the intent behind alias detection, it is important to understand the extent to which the process can be automated [2].

When aliases are listed on the same web page it can indicate there exists some form of relationship between them. In order to leverage this relationship, several methods are analyzed for alias detection based on social network analysis [3]. Social network analysis has permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee [4].

The networks in which aliases, extracted from web pages, are situated reveal certain aspects of the social network to which the alias corresponds. Since many people use several email addresses for related purposes, the system attempts to determine which email addresses correspond to the same entity by analyzing the relational network of addresses extracted from WebPages. In contrast to other identifiers, email addresses provide a unique mapping from address to a specific entity. Thus, no disambiguation is necessary when studying email addresses as identifiers for alias detection [4].

1) This system introduces a method that is able to group the returned citations from a search engine such as Google or Yahoo for a person-name query, such that each group of citations refers to the same person. In the output the system retains the basic search-engine returned further, within each group the system maintains the search engine

ranking order, and among groups the system maintains the relative order of citations as originally presented by the search engine [2].

The method considers three facets: attributes, links, and page similarity. For each facet the system generates a confidence matrix. Then the system constructs a final confidence matrix for all facets. Using a threshold, a grouping algorithm is applied on the final confidence matrix for all facets. The output is groups of the search-engine returned citations, such that the citations in each group relate to the same person [2].

Decision support analysis on data warehouses influences important business decisions; therefore, accuracy of such analysis is crucial. However, data received at the data warehouse from external sources usually contains errors, e.g., spelling mistakes, inconsistent conventions across data sources, missing fields. Consequently, a significant amount of time and money are spent on data cleaning, the task of detecting and correcting errors in data. A prudent alternative to the expensive periodic data cleaning of an entire data warehouse is to avoid the introduction of errors during the process of adding new data into the warehouse. This approach requires input tuples to be validated and corrected before they are loaded. There is much information that can be used to achieve this goal [6].

A common technique validates incoming tuples against reference relations consisting of known-to-be-clean tuples. The reference relations may be internal to the data warehouse or obtained from external sources. An enterprise maintaining a relation consisting of all its products may ascertain whether or not a sales record from a distributor describes a valid product by matching the product attributes of the sales record with the product relation; here, the product relation is the reference relation. If the product attributes in the sales record match exactly with a tuple in the product relation, then the described product is likely to be valid. However, due to errors in sales records, often the input product tuple does not match exactly with any in the product relation. Then, errors in the input product tuple need to be corrected before it is loaded. The information in the input tuple is still very useful for identifying the correct reference product tuple, provided the matching is resilient to errors in the input tuple. The system is referred to this error-resilient matching of input tuples against the reference table as the fuzzy match operation [7].

The goal of the system is to develop a robust and efficient fuzzy match algorithm, applicable across a wide variety of domains. A solution is needed to provide a strong foundation for adding domain-specific enhancements. Most data warehouses are built atop database systems. Robustness and efficiency is required to that the fuzzy match solution is implemented over standard database systems without assuming the persistence of complex data structures.

2) Recent surveys show that more than 80% of researchers working on data mining projects spend more than 40% of their project time on cleaning and preparation of data. The data cleaning problem often arises when information from heterogeneous sources is merged to create a single database. Many distinct data cleaning challenges have been identified in the literature: dealing with missing data, handling erroneous data, record linkage, and so on. The system addresses one such challenge called reference disambiguation, which is also known as ‘fuzzy match’ and ‘fuzzy lookup’ [5].

The reference disambiguation problem arises when entities in a database contain references to other entities. If entities were referred to using unique identifiers, then disambiguating those references would be straightforward. Instead, frequently, entities are represented using properties/descriptions that may not uniquely identify them leading to ambiguity.

The reference disambiguation problem is related to the problem of record reduplication or record linkage that often arises when multiple tables are merged to create a single table. The causes of record linkage and reference disambiguation problems are similar; viz., differences in representations of objects across different data sets, data entry errors, etc. The differences between the two can be intuitively viewed using the relational terminology as follows: while the record linkage problem consists of determining when two records are the same, reference disambiguation corresponds to ensuring that references in a database point to the correct entities.

Given the tight relationship between the two data cleaning tasks and the similarity of their causes, existing approaches to record linkage can be adapted for reference disambiguation. In particular, feature-based similarity (FBS) methods that analyze similarity of record attribute values can be used to determine if a particular reference corresponds to a given entity or not. Quality of disambiguation can be significantly improved by exploring additional semantic information. References occur within a context and define relationships/connections between entities.

A domain-independent data cleaning approach is proposed for reference disambiguation, referred to as Relationship-based Data Cleaning (RelDC), which systematically exploits not only features but also relationships among entities for the purpose of disambiguation. RelDC views the database as a graph of entities that are linked to each other via relationships. It first utilizes a feature based method to identify a set of candidate entities for a reference to be disambiguated. Graph theoretic techniques are then used to discover and analyze relationships that exist between the entity containing the reference and the set of candidates.

3) Now a day’s data mining techniques are widely used to analyze data for scientific applications and business decision-making. To build proper models and compute accurate results it is important that datasets being analyzed are accurately represented and interpreted. Many real-world datasets however are not perfect; they frequently contain various data cleaning issues such as incomplete, erroneous and duplicate data which need to be addressed before data mining techniques can be applied. As a result data mining practitioners and companies frequently spend significant effort on preprocessing of data to address cleaning issues that exist in their datasets to ensure high quality of the results [9].

This system addresses one common data cleaning challenge known as object consolidation. It arises most frequently when the dataset being processed is constructed

by merging various data sources into a single unified database, such as by crawling the web. In many real-world datasets objects/entities are not represented by unique identifiers, instead an object is represented by a description, used in a certain context, which may lead to ambiguity. An object might have multiple different representations in the dataset and also an object representation, in general, might match the description of multiple objects instead of one. The goal of object consolidation is to correctly group all the representations that refer to the same object.

The approach views the underlying database as an attributed relational graph (ARG) where nodes correspond to object representations and edges to relationships. This technique first used feature-based similarity to determine if two representations can refer to the same objects. If based on FBS similarity two representations can refer to one object, then the relationships between those representations are analyzed to measure the connection strength between them. Clustering techniques are then employed to consolidate the representations of objects based on the FBS similarity and connection strength among them.

One of the fundamental problems in data cleaning and information integration is determining when two tuples refer to the same real-world entity [7]. In information integration, determining approximate joins is important for consolidating information from multiple sources; most often there will not be a unique key that can be used to join tables in distributed databases, and the system must infer when two records from different databases, possibly with different structures, refer to the same entity. Traditional approaches to duplicate detection are based on approximate string matching criteria, in some cases augmented with domain specific rules.

This approach also makes use of attribute similarity measures, but in addition, it takes into account the similarity of linked objects. The links between objects of the same type, so that when two records refer to the same individual, that may in turn allow making additional inferences. In other words, the reduplication process is iterative.

In many applications, there are a variety of ways of referring to the same underlying entity. Given a collection of entity references, or references for short, the system would like to a) determine the collection of `true' underlying entities and b) correctly map the references in the collection to these entities. This problem comes up in many guises throughout computer science. Examples include computer vision, there is a need to out when regions in two different images refer to the same underlying object natural language processing where the system would like to determine which noun phrases refer to the same underlying entity and databases, where, when merging two databases or cleaning a database, there is a need to determine when two records are referring to the same underlying individual [3].

References are dissolved when they are connected to each other via relational links, as in the bibliographic domain where author names in papers are connected by co-author links. Now entity resolution becomes collective in that resolution decisions depend on each other through the relational links. Collective entity resolution improves performance over independent pair-wise resolution.

Another contribution of this system is an unsupervised Gibbs sampling algorithm for collective entity resolution. It is unsupervised because the system does not make use of a labeled training set and it is collective because the resolution decisions depend on each other through the group labels. Further, the number of entities is not fixed in this model; a novel sampling strategy is proposed to estimate the most likely number of entities given the references.

B. Existing System

The semantic web, where the semantics of information is indicate using machine – process able language such as Web Ontology Language (OWL) automatic processing of information assets tagged with OWL, focusing on their semantics rather than on the way they shown on the web. Information about web services can also be semantically tagged to describe their functionalities in terms of input, output parameters, preconditions, effects and invariants. These semantic web services can then be automatically discovered, compose into more complex services, and executed. Approach of optimization uses a combination of functional and nonfunctional considerations so that we can cover both perspectives.

The functional perspective comprises a set of metrics related to how well the functionalities of the constituent services fit together. Semantic quality is such core metric measuring the degree of semantic similarity between the outputs produced by constituent services and the inputs ability for the composition, indicating thee “required by their peers. Such a quality is one of the measures of the overall functional quality for the composition, indicating the “goodness of fit” between the functionalities of the constituent services.

C. Proposed System

Focus on semantic quality as the main indicator of functional quality. To measure the degree of semantic similarity, we use the concept of semantic link, which is defined as the semantic connection between the corresponding pairs of web service parameters, analyzed using DL-based matchmaking. This concept is used to evaluate, compare, and classify the quality of connections and their compositions. This is important to ensure semantic cohesion of data exchanged (or shared) between services closest and avoiding services “misfiring” and ignoring incompatible data.

Indeed some services can only interpret some semantic descriptions, rejecting any others. Web service compositions could thus be optimized and ranked using not only nonfunctional parameters such as the well-known Quality of Service (QoS) but also semantic quality as a core indicator of functional quality propose to unify both types of criteria in an innovative and extensible model, allowing us to estimate and optimize the quality of service compositions. Demonstrate the utility of this model and its advantage over single-perspective optimization models using a number of simulation experiments. To speed up the application of the proposed optimization model in a context of realistic scale, propose an approach using Genetic Algorithms (GAs) which adapts the methods.

III. SYSTEM IMPLEMENTATION

The term "software maintenance" is used to describe the software engineering activities that occur following delivery of a software product to the customer. The maintenance phase of the software life cycle is the time period in which a software product performs useful work. Maintenance activities involve making enhancement to software products, adapting products to new environments and correcting problems. Software product enhancement may involve providing new functional capabilities, improving user display and modes of interaction, and upgrading external documents. Adaptation of software to a new environment may involve moving the software to a different machine. Problem correction involves modification and revalidation of software to correct errors. The enhancement of this project can be accomplished easily. That is, any new functional capabilities can be added to the project by simply including the new module in the homepage and giving a hyperlink to that module. Adaptation of this project to a new environment is also performed easily.

Corrective Maintenance

Even with the best quality assurance activities, it is likely that they customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Adaptive Maintenance

An activity that modifies the software to properly interface with a changing environment. The system has been modified so that various change include to the new system.

Module Description

The project consists of three modules. They are,

1) Query Handler
2) Search Result Analyzer
3) Search Result Optimizer

1) Query Handler:

Search: It is the process of finding the particular details from some different locations.

Query: Queries are the primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format. Many database management systems use the Structured Query Language (SQL) standard query format.

Once the query is passed to the database, it will analysis the whole database and returns the details in which query matched by the lookup operation with the help of descriptive matched search.

Receiving the Query

Once the user enter the input in the search activity then it will divided into number of sub string.

Look up Process

During this process each substring must be matched with the web page description and it returns the index of that document if it matched by the substring.

Indexing

Once if we receive the index of full text of the pages it finds. To improve the performance of search engine, it will ignore the common words called stop words (such as the, is, on, or, of, how, why, as well as certain single digits and single letters). Stop words are so common that they do little to narrow a search, and therefore they can safely be discarded. Also it will ignore some punctuation and multiple spaces, as well as converting all letters to lowercase, to improve the performance of search engine.

2) Analyzers

It is the component that pre-process input text.

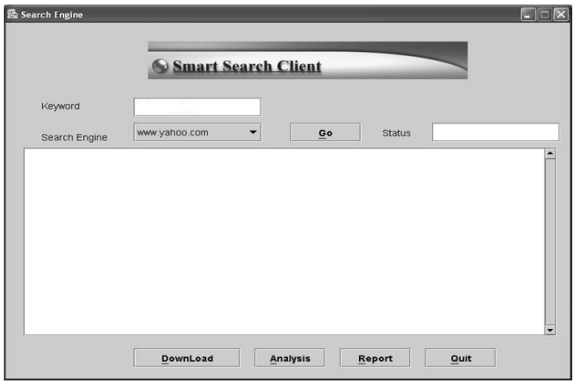

Fig. 1: User Interaction Details

Search Result Analyzer

3) Analyzers

It is the component that pre-process input text. They are also used when searching (the search string has to be processed the same way that the indexed text was processed). Therefore, it is usually important to use the same Analyzer for both indexing and searching.

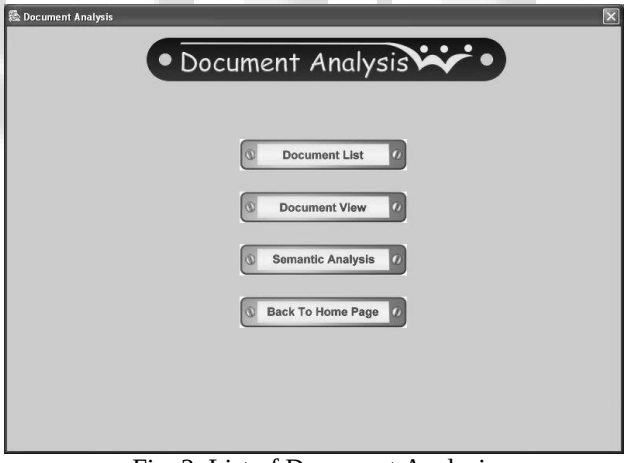

Fig. 2: List of Document Analysis

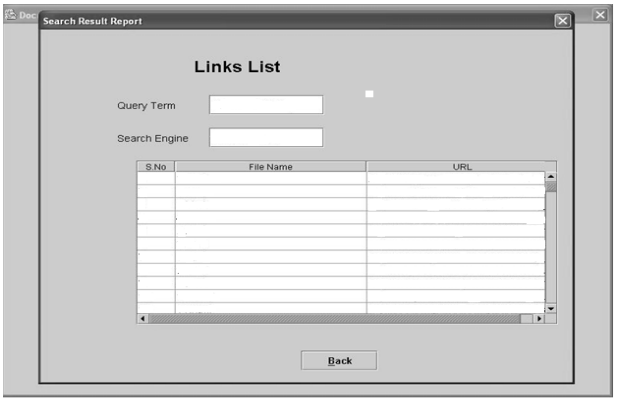

Fig. 3: Document List Analysis

4) Document List

It is used to gather the list of documents to be viewed by the user along with its URL.

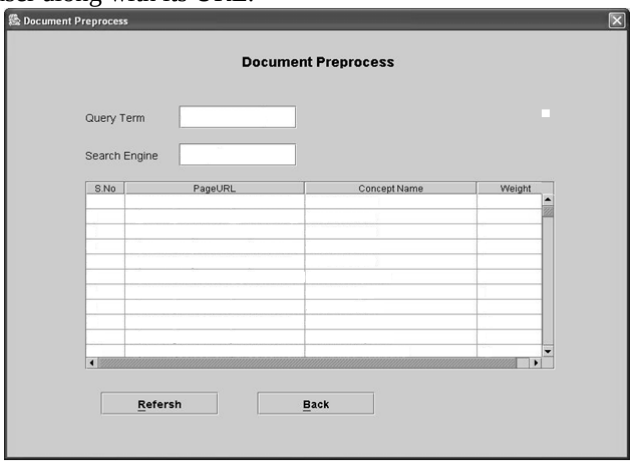

Fig. 4: Document preprocess

5) Document View

It is used to gather the list of documents to be viewed by the user along with its URL with the calculation of page factor. We can calculate the probability that the particular document to be viewed.

6) Filters

It is used to provide for simpler support for additional filtering (or enrichment) of analyzed streams, without the hassle of creating your own analyzer.

After getting the index of the input text we need to analyze the results index with the help of Genetic Algorithms (GA). It will use the concept of ranking and optimization concept. With the help of the algorithm we need to give the page factor (page weight or number of times the particular document to be used or viewed by the user).

7) Semantic Analysis

It is the process of finding the match with the help of semantic words. It is mainly used to improve the effectiveness of the performance. This module stores all the details of semantic web pages along with its weight factor.

B. Search Results Optimizer

It is mainly used to extract to the details from the database in the order of descending order. I.e. the document which have highest weight factor that should be displayed first not in the order of it will fetch from the database.

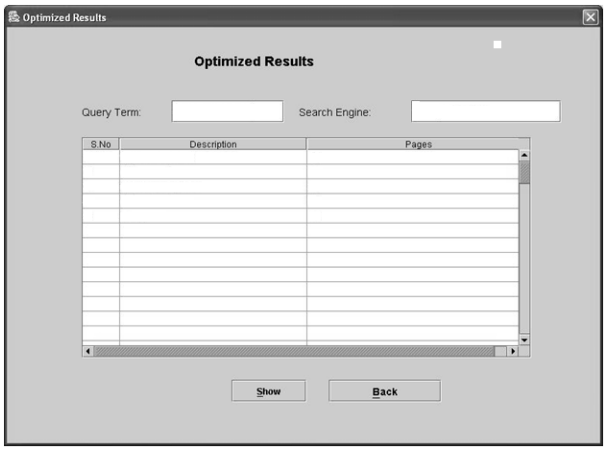

Fig. 5: Optimized Result page

IV. CONCLUSION

To address QoS-aware semantic web service composition in a context of significant scale, we propose a GA-based approach to optimizing web service compositions, which considers both the nonfunctional qualities of the composition, and the quality of semantic fit. Combining both allows us to consider both the user perspective to desired qualities, and the composition perspective of costs involved in underserviced alignment within the composition. The first feature of our approach is an innovative and extensible model to evaluate the quality of 1) web services (i.e., QoS), 2) their semantic links, and 3) their compositions. The semantic extension aims at evaluating the level of heterogeneity between the output and input parameters of services. In cases of low quality of semantic links, reflecting mismatches between exchanged data of services, data mediators are required to ensure seamless compositions. Composition designers are then expected to compensate the difference between services parameters by means of data mediators, the cost of this is provided by our semantic quality model.

The high costs of the data integration in the overall process of service composition can also be reduced by combining QoS and functional quality of semantic links proposed here. The second feature of our approach concerns the use of GA-based approach for discovering near-optimal compositions under a large number of candidate services. This relies on formalizing our extended quality model as an optimization problem with multiple constraints.

V. FUTURE ENHANCEMENT

In terms of further work, one direction is to consider compositions based on the contextual availability of web services. Indeed, our approach relies on a precomputed tasks model; these forces the goal to be satisfied according to the designed subtasks and constraints the choice among available services. Such precomputations may lead to unsatisfied goals depending on the context.

Another direction for future work is to consider how goals (and more specially preconditions and effects) of tasks and their candidate web services match together (e.g., from a semantic dimension point of view). For instance, a task can be fulfilled by services achieving a goal with more general or specific preconditions and effects, but which still fit the goal of the task. It would be interesting to consider further research in a more precise difference operator, which is also easy to compute in expressive DLs. According to the results it seems important to optimize DL reasoning to scale up the overall process of optimization. Determining the most appropriate parameters for the GA phase requires further experimentation if this is to be made truly usable for its target end users.

This paper presents some GA parameterization, but guidelines to set up these parameters need to be investigated. In addition, we will investigate how to improve the GA performance especially in the last iterations, when we want to preserve feasible solutions. Another area of investigation is the selection of initial compositions of tasks as a powerful tool to optimize the overall quality at present we take the task composition template as given. Two final considerations for further work would be the dynamic

distribution of CSOP to improve convergence time of our approach in larger domains, and security aspects of service composition quality.

REFERENCES

[1] ChristosFaloutsos, Kevin S. McCurley, and Andrew Tomkins ,(2004),"Fast Discovery of Connection Subgraphs" ACM Washington.
[2] Dmitri V. Kalashnikov, Zhaoqi (Stella) Chen, SharadMehrotra ,(2008)," Web People Search via Connection Analysis".
[3] Indrajit Bhattacharya LiseGetoor "A Latent Dirichlet Model for Unsupervised Entity Resolution" Department of Computer Science University of Maryland, College Park.
[4] Indrajit Bhattacharya and LiseGetoor "Iterative Record Linkage for Cleaning and Integration" Department of Computer Science University of Maryland College Park.
[5] Mitri v. kalashnikov and SharadMehrotra "Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph" ACM Transactions on Database Systems.
[6] RalfHolzer,BradleyMalin and Latanya Sweeney, (2005) ,"Email Alias Detection Using Social Network Analysis" Chicago, Illinois.
[7] SurajitChaudhuri ,Kris Ganjam and VenkateshGanti , (2003),"Robust and Efficient Fuzzy Match for Online Data Cleaning".
[8] Xin Dong, Alon Halevy and JayantMadhavan , (2005) ,"Reference Reconciliation in Complex Information Spaces".
[9] Zhaoqi Chen Dmitri V. Kalashnikov and SharadMehrotra "Exploiting relationships for object consolidation" Computer Science Department University of California.

