Resource Capability Discovery and Description Management System for Bioinformatics Data and Service Integration - An Experiment with Gene Regulatory Networks

Emdad Ahmed
Integration Informatics Laboratory
Department of Computer Science
Wayne State University
Detroit, MI 48202, USA
Email: emdad@wayne.edu

Abstract—Traditional legacy HTML based web sites/page can be thought of as web services because the dynamic web pages can take user input argument via web forms and response to user query. The ability of agents and services to automatically locate and interact with unknown partners is a goal for Web based Data Integration system. This "serendipitous interoperability" is hindered by the lack of an explicit means of describing what web pages are able to do and in order to do it what input it takes and what output it produces, that is what is their capabilities [1]. The tremendous success of the WWW is countervailed by the efforts needed to search and find relevant information. For tabular structures embedded in HTML documents, typical keyword or link-analysis based search fails. The next phase envisioned for the WWW is automatic ad-hoc interaction between intelligent agents, web services, databases and semantic web enabled applications. A large amount of information available on the Web is formatted in HTML tables, which are mainly presentation oriented and are not suited for database applications. As a result, how to capture information in HTML tables semantically and integrate relevant information is a challenge. We are envisioning another layer of web abstraction where user can query intra web document table like structure. Our prototype application is based on WebFusion and an ad hoc query language BioFlow [2], [3], [4], [5], [6] a software agent that can simulate a person interacting with web search forms and extracting information from the resulting pages by means of an API. We need to develop a framework which is able to query search web forms and the web page tables in a SQL way. In this context we also report a Java based implementation for integrating Flybase and AlignACE site.

Index Terms—Table structure, Table modeling, Web mining, Ontology generation, Semantic Web, Intelligent Wrapper, Web Information Extraction, HTML forms, Web Data Integration, extraction ontology, Hidden Web, Web Automation.

I. INTRODUCTION

Turning the current Web into a Semantic Web requires automatic approaches for annotation of existing data since manual annotation approaches will not scale in general. By "capability" we mean an operation offered by the provider of a data source to a user to extract data from the services data store and structure it for transfer to the customer [7]. By Web resource we mean any information, data structure, representation which can be queried by some language. Example of web resource are: URLs, links, Web Forms which can be filled for query, outgoing URLs, Keyword, attribute which can be found inside Meta Tag (this conforms to key string matching problem) or anywhere in web page; Repeated structure found in web page (e.g Table, list). Information extraction from the Web is the process of extracting data from a set of Web pages in order to fill a given database schema. Web information extraction presents two particular features: (a) the input documents are semi-structured: they are neither raw data (full text) nor strictly typed (relational tables), but a structure is provided (typically by HTML tags) (b) the information sources are heterogeneous, i.e. the same kind of information is often presented in different formats. A Web IE system receives two kinds of input: a DB schema that represents the information the user wants to retrieve; a set of IS (Web sites) containing the data to be extracted.

Such a system aims at populating the database with data extracted from the sources and must provide mechanisms for dealing with semi-structured documents and for accessing heterogeneous information sources. The presence of heterogeneous information sources, whose models are not known a priori, makes scalability one of the main features for these systems. In other words, the system must be easily programmed for dealing with a large number of different information sources. An important design element for these systems is the representation techniques used for modelling the database schema and the information sources. Indeed, this issue permits a classification of approaches to Web Information Extraction, which are as follows: Manual extraction, Procedural extraction, Declarative extraction and Automatic extraction.
A. Motivating Example

Example 1. Suppose a user want to do the following series of activities on a site:

- Go to www.amazon.com.
- Type "Ruby" into the search text field.
- To narrow down the results, click "Books",
- then for further narrowing click "Computers & Internet" in the left sidebar.

By fetching the web page from the URL and parsing and analyzing the links and auto form filling, the above tasks can be automated by our system. We call it Web User Activity Automation. Web task automation can be done by two different ways: Recorded Macro and Programmable API. We will go for the second one.

Example 2. Suppose a client want to search for Book from two sites www.amazon.com and www.bn.com at once and see the combined result. Both the site offer advanced search web form. Traditionally the client has to visit both the sites iteratively filling the form and gather data manually. If a Web Data Ingetration System as envisoned by ours is present, the client has to submit the query once and the system will do the rest automatically on the fly.

Example 3. Let us consider the following SQL query for book search from amazon site.

```
SELECT Title, Rating
FROM amazon.com
WHERE Title like "Discrete Mathematics" AND Rating=5;
```

By analyzing the advanced search option of amazon, we see that Title attribute can be mapped to Interface Schema, whereas Rating attribute can be mapped to Result Schema. For the above example with the Interface that are available in amazon site, user query can be satisfied only for the Title attribute, not for Rating attribute. Consequently, user will see a lot of tuples containing all the Rating returned in response to query, whereas he/she is only interested to see record corresponds to Rating 5. Our system can solve these types of user query by efficiently mapping user query to site.

Technical Challenges: The web forms of the two sites look the same but how to automatically fill the form is much more difficult as will be clear from the following discussion. The client will have a GUI, the query will be submitted from there. The query need to be rewritten as per the mappings of the respective site. Our proposed web resource capability description/ discovery method will take care of the mappings. Each site’s capability (form information) will be captured and stored in resource description repository. The envisioned overall architecture is shown in the following figure 1. In response to the form, we will have tabular data. Semantic enrichment of the tabular data will be made semi-automatically and through ontology and human knowledge. Tabular schema will also be stored in the resource description repository. Thus our resource capability description will be focused toward handling web form and representing table schema, i.e column name identification and label assignment. The rest of the paper deals with based on the above two ideas. The current research is our ongoing work where we proposed and identified a model for web data integration. The current focus of our research is to describe a web page so that some body else can later use to query source. We have developed prototypes for bioinformatics web data integration. To the end, our system can solve Web Data Integration problem in an ad hoc manner as follows:

```
Define function RegulatoryRegion
Extract 800bp sequence
Using wrapper FlyBaseToACE in ontology GeneMapping
From URL Flybase
Submit FBgnNum string;

Define function motif
Extract 6..8bp sequence
Using wrapper FlyBaseToACE in ontology GeneMapping
From URL AlignACE
Submit 800bp string;
```

B. Related Work

Turning the current web into a semantic web requires automatic approaches for annotation of existing data since manual approaches will not scale in general. [8] presented a method for automatically filling in forms to retrieve the associated dynamically generated pages. They consider the task of duplicate record elimination and uses a stratified sampling approach to decide when a particular source has been fully extracted. Using their approach automated agents can begin to systematically access portions of the "Hidden Web". The Hidden Web Exposer (HiWE) project at Stanford proposed a way to extend crawlers beyond
the Publicly Indexable Web (PIW) by giving them the capability to fill out Web forms automatically. The related work references are the following: [9], [8], [10], [11], [12],[13], [14], [15] presented DeLa, which reconstructs (part of) a “hidden” back-end database. It does this by sending queries through HTML forms, automatically generating regular expression wrappers to extract data objects from the result pages and restoring the retrieved data into an annotated (labelled) table. The present research work is our ongoing work where we have identified and proposed ontology based and declarative workflow query language for ad hoc web data integration on the fly [16], [17]. Our experimentation on Bioinformatics Web Data and Service Integration has also been reported in another work [18].

II. DEEP WEB QUERY MODEL

Structured Web databases can be queried via query forms or through Web service interfaces. We uniformly refer to both access methods as “query interfaces”. Through query interfaces, data consumers (e.g. end users) are able to express their information needs by imposing selection conditions on certain attributes of interest. Our system views a Web database as a single relational table $DB$ with a set of queriable attributes $A_q = \{attr_{q1}, attr_{q2}, \ldots, attr_{qn}\}$ (interface schema) and a set of result attributes $A_r = \{attr_{r1}, attr_{r2}, \ldots, attr_{rm}\}$ (result schema). Each $attr_{qi} \in A_q$ represents the queriable attribute through the query interface, while the result attribute $attr_{rj} \in A_r$ corresponds to the attributes displayed in the result pages. Each query operation can be modelled using SQL syntax as:

```
SELECT \{attr_{r1}, attr_{r2}, \ldots, attr_{rm}\}
FROM DB
WHERE attr_{q1} = val_{q1}, attr_{q2} = val_{q1}, \ldots, attr_{qn} = val_{qn}
```

where $val_{qi}$ is the corresponding attribute value filled into the query form.

We model the web site as $S \subseteq Q \times R$, where $Q$ is the query interface schema and can be represented as $Q \subseteq F \times P$ and $R$ is the result schema, can be represented as $R \subseteq L \times V$. The semantics of the above definition is as follows: we have a set of form label $F$ and a set of corresponding parameter $P$. In response to the web form, we will have tabular data, which have a set of values $V$. The label $L$ of the values $V$ may or may not be in the web page. One of the major fundamental research problem is how to find $L$, given $F, P, V$. This is a separate research topic and we address the problem somewhere else.

III. ONTOLOGY DEFINITION

Ontologies are a convenient way to describe relationships among the terms in the universe being modeled for the purpose of heterogeneous information integration on the web. It is also a convenient way to describe extraction rules for web resources in the form of wrappers. So, in BioFlow, we allow schema mapping relationships and data extraction rules for web sites be included for the use by the query engine. An ontology mainly captures three types of relationship: synonym, isa and part of [19]. The general structure of ontology definition is shown below - for a given ontology $o$, its definition contains a set of schema mapping statements and a set of wrapper statements. The syntax of the mapping statements and wrapper statements are shown next.

define ontology $o$

schema mapping statements/wrapper statements;

end ontology $o$;

mapping $m : URL \ u \rightarrow a_1 : b_1, \ldots, a_k : b_k$

$c_1 \#d_1, \ldots, c_m \#d_m, e_1 @ f_1, \ldots, e_n @ f_n$;

wrapper $w : URL \ u \rightarrow a_1 :: r_1, \ldots, a_k :: r_k$;

where $a_i, b_j, c_k, d_l, e_m, f_n$ are terms (including nested terms), and $r_p$ are regular expressions corresponding to XPath expression of the table in an HTML page, and where $a : b$ means $a$ is a synonym of $b$ (and vice versa), $c \#d$ means $c$ is a homonym of $d$ (and vice versa), $e @ f$ means $e$ is an antonym of $f$ (and vice versa), and finally $a :: r$ means $a$ is an attribute located at $r$.

As a simple example of ontology, we present here a table pertaining to Flybase site. We call the table ontology GenEMapping. The above ontology shows mapping between

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG Number</td>
</tr>
<tr>
<td>CG12055</td>
</tr>
<tr>
<td>CG6871</td>
</tr>
<tr>
<td>CG7699</td>
</tr>
<tr>
<td>CG1903</td>
</tr>
<tr>
<td>CG6864</td>
</tr>
<tr>
<td>CG4581</td>
</tr>
<tr>
<td>CG1766</td>
</tr>
<tr>
<td>CG10120</td>
</tr>
<tr>
<td>CG3476</td>
</tr>
<tr>
<td>CG2107</td>
</tr>
<tr>
<td>CG6050</td>
</tr>
</tbody>
</table>

a CG number to a corresponding FBgn Number gene, its direction, forward or reverse as well as its corresponding range in the chromosome arm. We went further in geneontology site to see synonymous name for the Eftum gene. The following three names showed up: CG6050; 1(3)L4569 and mtEF-Tu. We also get the Flybase Database entry field as FBgn0024556. We report in the above table last row only one mapping (CG6050, FBgn0024556), whereas user can pose query using any one of the set \{CG6050, 1(3)L4569, mtEF -Tu, FBgn0024556\}. A system is required which can rewrite user query variable to the site. We need to use a most Generalized Term as query variable to be submitted into the site. For the Flybase site, the Generalized Term happens to be the FBgn number. We store all the synonymous relationship in relational table.
which is a more a nomenclature and taxonomy, than a formal ontology, is highly successful and extensively used. Although GO is technically a nomenclature rather than an ontology it has been successfully used to annotate large volumes of data and consequently support interoperability and integration from heterogeneous data sets. Applied to Bioinformatics, an ontology is a "controlled vocabulary for the description of the molecular functions, biological process and cellular components of gene products". Current data integration approaches should be enhanced by ontological knowledge and ISA relationships in ontologies should be explored. [21]  

IV. WEB FORM FILLING  

Information transmitted to the server in a CGI request is fundamentally just a list of (name, value) pairs with appropriate URL encoding string. Thus we can characterize a form with n controls as a tuple F = <U, (N1, V1), (N2, V2), ..., (Nn, Vn)>, where U is the URL to which the encoded CGI request is sent, and the (Ni, Vi) are (name, value) pairs to be sent. [10] described 5 heuristics to deal with the from filling problem. There are two ways to submit a form for CGI processing. HTTP POST method submit the form in the body of the request. HTTP GET method submit the form as part of the URL. A question mark (?) separates the base URL and action path. This query is then sent directly to the Web site of interest. It has the same effect as that of a user clicking the Go button without selecting or typing anything on the Web form. For our experiment with Flybase and AlignACE site form filling, the following Java code snippet captures those functionality:

```java
url1 = urlString.toString();
FlyBaseToACE fetcher1 = new FlyBaseToACE(url1);
if (HEADER.equalsIgnoreCase(option) ) {
    pageHeader1 = fetcher1.getPageHeader();
}
else if (CONTENT.equalsIgnoreCase(option) ) {
    pageContent1 = fetcher1.getPageContent();
    rangeIndex = pageContent1.indexOf("Sequence location");
    if (rangeIndex != -1) {
        ourRange = pageContent1.substring(rangeIndex+26, rangeIndex+51);
        checkForwardReverse = pageContent1.substring(rangeIndex, rangeIndex+60);
        computedRange = deleteComma(ourRange);
        url2String.append("http://www.flybase.org/cgi-bin /getseq.html?range=");
        url2String.append(computedRange);
        url2String.append("&source=dmel&format=");
        url2String.append("DecoratedFasta&address=");
        if (checkForwardReverse.indexOf("+") != -1) {
            url2String.append("-2000"); // the gene is forward
        } else {
            url2String.append("+2000"); // the gene is reverse
        }
        url2 = url2String.toString();
        FlyBaseToACE fetcher2 = new FlyBaseToACE(url2);
    }
    else {
        System.out.println("THE GENE IS REVERSE COMPLEMENTED,

add 2K downstream");
        pageContent2 = fetcher2.getPageContent();
        output800bp = process2Kdown(pageContent2);
        url3 = "http://atlas.med.harvard.edu/cgi-bin/alignace.pl?

name=Grossophile&nc=10&xp=10&gc=0.38&seq=";
        url3 += url1 + output800bp + pageContent3;
        FlyBaseToACE fetcher3 = new FlyBaseToACE(url3);
        pageContent3 = fetcher3.getPageContent();
        log (pageContent3);
    }
}
```

A. Need to consider Web Form DOM  

In spite of having a tag in the HTML specification called label for the declaration of a label, it is almost not used. Form labels refers to which is visually adjacent to the left or on the top of a form field. Labels in Web Form always appear in HTML page, but DOM do not catch it, programmers have to parse the page and extract label. Based on the above discussion, we propose to take care of the web form label. DOM for a typical web form look like:

```
document.form.name.element.value (Existing)
document.form.label.name.value (our proposal)
```

B. HTML tables - location and extraction problems  

Data on the Web in HTML tables is mostly structured, but we usually do not know the structure in advance. Thus we cannot directly query for data of interest. [9] proposed a solution based on document-independent extraction ontologies. DeLA uses four heuristics to assign label for the tabular data. As for their second heuristics, when data

### Table II

<table>
<thead>
<tr>
<th>Generalized Term</th>
<th>Synonym Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBgn00024556</td>
<td>EF-Tum</td>
</tr>
<tr>
<td>FBgn00024556</td>
<td>CG6050</td>
</tr>
<tr>
<td>FBgn00024556</td>
<td>l(3)L4S69</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>mEF-Tu</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>GAPDH</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>BESTriCH12586</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>CG12035</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>GAPDH dehydrogenase</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>GA3PDH</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>Gapd</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>GAPDH-1</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>GAPDH-I</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>Gapdh43E</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>gh12586</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>glyceraldehyde 3 phosphate dehydrogenase1</td>
</tr>
<tr>
<td>FBgn00001091</td>
<td>Glyceraldehyde-3-phosphate dehydrogenase1</td>
</tr>
</tbody>
</table>

as multivalued dependency [20] as follows: GO ontology
is represented in voluntary labels in table headers such as \(<\text{TH}>\), \(<\text{THEAD}>\), then the label assignment is trivial. We take whatever we find inside the heading. We experimented with the following site: http://www.mrc-lmb.cam.ac.uk/genomes/FlyTF/quite_sure_TFs.html to test the functionality and materialize the web response in MySQL relational DB. We use MySQL as it outperform other relational DB for Data Intensive Web Application (DIWA) [22]. On the fly materialization of web page table need to take care of the following delimiter characters and clean those, otherwise they will conflict with the DB system’s DDL and DML.

String delims = "!@#$&* ()_-+=[]\;,.";

More heuristics for column label assignment are as follows: Inner Label Assignment (extract repeating prefix as label of column); Inter Label Assignment (consider consecutive two columns, first one is the label, second one is the data); Column Splitting (first portion is the column label, second portion is the data).

We wish to add more heuristics such as any consecutive four character in the set "ACGT" will not be anything other than a sequence (of course exception is there like TATA, which is a company name as well as a gene sequence) We also wish to add that only syntactic labeling is not enough. Suppose we get X as the repeating prefix character in a column of all the data. Then according to our heuristics it will be labeled as X. But X can be chromosome as well as geometric coordinate. In order to mitigate the confusion, we plan to use web knowledge source such as Wikipedia, Google etc to resolve the concept extraction, thereby enabling semantic labeling.

V. INTEGRATING FLYBASE TO ALIGNACE SITE

In this section, we briefly discuss our experience with implementation of a Bioinformatics Web Data Integration system. Flybase is a site about a database for drosophila genetics and molecular biology. We search the site for gene, say sin3. In response to the quick search, a overview page shows the attribute name and values e.g sequence location, gene direction etc. We extract those from the overview page. Then we get the genome region, the result is given in detail page as Decorated Fasta format. The Flybase site has limited query processing capability such as it can take 2K 5K sequence to update region display. But in our program, we allow user to add arbitrary region, then also allow to add arbitrary up and down sequences. Our target is to extract say 800 bp from start sequence from sin3 gene. The whole process is automated as a sequence of web access as scientific workflow. Then the extracted sequence is automatically submitted in the AlignACE site to extract sequence motif, say 6 to 8 bp. We have plan to extend the functionality to integrate from AlignACE to TRANSFAC as well.

VI. IMPLEMENTATION ARCHITECTURE

The high level architecture of the system has been shown in Figure 1. Here we implement the systems as a layered architecture consist of four layers as depicted in the following Figure 2. The first layer is the entire world-wide web on the Internet. Because HTML documents are not only generated automatically by specific application programs such as FrontPage, DreamWeaver, but also written by human manually, it is common that some documents have syntactic errors. Tools like JTidy can be used to fix the HTML tag errors. The second layer is the Intelligent Wrapper that will implement our proposed Label Assignment method. It accesses the world-wide web through the Internet (Java URL and URLConnection class), fetches HTML documents and converts them into web objects as nested table. In order to speed-up query processing, the Intelligent Wrapper will also cache web objects in the local resource description repository and builds proper indexes on these web objects. It will allow the user to adjust web objects by adding, adjusting or deleting attributes or objects and it can also learn the pattern from the user’s interaction and then process similar HTML documents accordingly. Not only the proposed Wrapper will extract tabular data from web, but also it will map the data into attribute-value construct so that the system can later query as ordinary relational database. That is why it is called the Intelligent Wrapper. The third layer of the system is the Search and Inference Processor that will implement BioFlow. It is mainly in charge of query processing. It obtains queries to be processed from the user interface layer and checks whether or not web objects involved are in the local resource description repository. If web objects involved are not in the local resource description repository, then it invokes
the Intelligent Wrapper to fetch them. The last layer is the Intelligent User Interface. Two kinds of interfaces will be provided: textual interface and browser interface (we have already implemented using Java JEditorPane). They will provide different kinds of environment for the user to issue commands and express queries, perform syntactical analysis, and pass valid commands in internal format to the Search and Inference Processor. They also display query results generated by the Search and Inference Processor to the user in different format, the web objects generated by the Intelligent Wrapper, and original web document like Internet Explorer.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented an implementation of a procedural and logical approach to web resource capability description/discovery. By capability we mean what a web site can do for us and in order to perform its tasks what input it needs and what output it produces. In this connection we have identified web form to be the gateway. In response to web form, we assume a tabular output. By our Intelligent Wrapper, we convert the repeated structure into relational model, we can identify attributes and values in the table by our label assignment heuristics, then we can apply normal DB SQL like queries on it.

The main contribution of this paper is that we have presented a new top-down approach how to query HTML pages declaratively, which even Google search engine is missing i.e Google can NOT search within a table of HTML document. We have proposed a novel architecture consisted of four layers which can modularize the activities. We have proposed to take care of the Web Form DOM as it will enable greater web information integration as discussed in our paper. Our Java based implementation used the classes URL, URLConnection, JEditorPane, HTML Swing Parser etc. To the best of our knowledge, such a integrated solution is not available anywhere although some other systems can only extract/export region of interest web data. Ours is an integrated solution, which we plan to extend further by the concepts as discussed in the paper.

ACKNOWLEDGMENT

The work is partially supported by Wayne State University, Computer Science Department conference, travel fund.

REFERENCES