Resource Capability Discovery/ Description Management System - Labeling Anonymous Dataset for Web Data Integration

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

Preliminary Thesis Proposal
Abstract

Formulating and executing queries over distributed, autonomous and heterogeneous resources is an important topic within e-Science in general and Bioinformatics in particular. The advent of the Internet and the Web and their subsequent ubiquity have brought forth opportunities to connect Biological information sources across all types of boundaries. Examples of such information sources include databases, XML documents, and other unstructured sources. Uniformly querying those information sources have been extensively investigated in this research. We have surveyed the current research on the fundamental problems to efficiently process queries of Web Data integration systems. Bioinformatics domain has been chosen to illustrate the query processing capabilities over the heterogeneous information sources. Although our application domain is Bioinformatics, our approach is not limited to a specific domain. We surveyed different approaches to data integration such as warehouse, mediator-based, navigational, vertical integration and horizontal integration. The ontology is also central to the query processing and reconciliation of heterogeneities. For efficient query processing, we advocate for the selection of relevant information sources and the use of meta data. We propose that the current data integration approaches should be enhanced by ontological knowledge and ISA relationships and inter-ontology mappings in ontologies should be explored. The transition of the World Wide Web from a paradigm of static Web pages to one of dynamic Web services provides new and exciting opportunities for Bioinformatics with respect to data dissemination, transformation, and integration. Integration of data and service from heterogeneous sources demand significant advances in middleware system, which we plan to implement.

The web document can be seen as a structure composed of different types of information units (such as forms, tables, titles and paragraphs). Increasing amount of data on the web highlights the need for its efficient and effective management. In this light, numerous researchers have proposed to use RDBMSs to store and query web data using the SQL language. Using relational database technology as a basis for storing and querying web data is a reasonable choice as this technology is well understood and known to have good performance.

Scientific workflows in Life Sciences are usually complex as they use many online databases, analysis tools, publication repositories and customized computation intensive desktop software in a coherent manner to respond to investigative queries. Most often, these investigative queries are ad hoc, and ill formed or even one time queries. In such cases, developing customized workflows becomes a major undertaking which is truly expensive, prohibitive and resource intensive. High development costs often act as deterrent to many interesting queries and thus act as barriers to promising ontime scientific discoveries. In this research proposal, we introduce a new paradigm, resource capability discovery/ description in complement to our new scientific workflow query language, called BioFlow, that exploits many recent developments in Internet communication, database features, wrapper and mediator technology, ontology, scientific workflow and web data integration. We will demonstrate that fairly complex web-based workflows can be effortlessly and declaratively expressed in an ad hoc fashion. We also report a prototype implementation of the system in Java that features most of the systems powerful features as a proof of feasibility of our approach.

We have been working toward developing a framework for automated information extraction from Internet documents. Specifically, we are developing an intelligent column name assignment algorithm for online table wrappers. In most web documents, information is generally displayed
in some form of table if it is generated from a structured database. Wrappers do a good job at identifying and extracting the tables, but, current technology does not support assigning meaningful column names to the extracted tables when they are missing. Such assignment functions become essential when automated schema mediation is needed and user intervention is not possible or desired.

In our lab, we are developing a new query language called BioFlow for autonomous querying of heterogeneous Life Sciences databases. BioFlow is the native query language of the LifeDB database system that we are building. The wrapper generation systems FastWrap and PickUp we are using will be exploiting the proposed method to assign label for the extracted information.

When linking information presented in documents as tables with data held in databases, it is important to determine as much information about the table and its content. Such an integrated use of Web-based data requires information about its organization and meaning i.e. the semantics of the table. The goal of giving a well-defined meaning to information is envisioned by the next generation web based systems, namely the Semantic Web. The traditional Legacy Dynamic web application is Web Form based, which have well defined Interface Schema (i.e form label tag, name-value input). Having got result pages in response to web form, the content information needs to be assigned to suitable labels. These labels can be found on the wrapped page or need to be derived. The process of finding and assigning these labels to values is called "Labeling". Here in this research, we surveyed some authoritative works in semantic labeling of web data, namely DeLA, Labeller component of RoadRunner, PANKOW, C-PANKOW etc to name a few. It is our claim that the earlier works are able to solve the present problem partially. We propose a three step approach to solve the problem of labeling. First we extract concepts from HTML web page (e.g meta tag and form label tag) as well as from user SQL query variable, secondly we formulate speculative query e.g Hearst Pattern queries to Web Search Engines (e.g Google, Yahoo and MSN), finally a ranked order list of labels for a set of instances are presented to Knowledge Engineer with the most relevant view of a statistical fingerprint in order to take a final decision. Previous work in this field rely on extraction ontologies, we eliminate the need for domain specific ontologies as we could extract concept from the HTML web page by itself as well as from user query variable.

We report here our experimental results on a number of domains e.g music, movie, watch, political, demographic, athletic obtained through different search engine such as Google, Yahoo and MSN. The comparative probabilities of attributes being candidate labels are presented which seem to be very promising, achieved as high as 93% probability of assigning good label to anonymous attribute. To the best of our knowledge, this is the first of its kind for label assignment based on multiple search engines' recommendation.

**Keywords:** Query processing, Query optimization, Web, Data Integration, mediators, metadata, Bioinformatics, Ontologies, Semantic Web, Web Services, wrapper, application programming interface, semantic matching, gene regulatory network, labeling, anonymous dataset.
0.1 Problem Description: Resource Capability Discovery and Description Management System

Despite numerous proposals for web services and semantic web technology, HTML is likely to be the primary vehicle of the information interchange on the Web at least for the next few decades. Organizations will publish and export their data in HTML to facilitate inter and intra-organization information sharing. Almost all the online data sources and tools in the Bioinformatics/Life sciences heavily depend on HTML and legacy web technology. Note that HTML is not just being used for publishing static documents, queryable information sources provide access to the data (through web form) they store by defining an external view of it in HTML. Ideally, the union of all these static HTML pages together with HTML views of queryable information sources will make the web viewable as an enormous HTML repository.

Unfortunately, this vision is a long way from reality, even the most sophisticated web search engine (recent Google Base beta version incorporated structured query) [39] only allows for keyword search. Keyword-based searching restricts the types of questions people can ask. For example, users cannot make requests like "Find me a Blue Honda Civic for under $20000 - it should be 2006 or newer and have less than 10K miles on it". The required information is out there on the web, but traditional search engines cannot answer this type of request because they do not know how to match the specified concepts in the request to data instances on the web. The reasons behind this are many fold. First, there is no uniform interface to both static documents and queryable information sources to date, research has focused either on searching/ querying static documents, or on searching/ querying queryable information sources, but not on a combination of the two. Second, the attempts that have been provided for unifying views over multiple queryable information sources do not scale well. These approaches simply do not extend to the scale of the present Internet, let alone the Internet of the future. In our thesis work, we propose to prototype and evaluate approaches that do present a scalable, unified view of the Internet’s HTML documents and queryable information resources. We propose an algorithm LADS for Labeling Anonymous Data Set, which can label/ annotate tabular web document. The algorithm has been tested on a number of domains, yielding very promising results. Details are discussed in the chapter Column Name Identification.

We believe that we have identified a promising approach toward solving these problems. First, in our previous work [16, 78, 18, 15, 14, 17, 32, 19, 28, 68, 35], we have implemented a number of prototype systems such as WebFusion, PickUp and OntoBuilder that is a much more scalable approach toward information integration than other approaches (discussed in the Related Work chapter). Now we are focusing on implementing resource capability discovery and description management system. It is more scalable because it will solve more targeted problems than most web integration works. It is our belief that this targeted problem provides most of the functionality
people want for Web Data Integration. We will discuss the system in detail in the chapter Overview of the On Going Work. Second, as part of the LifeDB project, we have been a member of the team that is implementing web data integration on the fly using ad hoc query. We believe that our research work will benefit the LifeDB project to a great extends.

Our goal and immediate focus of the research is to build a system that is able to determine site capabilities. By capability we mean (i) what a site is about, (ii) what kind of resources it can provide, (iii) what it needs as inputs to provide those resources, (iv) how does it expect those inputs to be provided, (v) what specifics the requesting sites may need to know and follow to get access to those resources, (vi) is it part of a larger resource provider, and (vii) what are the relationships with the other resources. By resource we mean data. By data we mean structured data, possibly represented in some unstructured way.

Note that, in the above definition, web form is not a resource. It is a means to get access to the resource. The same applies to hyper links, it is not a resource, it is means to get access to the other resources. Capability means a machine understandable description of sites input/output behavior and its interrelationship with other resources. Interrelationship can be also within the site.

We will need systems and languages that can be used to extract the capabilities of arbitrary sites and expressed in that language, may be even RDF, OWL or RuleML. Then we need a system that can use these descriptions for the individual sites to create interrelationships, also to facilitate two operations: link and combine. We will also need a system to map queries to sites that best matches the query evaluation goals and the information needs. We need to demonstrate that given a description \( \mathcal{D} \), a query \( Q \), \( \text{map}(Q, \mathcal{D}) \) is a number between 1 and 0, such that for any other description \( \mathcal{D}' \), \( \text{map}(Q, \mathcal{D}) > \text{map}(Q, \mathcal{D}') \). We also need to show that \( A = \text{eval}(Q, S) \) is correct and intuitive where \( \text{eval} \) is an evaluation function that computes the query \( Q \) at site \( S \) corresponding to description \( \mathcal{D} \) to obtain the response \( A \). To show this correspondence we will need to establish that given a database \( d \), the query \( Q \), and the response \( A' \) of \( Q \) on \( d \), \( A' = A \) always holds if \( \text{semantics}(S) \) is equivalent to \( \text{semantics}(d) \). Furthermore, we want to show that \( \text{semantics}(\mathcal{D}) \) is equivalent to \( \text{semantics}(d) \).

To the end, to complete my dissertation research, we propose to investigate the following research problem:

- To treat traditional legacy dynamic web site as web service, as Web form takes some user input and produce result, may be structured/unstructured
- To devise algorithm for Web site composition, whether two sites are composable or not
- Devise algorithm for Web source selection to answer queries, i.e a subset of all the sites will be selected based on our mapping function \( \text{map}(Q, D) \), i.e structural comparison between a source and a target schema using DB and IR technique
• Devise algorithm to map user query to site query

• Develop and implement a system which takes SQL like user query and return result. We proposed an algorithm LADS in this regard

To demonstrate our concept and running example, we will be using the following sites from Bioinformatics Life Science Domain:

• www.flybase.org (A Database of Drosophila Genes and Genomes)

• http://wingless.cs.washington.edu/YMF/YMFWeb/YMFInput.pl (YMF 3.0: Finds short motifs in DNA sequences)

• http://atlas.med.harvard.edu/cgi-bin/alignace.pl (AlignACE 3.0: Find short motifs in DNA sequences)

• http://www.benoslab.pitt.edu/stamp (Alignment, Similarity and Database Matching for DNA Motifs)

To achieve this, first we will develop data model for describing the web, second we will investigate the feasibility of using the model for mapping query to sources based on the use cases identified from real-life Bioinformatics scientific workflow; finally limitations of the system’s query processing capability will be identified and insights into the requirements for an ad hoc query language, query processing, and optimization will be reported. It is our argument that existing relational database technology can go a long way toward supporting such a system. To the best of our knowledge, this is the first attempt of using web’s intra-document to store and query data in scientific workflows.
## Contents

0.1 Problem Description: Resource Capability Discovery and Description Management System .......................... 3

1 Introduction ......................................................... 8

1.1 Problem Statement/ Description .................................. 10
1.2 Motivating Example ................................................. 12
1.3 Web Knowledge ..................................................... 13
    1.3.1 Types of Web Knowledge Sources .......................... 13
    1.3.2 Query Capabilities of Knowledge Sources ................. 13

2 Related Work ......................................................... 14

2.1 DeLA ............................................................... 16
2.2 Labeller component of RoadRunner .............................. 17
2.3 ViPER ............................................................... 17
    2.3.1 Inter Column Label Assignment ............................ 17
    2.3.2 Inner Column Label Assignment ............................ 17
2.4 ADeaD .............................................................. 18
2.5 ADEL ............................................................... 19
2.6 Multi Annotator .................................................... 19

3 Progress Report: Overview of the On Going Work ................. 20

3.1 Resource Description ............................................. 21
    3.1.1 Table Definition ............................................. 21
    3.1.2 Process Declaration ......................................... 22
3.1.3 Ontology Definition .................................................. 22
3.1.4 Computable Function Definition ................................. 25
3.2 Describing and Computing Capabilities of Sources and Mediators ........................................... 26

4 Progress Report: Column Name Identification ................. 28
   4.1 Labeling ................................................................. 28
      4.1.1 Language Patterns ............................................. 29
      4.1.2 Labeling Algorithm .......................................... 30
      4.1.3 Probabilistic Labeling of Anonymous Data ............... 33
      4.1.4 Candidate Label Selection ................................. 34
      4.1.5 Affinity-Based Labeling .................................... 35
      4.1.6 Speculative Labeling ...................................... 36

5 Proposed Work and Possible Future Work ......................... 43

6 Conclusions ............................................................. 46

Bibliography ............................................................. 48
Chapter 1

Introduction

In literature, data extraction techniques for HTML and semi-structured data in general have been exhaustively studied and a number of automatic and semi-automatic approaches proposed. However, in real-life scenarios data extraction capabilities are only half of the game. Password protected sites, cookies, non-HTML data formats, JavaScript, Session ids, Web Form iterations and dynamic changes on Web sites are the obstacles that make Web data extraction difficult in real-life application scenarios. [38]

Most of the existing web data extraction systems cannot assign field labels to the extracted data records. Recently, some automatic approaches to assigning the semantic meaning for data has been proposed. Most of the systems are heuristic based and can solve the problem partially. Here in our research, we wish to develop a complete framework for data annotation. IE and labeling/annotation are two separate process. Note that we are not working on wrapper generation at all, the wrapper generation technology is mature enough and a lot of works have been done in this field. But to the best of our knowledge, only a very few works have been done in terms of column name identification and data annotation, which is a classification process. We will show that our labeling process yield better results on structured data such as web page table.

The database community has devoted a large amount of work on integration of data either materialized within data warehouses or non-materialized through mediation systems. For an exhaustive list of Data Integration projects world wide, refer to [80]. However, the quantity of data sources made available and their significant increase explain the need for non-materialized access to Biological data. [45]

There is a high demand for collecting data of interest from multiple Web databases. For example, a comparison-shopping system (e.g shopping.com) needs to collect the price, availability, and other information of the same product from multiple providers. Such kinds of applications require that the collected data be semantically labeled so that they can be appropriately organized/ stored for subsequent analysis.
Despite the Web wrapper’s long track record, automatic labeling of extracted data has only recently begun to be addressed. Since wrappers are built automatically, the values that they extract are anonymous and a human intervention is still required to associate a meaningful name to each data item. The automatic annotation of data extracted by automatically generated wrappers is a novel problem, and it represents a step towards the automatic extraction and manipulation of web data. [6] An urgent need in Web Data Integration is the discovery of suitable resources and the marshalling of those resources to work together to perform a task. [58]

To minimize user effort in an IR process and enable tools to scale with the growth of the web, we explore the problem of automatically interacting with online information sources in the hidden web. This problem has four aspects:

- **Information Discovery**: How to automatically locate the web sites containing structured data of interest to the user?

- **Information Extraction**: How to induce wrappers to extract relevant data objects from discovered web sources?

- **Information Understanding**: Having extracted data objects with complex structures, how to automatically or semi-automatically annotate or label the fields of the extracted data?

- **Information Integration**: How to integrate the various data objects from multiple sources with or without knowing their schemas?

In this research we concentrate our investigation on the Information Understanding aspect i.e how to annotate or label the fields of the extracted data. This is a novel research problem and it still waits for a good solution.

A lot of works have been done with regard to web data extraction and wrapper generation. But most of the works export the web data as anonymous dataset, without assigning meaningful label to those. To the best of our knowledge, there are only two generic methods for the automatic labeling of anonymous data: DeLA (Data Extraction and Label Assignment) and Labeller component of RoadRunner. The previous approaches to labeling have two drawbacks. First, typical Web pages often omit labels, which are understood from the context by a human. Second, this approach restricts one to using only those labels chosen by the Web content providers, which may not be the most appropriate or most descriptive ones. Our method, on the other hand, does not suffer from either problem. Our probabilistic model estimates the appropriateness of a label regardless of where it come from, allowing the user to provide her own set of labels.

We propose a novel and highly effective method for automatically labeling anonymous dataset based on a simple probabilistic model that takes into account the affinity between a set of values (i.e an anonymous attribute) and potential attribute labels. Therefore the anonymous dataset
can be materialized into a suitable relational DB [4]. The probabilities are estimated by counting the number of answers to speculative queries, obtained from popular Web search engines such as Google, Yahoo and MSN. Estimating probabilities based on hits count is referred to as Web Statistics. Intuitively, a speculative query formulates a hypothesis that a given term is a good label for an attribute in the anonymous dataset. The search engines are used as an oracle to determine how plausible such a hypothesis is. Our approach is to search the Web for documents containing certain text patterns (Hearst) commonly used to enumerate instances of classes of objects. We exploit these patterns to mine frequently occurring terms that can be used as labels.

The present research work is our ongoing work for developing automatic techniques for labeling attributes in a page, where we have identified and proposed ontology based and declarative workflow query language for ad hoc web data integration on the fly [5]. In our previous work [16, 78, 18, 15, 32, 28], we have implemented a number of prototype systems such as WebFusion, PickUp and OntoBuilder that are much more scalable approach toward web data integration.

1.1 Problem Statement/ Description

Let us introduce an example to represent the overall context and the issues we are trying to address. Consider the HTML page in response to a web form from Flybase site shown in Figure 1.1. Feeding our wrapper generator system with a set of sample pages like these, we obtain a grammar description of the common template as follows:

\[
\text{<html>... (<b>Name</b>$J<br><b>Annotation symbol</b>$K<br>...<b>FlyBase ID</b>$Q<br><b>Sequence location</b>$R>...\text{/html>}
\]

This grammar works as wrapper by considering the non-terminal symbols ($J, $K, $Q, $R ...) as placeholders: during the parsing of the input pages, they match with the strings to extract. Data extracted from the page by this wrapper are shown in Table 1.1. The extracted data are anonymous, whereas we observe that useful labels are available on the pages themselves as part of the common template. For instance, the boldface "Name" compares nearby the extracted value "Sin3A", "Sequence location" is close to "2R:8462743..848039", "2R:3679403..3680885" and so on. Many of the anonymous fields extracted by the wrapper could be named after some terminal symbol of the grammar by itself. For example, the terminal symbol Name could be the label for all the values of $J (corresponds to anonymous attribute $A_3)$. One of the goal of the labeling problem that we are going to address is that of analyzing the output of the web wrapper in order to locate inside the common template meaningful labels for the anonymous datasets. We notice that the labels $L$ are invariant among pages, whereas the data values $V$ are variant. The labeling problem is that of associating to each variant a meaningful label chosen among the invariants.

In order to achieve interoperability among deep web sources, deep web interfaces need to be wrapped. In case of automatic wrapper design, automated semantic decisions (i.e. identification of label-value relationships) need to be supported by the wrapper generator. We identify three main problems: Finding of Parameter values, Labeling and Relabeling containing the question on how to identify relations between labels and values, or more in general between concepts and instances or among concepts.
Labeling Having got result pages, the content information needs to be assigned to suitable labels. These labels can be found on the wrapped page or need to be derived. The process of finding and assigning these labels to values is called "Labeling". A severe limitation of the previous methods as reported in the literature are their dependence on finding the label in the web page itself, which may not be the case in most cases such as search engine result pages, amazon.com book search pages etc. Therefore we wish to solve the Labeling problem by three step process consist of:

- Extracting concepts from HTML web page (from Form and meta tag) as well as from user SQL query variable
- Extracting concepts from output Result pages (we developed a set of heuristics for column name identification)
- By using Web Knowledge such as WikiPedia and Web search engines, we could assign label for the anonymous dataset by making use of Language pattern and speculative queries and labeling.
<table>
<thead>
<tr>
<th>Table 1.1: Subset of the Genes Experimented with Flybase Site</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A1</strong></td>
</tr>
<tr>
<td>CG8815</td>
</tr>
<tr>
<td>CG12055</td>
</tr>
<tr>
<td>CG6871</td>
</tr>
<tr>
<td>CG17903</td>
</tr>
<tr>
<td>CG7070</td>
</tr>
<tr>
<td>CG4581</td>
</tr>
<tr>
<td>CG7176</td>
</tr>
<tr>
<td>CG10120</td>
</tr>
<tr>
<td>CG3476</td>
</tr>
<tr>
<td>CG2107</td>
</tr>
<tr>
<td>CG12653</td>
</tr>
<tr>
<td>CG15321</td>
</tr>
<tr>
<td>CG15319</td>
</tr>
<tr>
<td>CG8316</td>
</tr>
</tbody>
</table>

1.2 Motivating Example

The problem of labeling degenerates to finding concepts extracted from meta tag, web form label as well as from user SQL query variable, then associating the extracted concept label to attributes in anonymous dataset extracted by Web wrappers. Once the concepts have been identified, then assign the concepts to the output data to assign column label. The problem is how to relate a concept to the instances of output web page table? In other words, given a set of values (of same domain), how to come up with a suitable name/ concept for the values? Work has been done to assign Form label to table column name. Idea is if a keyword search associated with a Form label appear in a column, the column can be labeled the same as Form label. Work has been done to use Web knowledge such as WikiPedia to enrich knowledge like Shrek is a Movie. ("Movie", "Shrek"), where Movie is a concept and Shrek is an instance of the concept. We need to develop a system which can label a set of concepts, not just a single concept, like we need to be able to recognize a number like 4.8 as price of a movie ticket, as well as 48202 as Zip code of the Cinema complex. Example: Given {Toyota, GM, Ford, Fiat, Honda}, assign label of the data as make. Another example: Given {2000, 1999, 1980}, assign label of the data as year. More example: Given {Apple, Orange, Banana}, assign label of the data as fruit. More example, given, the set containing {India, Pakistan, USA, UK}, then it has to be labeled with the category country name. Figure 1.2 shows our overall envisioned system, resource capability discovery/ description management system. Labeling component is the core of the system. In response to the Web form, we will have tabular data. Semantic enrichment of the tabular data will be made semi- automatically and through human knowledge. Tabular schema will also be stored in the resource description repository. Thus our resource capability discovery/ description will be focused toward handling web form and representing table schema. The rest of the report deals with based on the above two ideas.
1.3 Web Knowledge

1.3.1 Types of Web Knowledge Sources

We categorize the Web knowledge sources into the following: Web Encyclopedias (e.g WikiPedia, Wikitionary, Wikibooks and Wikinews); Search Engines/ Web Directories (e.g Google, Yahoo, MSN, Altavista); Web Thesauri (e.g http://freethesaurus.net)

1.3.2 Query Capabilities of Knowledge Sources

By querying web knowledge sources, we wish to capture concept given some instances. The main application in the wrapping context is the labeling problem - a potential example instance has been extracted and the knowledge source is used to deliver a suitable concept. Queries of Instance → Concept type are restricted to knowledge sources, which contain instances. An example for the application of Google is a query containing a specific ISBN of a book and the result of the concept name "ISBN".
Chapter 2

Related Work

WebSQL is a declarative query language for extracting information from the web. The language emphasis is on extracting connectivity information from pages (e.g. pages that are two clicks away from Flybase site). WebSQL regards HTML documents as monolithic objects, and its analyses are limited to simple text matching techniques. [7]

ExDB is an "extraction database" that extracts structures from web text and supports structured queries over them. Using IE systems that are domain-independent and unsupervised, such as KNOWITALL, it extracts data values (e.g. "Einstein", "Switzerland"), binary relationships (e.g. "Einstein" was born in "Switzerland") and semantic types (e.g. "Switzerland" is a country). The tuples are loaded into a probabilistic database, which records for each tuple the probability of that tuple being true. ExDB supports structured probabilistic queries that use a Datalog-like notation.

Google’s PAYGO is a data integration architecture intended to manage structured data on the Web scale. Therefore, it has to model any kind of structures, which can come from any domain (e.g. school, government, sports etc) and from any source (e.g. queryable HTML forms in the Deep Web, Flickr, Google Base etc). PAYGO is based on the philosophy that a system should be able to incrementally evolve its understanding of the data.

Web-at-a-Glance (WAG) [13] is a system to assist the user in information retrieval and discovery by gleaning the most relevant information from a web site or several web sites. WAG can be realized as an active index system. WAG consists of three components: searcher, page classifier and the conceptualizer. A prototype active medical information system has been built with WAG as a component.

[52] reports WebDB, which is based on object-relational concepts. WebDB allows users to access web’s intra-document structures, such as tables, forms etc.

[22] motivates "data-centric" approach to web search which can support complex structured queries (e.g. "bed and breakfast units within one mile of the beach in Hawaii costing less than $150 per night"). The basic idea to solve the problem is first to convert the web text to a multi dimensional vector representation, where dimensions corresponds to words in the text, then to use machine learning or statistical techniques (e.g decision trees, naive Bayes) to classify the vectors in the multi dimensional space. This in turn requires to write site-specific "wrappers" that extract information based upon regularities of the HTML tag structure that is present in a single web site.
Their work also report the WHIRL integration system. [23]

Query Tunneling has been proposed in [42] in MiWeb prototype system where the user will not only be able to query on Interface Schema, but also on the Result Schema. Wrapping of sources by Query Tunneling hides restrictive query interfaces and makes such sources fully queryable based on their Result Schema. The process of Query Tunneling is divided into two main steps, Query Relaxation to make a higher order query suitable to a restricted interface and Result Restriction in order to filter the result using the original query. Web sources that are accessible through small query interfaces has been wrapped in order to support higher level query capabilities. Query relaxation procedure either substitute (for example synonym) or eliminate attributes (e.g user query variable need not be mapped to web form).

[33] surveyed the semantic enterprise information integration and the need for describing the meta-data across an enterprise.

[8] argues that most of the Bioinformatics concept can be represented by a conceptual model. However, most of the data sources in Biology are not that simple and are deeply nested. For example to put SWISS-PROT database into a relational database in the 3NF would require to break every SWISS-PROT record into nearly 30 pieces in a normalization process. [75]

Automatic web form filling and data extraction from web has been deeply investigated in the work of [31, 53, 29, 30, 26]. They propose to use extraction ontologies for domain such as car site. [47] presented a survey on web data extraction tools. They also report a prototype implementation of a system, DEByE (Data Extraction By Example) [46]

[43] described a logical framework for web service discovery and report a F-Logic implementation using Flora-2 system.

[70] reports Squeal, a structured query language for the web, where user can query the web as if it were in a standard relational database.

[49] presents WEBLOG that is capable of retrieving information from web. WEBLOG is inspired by SCHEMALOG, a logic for multidatabase interoperability. [44] presents W3QS, a SQL-like high level language for accessing the web. A W3QL query specifies a graph to be matched with portions of the www (graph nodes corresponds to www pages, edge nodes to hyperlinks).

[12, 11] report INDUS (INtelligent Data Understanding System), a Federated Ontology-Driven Query Centric system.

[9] introduced BioFast that deals with linked life science sources.

[36] report WISE system which can integrate complex Web search interfaces of the deep web.

[25] report Winagent which is like a softbot for collecting hidden web contents.

[61, 62] report TARTAR which can transform arbitrary tables into logical form.

[48] report automatic generation of agents for collecting hidden Web pages for data extraction.

[76] report a semantic approach to Internet tabular information extraction. [20] report PANKOW and C-PANKOW. Their work annotate a Named Entity (NE) by using linguistic pattern, usually called Hearst pattern [37] and by using search engine. They showed that automatic annotation using Google is a useful approach.

[73] presented Pointwise Mutual Information (PMI-IR), a simple unsupervised learning algorithm
for recognizing synonyms, based on statistical data acquired by querying a Web search engine.

[10] described the WebTables system and introduced attribute correlation statistics database AcsDB. [64] used Google for semantic annotation of Biomedical Literature. They obtained semantic annotation accuracy of 52% on words not found in the Brown Corpus, Swiss-Prot or LocusLink.

Turning the current web into a semantic web requires automatic approaches for annotation of existing data since manual approaches will not scale in general. [61] have shown an approach for automatic table data annotation. They have extended the work in [62]. 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: [31], [53], [79], [48], [76], [57], [55], [54], [56], [51], [63]. Researchers have addressed the problem of extracting data from Web sites, however, very little work on the semantic labeling of the extracted content has been made. In this section, we briefly discuss some of the authoritative works in this field.

### 2.1 DeLA

In Data Extraction and Label Assignment (DeLA), the main idea of the labeling process is that "form elements will probably re-appear in the corresponding fields of the data objects". To assign labels to the columns of the data table containing the extracted data objects, i.e. to understand the meaning of the data attributes, [74] employed the following four heuristics:

**Heuristic 1:** Match form element labels to data attributes The search form of a web site through which users submit their queries provides a sketch of the underlying relational database of the web site. If we make the assumption that the web site designers try their best to answer user queries with the most relevant data, keyword queries submitted through one specific form element will re-appear in the corresponding attribute values of the data objects. Therefore, for each form element with its keyword queries, if the keywords mostly appear in one specific column of the data table, then we can assign the label of that form element to the column.

**Heuristic 2:** Search for voluntary labels in table headers The HTML specification defines some tags such as `<TH>` and `<THEAD>` for page designers to voluntarily list the heading for the columns of their HTML tables. Moreover, those labels are usually placed nearby the data objects. Therefore, the HTML code near (usually on the top of) the contained data objects is examined for possible voluntary labels. We need to augment header using an ontology i.e representative terms of a domain. In many situations the table header alone is not enough to describe the semantics of that table. Another approach is then used, namely: the table header is used to extract the corresponding concept from the domain ontology and search for that concept and all related synonyms in the text.

**Heuristic 3:** Search for voluntary labels encoded together with data attributes Some web sites encode the labels of data attributes together with the attribute values. Therefore, for each column of the data table we try to find the maximal-prefix and maximal-suffix shared by all cells of the column and assign the meaningful prefix to that column and the meaningful suffix to the column next to that column as the labels.

**Heuristic 4:** Label data attributes in conventional formats Some data have a conventional format, e.g. a date is usually organized as "dd-mm-yy", "dd/mm/yy", email usually has the symbol "@", price usually has the symbol "$", etc. Thus, such information is used to recognize the corresponding
data attributes. Note that the form elements and the data attributes do not need to be perfectly matched. Therefore, the label assigner may not be able to assign meaningful labels to all of the data attributes. DeLA also allow users to add a label to unassigned attributes and to modify the assigned labels. DeLA demonstrated the feasibility of heuristic-based label assignment and the effectiveness of the employed heuristics, which sets the stage for more fully automatic data annotation of web sites.

2.2 Labeller component of RoadRunner

Since web pages are designed to be presented on a browser to a human user, usually values and labels are visually close to each other. Therefore, first LABELLER computes the coordinates of the bounding boxes of every data-value and every label in a given sample page. Then it tries to find the optimal association label/ data-value by analyzing their spatial relationships. Arlotta et al have [6] developed several heuristics to establish the correct associations:

- values and labels are close to each other
- usually a label is vertically, horizontally, or centrally aligned to its associated values
- labels are usually placed either to the left or above values
- it is not allowed that either a label or a value is between another value and its label

The wrappers produced by [24] need a post-processing phase to annotate with more semantic labels the extracted attributes, which are initially anonymous

2.3 ViPER

Visual Perception-based Extraction of Records (ViPER) mainly developed two heuristics: inter-column label assignment heuristics and inner-column label assignment heuristics to assign column label to data items. [67]

2.3.1 Inter Column Label Assignment

[67] considered three types of fix column inter relations which refer to label assignment strategies: left-to-right, right-to-left and in case of vertical orientations up-to-down assignment. Given a fix column $C_i$ with data item $s$ and its predecessor column $C_{i-1}$ and successor column $C_{i+1}$, it test whether there exists data items from column $C_{i-1}$ or $C_{i+1}$ which are rendered in the same horizontal axis with respect to the bounding box of $s$.

2.3.2 Inner Column Label Assignment

Columns are scanned for fix non-numerical sub-tokens appearing in each of the merged token trees. Suppose a field value appear as ”Save: $10.00 (13%)”. The fix token here (save:) which is not
numeric type become removed from individual rows and added as label.

In addition to the above two heuristics, [67] also advocate for column splitting, basically based on repeated prefix string. We may also use ontologies to improve text segmentation. For example, if we know that the term "Canon" appears only in the "Manufacturer" field and "8MP" only in the "Megapixels" field, we can split "Canon X300 8MP" into two text segments, thereby adding more semantics on the labeled data.

2.4 ADeaD

ADeaD (Data Extraction and Annotation) is a system to automate the data extraction from web pages and assigning the text of semantic meaning for each data element. It consists of four implementation components:

- **Wrapper Generation** Two or more original pages are randomly selected from Web page set as samples. Based on tag tree comparison, system deduces data schema and page template, and then generates wrapper program. Here only the data structure can be obtained, without semantic meaning for the data elements.

- **Data Extractor** With extraction wrapper, the data extractor extracts data from Web pages. These data are organized in data schema.

- **Annotation Candidate Generator** Page template contains HTML tags and text. Most of the text is indicative of the semantic meaning of the adjacent data elements. They are picked out as annotation candidates.

- **Data Annotator** [69] developed several heuristics to link text candidates with data elements. Varied page templates required different methods to establish association.

In the following paragraph, we only discuss its data annotation component in detail. Annotating data extracted from automatically generated Web pages relies on the observation that important information about the semantics of data is often available on the Web pages themselves. Annotation text is visually close to the data element. There are two kinds of annotation text with the data value. The first is that the annotation text is adjacent with the data value in the depth-first traverse of the page tree. Another pattern is a tuple of annotation text following with multi data value tuple instance, for example, table format. [69] developed the following heuristics to compute the association possibility of the annotation candidate with each data element:

- The annotation text and the data elements are direct close to each other, in vertical or horizontal.
- The text either to the left or above the data elements have high priority.
- If an annotation tuple is followed by a data element tuple, they must have length.
2.5 ADEL

Automatic Data Extraction and Labeling ADEL [50] system can automatically extract records from web sites and semantically labels the fields or mapping them to a schema. ADEL achieves precision of 64% and recall of 89% on extracting and labeling data columns. [50] used learned patterns to map columns to data fields. The basic premise is to check how well a field describes a column of data, given a list of patterns that describe the data field and its mean length. [50] have developed a set of heuristics to score how well a field describes a column. The column is assigned the field with the highest score. Factors that increase a field’s score include:

- Number of patterns that match examples in the column
- How close examples are in length to the field’s mean length
- Pattern weight - where the more specific patterns are given higher weight

There are some shortcomings for the ADEL system, such as: ADEL is not able to process forms, instead it manually retrieve list pages and provide them to the system. ADEL manually segment fields on \r\n i.e carriage return and line feed character as field separator. More sophisticated field separator is required. In case of record boundary identification, we can select the symbol (non-letter and non-digit) with the most occurrences as the separator. The most significant challenge for automatic labeling is to deal with inconsistent data formats. ADEL failed in several cases due to lack of data transformation rule such as "sedan 4dr" and "4dr sedan" refer to the same concept. Other differences included such as "Exterior" and "Color" refer to the same concept. All these exceptions advocate for our proposal to use of ontologies in the labeling process. [3, 72]

2.6 Multi Annotator

[59] proposed a multi annotator approach to tackle the annotation problem with each basic annotator exploiting a different type of features. Multi annotator approach first aligns the data units into different groups such that the data in the same group have the same semantics. Then for each group, they annotate it from different aspects and aggregate the different annotations to predict a final annotation label. [59] defined 6 basic annotators to label data units, with each of them considering a special type of patterns/ features: Table Annotator; Query-based Annotator; Schema value Annotator; Frequency-based Annotator; In-text prefix Annotator and Common knowledge Annotator. Some of the basic annotators they use are also used by DeLA. However, their work differs significantly from DeLA. Firstly, the data alignment method is not based on HTML tag tree. Instead, they utilize new features that can be automatically obtained from the result page including the content and data types of the data units. Secondly, it uses both the LIS as well as an integrated interface schema of multiple Web databases of the same domain, whereas DeLA uses only local interface schema. Third, they use a probabilistic model to combine the results of different annotators while it is not clear how the heuristics in DeLA are combined. They report that every annotator contributes positively to the overall performance of labeling. They also illustrated how the use of the integrated interface schema can help alleviate the local interface schema inadequacy problem and the inconsistent label problem.
Chapter 3

Progress Report: Overview of the On Going Work

The current research is our ongoing work from [5] where we proposed and identified a model for web data integration. As we have stated above, the current focus of our research is to describe a web page so that somebody else can later use it 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 [1, 2]:

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;

The current prototype of the system consists of three primary components: the global schema provides a standard view of the information, the resource description defines the view of the information sources using ontologies defined in the global schema and describes how to interact with the sources for extracting information, and the query engine processes user queries in standard SQL. The global schema is a collection of ontologies. An ontology is a grouping of terms (or vocabularies) describing a concept.

The system interacts with information sources using resource description. The role of a resource description is two fold: describing contents and capability of a source in terms of the global schema and providing information how to extract data from the source [27]. The resource description guides the query engine to extract information from the source by providing regular tag expression that
map the set of local data into the tuples of the global schema. For example in order to find a gene sequence from flybase site, we know the DNA sequence pattern \( A, C, G, T \). It is very uncommon that four consecutive letter like DNA alphabet will be anything else other than gene sequence, in other words no plain English word could be misinterpreted to be start of a gene sequence. It may be mentioned here that, we are assuming that we do not have direct access to database in Flybase site, we only rely on what the site gives us in Decorated Fasta Format in response to a web form.

The resource description can use terms from one or more ontologies in the global schema. It need not conform to any namespace nor have any restriction on choosing a set of vocabularies from various namespaces. This allows the user to describe sources as close as possible to the original semantics of contents of information sources and avoids losing or adding information due to a lack of expressive freedom. A user query is formulated over the global schema and the system processes it using the resource description.

### 3.1 Resource Description

Broadly, resources can further be classified into four categories - data, ontologies, processes, and computable functions. All four categories of resources can reside on a local machine or at a remote site, all connected by a network and are assumed to be accessible (we note that some web resources may not be accessible due to access and authentication issues, as well as some web form can not be automatically filled due to JavaScript validation, password protected web site, hidden form fields etc). BioFlow recognizes two types of data - relations or tables, and XML. Consequently, it can interchangeably manipulate and store tables and XML data on local or remote machines. However, while the input and output format of the data resources can be a table or XML, the internal view of data resources is completely relational (first normal form relations). The following few subsections have been adapted from [28].

#### 3.1.1 Table Definition

Similar to SQL’s create table statement, BioFlow offers creation and use of data resources using create database statement as follows. In a way similar to SQL, alter table, drop table and create index on table statements are also available. In the statements below, the format clause specifies the representation format of the resource and optionally declares the location of the resource. The scheme of a data table is always in first normal form when format table is used, otherwise, it can be arbitrarily nested as shown. Table format is assumed in a local machine if none specified (default).

```sql
create datatable employee
name varchar(60);
age integer [format table [at URL]];

create datatable flight
origin string;
destination string [format XML [at URL]];
```
Once created from web as part of a database, these web resources can be accessed just by using the database. A database is defined using the following statement. If the database exists already, it will be opened and all its resources will be accessible, it will be created otherwise. Likewise, access to the resources in a database can be closed by the close database statement as shown below.

```
open database student;
close database student;
```

Each BioFlow program has the following general structure.

```
open database d;
resource description statements;
begin
data manipulation statements;
end
close database d;
```

As we mentioned before, the resource capability discovery/description management system will complement the task of BioFlow, we note that the resource description statements part of the general BioFlow structure conform to generic database data definition language (DDL).

### 3.1.2 Process Declaration

BioFlow allows reuse of previously defined and stored process definition. Previously defined and stored processes can be included in a BioFlow program using the following simple include statement. Processes $p_1, p_2, \ldots, p_k$ are assumed to be accessible having no conflicts with the other resources in the program.

```
include process $p_1, p_2, \ldots, p_k$;
```

A process in BioFlow is a self-contained unit of computation or a BioFlow program that can be compiled, executed and stored separately, and when needed can be retrieved or included in other programs or processes. Thus each process is uniquely named, has its own resource description and data manipulation components, and executes in separate independent process threads.

We note that traditional database stored procedure concept is not enough for the web. We will need site specific wrappers to convert web pages into structured relation and a series of user defined function (UDF) like in our system process2Kup, process2Kdown etc to clean up the messy web data and convert those into process definition.

### 3.1.3 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 use by the query engine. An ontology mainly captures three types of relationship: *synonym*, *isa* and *partof* [72]. 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 \( m : \text{URL } u \rightarrow a_1 : b_1, \ldots, a_k : b_k \)
c\(_1\#d_1, \ldots, c_m\#d_m, e\(_1\@f_1, \ldots, e_n\@f_n; \)
wrapper \( w : \text{URL } u \rightarrow a_1 :: r_1, \ldots, a_k :: r_k; \)
```

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*.

### Table 3.1: GeneMapping

<table>
<thead>
<tr>
<th>CG Number</th>
<th>FBgn Number</th>
<th>F/R</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG12055</td>
<td>FBgn0001091</td>
<td>R</td>
<td>2R:3679403..3680885</td>
</tr>
<tr>
<td>CG6871</td>
<td>FBgn0000261</td>
<td>F</td>
<td>3L:18815706..18821294</td>
</tr>
<tr>
<td>CG4769</td>
<td>FBgn0035600</td>
<td>F</td>
<td>3L:5353290..5355898</td>
</tr>
<tr>
<td>CG17903</td>
<td>FBgn0000409</td>
<td>F</td>
<td>2L:16719874..16722659</td>
</tr>
<tr>
<td>CG7070</td>
<td>FBgn0003178</td>
<td>F</td>
<td>3R:18193234..18198464</td>
</tr>
<tr>
<td>CG4581</td>
<td>FBgn0025352</td>
<td>R</td>
<td>2R:19754124..19756139</td>
</tr>
<tr>
<td>CG7176</td>
<td>FBgn0001248</td>
<td>R</td>
<td>3L:8349503..8354628</td>
</tr>
<tr>
<td>CG10120</td>
<td>FBgn0002719</td>
<td>R</td>
<td>3R:8538819..8548267</td>
</tr>
<tr>
<td>CG3476</td>
<td>FBgn0031881</td>
<td>F</td>
<td>2L:7027594..7028959</td>
</tr>
<tr>
<td>CG2107</td>
<td>FBgn0035383</td>
<td>F</td>
<td>3L:2981023..2983555</td>
</tr>
<tr>
<td>CG6050</td>
<td>FBgn0024556</td>
<td>F</td>
<td>2R:9395063..9396838</td>
</tr>
</tbody>
</table>

The above ontology shows mapping between 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, l(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, l(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 as multivalued dependency [66] as follows:

<table>
<thead>
<tr>
<th>Generalized Term</th>
<th>Synonym Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBgn0024556</td>
<td>EfTum</td>
</tr>
<tr>
<td>FBgn0024556</td>
<td>CG6050</td>
</tr>
<tr>
<td>FBgn0024556</td>
<td>l(3)L4569</td>
</tr>
<tr>
<td>FBgn0024556</td>
<td>mtEF-Tu</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>GAPDH</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>BEST:GH12586</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>CG12055</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>G3-P dehydrogenase</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>GA3PDH</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>GADPH</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>Gapd</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>GAPDH I</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>GAPDH-1</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>GAPDH</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>Gapdh43E</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>gh12586</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>glyceraldehyde 3 phosphate dehydrogenase1</td>
</tr>
<tr>
<td>FBgn0001091</td>
<td>Glyceraldehyde-3-phosphate dehydrogenase1</td>
</tr>
</tbody>
</table>

There are three types of ontologies - informal, semi-formal and formal. Many real-world ontologies may be described as semi-formal ontologies (as opposed to formal ontologies). Semi-formal ontologies are ontologies that may be populated with partial or incomplete knowledge, may contain occasional inconsistencies, or occasionally violate constraints (e.g. all schema level constraints may not be observed in the knowledge base that instantiates the ontology schema). There is a very large body of work that can and needs to be done using semi-formal ontologies.

GO ontology [40, 71, 58], which is 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"* [34]

Current data integration approaches should be enhanced by ontological knowledge and ISA relationships in ontologies should be explored. [41]
3.1.4 Computable Function Definition

Computable functions can be an executable program, called a program function, stored in a local or remote machine, or a web form, called a form function, in a remote site all with a URL. In general, all functions (program functions or form functions) are named, accessible at a URL, have a defined set of ordered name-value input parameters, and a set of n-ary output values for each m-ary input. Program functions stored at a local or remote machine can be declared using the following user defined function, or UDF, declaration. The UDF can be any executable program in high level languages such as C/C++ or Java. The requirement these functions must satisfy is captured precisely in the declaration below.

\[
\text{define function} \quad \text{price} \\
\text{extract} \quad x \text{ integer}, \ y \text{ integer}, \ z \text{ float} \\
\text{from URL} \ u \\
\text{submit} \ (a \text{ integer}, \ b \text{ string}, \ c \text{ string});
\]

The form functions are slightly different and require the using clause as shown below. Form functions generally return a table in response to a set of inputs and thus require instructions for the use of wrappers stored in some ontology. The using clause shown below is used to name the wrapper \( w \) to be used to extract the table containing the output fields from the response returned by the function. The list of attribute names in the extract clause must be a subset of the attribute list declared in wrapper \( w \) for URL \( u \). The output of a form function always is in XML format, and will include a set of rows for each set of input values. Hence, it is possible for the wrapper to extract attributes from a web document in XML format with arbitrary nesting.

\[
\text{define function} \quad \text{price} \\
\text{extract} \quad x \text{ integer}, \ y \text{ integer}, \ z \text{ float} \\
\text{using} \ w \text{ in ontology} \ o \\
\text{from URL} \ u \\
\text{submit} \ (a \text{ integer}, \ b \text{ string}, \ c \text{ string});
\]

While the syntax of the two functions appear similar, their use is distinguished in the following way. The form functions can be invoked as part of a create view statement or independently using the call statement as follows.

\[
\text{call} \quad \text{price with} \ (a, \ b, \ c);
\]

In the call statements above, the list of arguments in the with clause must match in type with the arguments in the submit clause of the define function statement in the ordered list. However, if a different order is preferred, the parameters of the function can be listed as shown to match the arguments in with clause. In this example, \( c \) will be passed to \( p \), \( a \) to \( q \) and finally \( b \) to \( r \) in the submit clause of define function statement.

\[
\text{call} \quad \text{price} \ (q, \ r, \ p) \text{ with} \ (a, \ b, \ c);
\]
In the other form of call statements below, price will be invoked for each row or \( r \) in the select statement that satisfies condition \( c \). Here too, the attributes in the select clause must match in type with the arguments in the submit clause of the define function statement in the order list.

\[
\text{call price} \ (p, \ q, \ r) \ \text{with select} \ a, \ b, \ c \ \text{from} \ r \ \text{where} \ c;
\]

### 3.2 Describing and Computing Capabilities of Sources and Mediators

Internet data sources have a wide variety of query-processing limitations. Some of these are [77]:

- **Condition-Attribute Restrictions**: Disallowing condition specification on certain attributes; requiring that a particular field be filled in.
- **Condition-Expression-Size Restrictions**: Limiting the number of conditions in the condition expression.
- **Condition-Expression-Structure Restrictions**: Allowing only atomic condition expressions; allowing only conjunctive queries; restricting expressions based on the structure of a form.

Our source description language is based on context-free grammar (CFG), standard parsing technology can be used to check for the supportability of a source query very efficiently. The following example illustrates the description.

**EXAMPLE 1** Suppose we have a source \( R(\text{make}, \ \text{model}, \ \text{year}, \ \text{color}, \ \text{price}) \) that provides information about cars for sale. The query capabilities of \( R \) may be described in our language as follows:

1. \( s \rightarrow s_1 | s_2 \)
2. \( s_1 \rightarrow \text{make} = $m \land \text{price} < $p \)
3. \( s_2 \rightarrow \text{make} = $m \land \text{color} = $c \)
4. \( \text{attributes} :: s_1 : \text{make, model, year, color} \)
5. \( \text{attributes} :: s_2 : \text{make, model, year} \)

In the above source description, symbols starting with "," are nonterminals. We use \$p\ for integer constants; \$c\ and \$m\ for string constants. The first three rules are standard CFG rules that describe the condition expressions \( R \) can evaluate. For instance, Rule (2) states that \( R \) can evaluate conditions like \((\text{make} = "BMW" \land \text{price} < 40000)\) and Rule (3) states that \( R \) can evaluate conditions like \((\text{make} = "BMW" \land \text{color} = "red")\). The last two rules indicate the attributes that can be exported by \( R \).

In the following, we present more example of our framework for describing the query capabilities of Internet data sources and mediators. In our framework, each data source exports a set of relational views. Conceptually, a query to a source is submitted by filling out a form on one of the views exported by the source. The query specifies values for some attributes of the view, and the
result of executing the query is a set of tuples of the source view on which the query is posed. The following example illustrates a source view and answering queries on that view.

**EXAMPLE 2** Consider a source that exports a view $R(X, Y, Z)$. Let the set of tuples in source view $R$ be $\{(x_1, y_1, z_1), (x_1, y_2, z_1), (x_2, y_2, z_2)\}$. Query $R(X, Y, z_1)$ results in $\{(x_1, y_1, z_1), (x_1, y_2, z_1)\}$, while query $R(X, y_1, Z)$ returns one tuple: $(x_1, y_1, z_1)$.

A mediator integrates data from multiple sources and exports a set of integrated views. Each integrated view is defined in terms of source views and/or other integrated views. The set of operations used to define integrated views are *union*, *join*, *selection* and *projection*.

**EXAMPLE 3** Consider a mediator with five data sources. Let the respective source views be $R_1(X, Y, Z)$, $R_2(X, Y, Z)$, $R_3(X, Y, Z)$, $R_4(Z, U)$ and $R_5(U, V, W)$. Let the mediator define the following two views: $M_1(X, Y, Z)$ is the union of $R_1$, $R_2$ and $R_3$; $M_2(X, Y, Z, U, V, W)$ is the join of $M_1(X, Y, Z)$, $R_4$ and $R_5$. Users can pose queries on the views of a mediator in a manner similar to submitting queries on source views. For instance, one can specify $M_2(x_1, Y, Z, U, V, w_1)$ as a query to the mediator.
Chapter 4

Progress Report: Column Name Identification

4.1 Labeling

Labeling is mostly interesting at the result interface schema (RS). The main task for a wrapping tool on result level is to identify dynamic content (values) and to assign a suitable label to all content items. The latter use-case is called Labeling. Labeling use case is based on the identification of a concept-instance relationship. In case of labeling, the query to knowledge source is value based. The distinguishing between static structure and dynamic content is done by wrapping tools such as RoadRunner, PickUp, FastWrap. The assignment of suitable values is an open question, which we will solve by using a series of steps including web knowledge. Two variants of the labeling use-case need to be distinguished:

**Label can be extracted from Page** Both sets, the label set $L$ as well as the value-set $V$ are known - the task of the Labeling component is to identify correspondence between $l_i \in L$ and $v_j \in V$. A knowledge source can be applied in this scenario as ranker among different relationships. Thus a web knowledge source is queried with each possible pair $(l_i, v_j)$ in order to get a measure for the probability of this semantic relationship. Best knowledge sources for this use cases are search engines or web encyclopedia due to their huge amount of information content including instances. A probability measure can be derived approximately from the number of results the knowledge source delivers. We have done some experiment with Web search engines such as Google, Yahoo and MSN. The details are discussed in the section Speculative Labeling.

**Label is NOT available on the Page** In this case only values $v_1, \ldots, v_n \in V$ can be extracted from page. Labels are unknown at first. The knowledge source has to identify matching labels $l_1, \ldots, l_m \in L$ such that each found value can be assigned to one of these labels. Thus two queries to the knowledge source are needed: First a query, which identifies a suitable label for each of the values given on page, second a pruning to validate label-value pairs by using another web knowledge source. The best $k$ label candidates are then presented to the user to choose the label that fits.
4.1.1 Language Patterns

Although it is generally assumed that improvements in language processing will be made through the integration of linguistic information and statistical techniques, the reality is that language is very diverse and looking for specific patterns of words that repeat enough to be statistically significant tends not to be a very fruitful task: sequences longer than three words are not generally repeated often enough to be statistically significant. At the same time, the identification of named entities: names, dates, places, organizations etc., has proved to be a very useful preliminary task in many natural language processing systems. The rationale for using Language patterns is that certain English text fragments indicate semantic relationships between terms. First Hearst has stated out such indicating patterns. Cimiano and Staab [21] have used these patterns in their system PANKOW (Pattern-based Annotation through Knowledge on the Web) in context with Google in order to annotate web pages. The PANKOW system uses the number of pages returned by Google in order to classify a specific concept from a given ontology. They have extended the work in C-PANKOW. Useful Patterns reused from Hearst in our context are:

- **H1** < L > such as < V|L' >
- **H2** such < L > as < V|L' >
- **H3** < L >, (especially | including) < V|L' >
- **H4** < V|L' > (and | or) other < L >
- **D1** the < V|L >
- **D2** the < L|V >
- **A** < V|L' >, a < L >
- **C** < V|L' > is a < L >

The patterns denoted H1 . . . H4 refer to Hearst, D1 and D2 are "Definite-Patterns", they employ definition sentences, actually refer to some entity previously mentioned in the text, A-Pattern refers to an apposition and C retrieves copula-structures in text, which is an intransitive verb which links a subject to an object, an adjective or a constituent denoting a property of the subject. All of the above mentioned patterns are able to recognize is-a relationship between a concept label L and instance value V or synonym, subconcept, is-close-to relationship between concepts. The above patterns would match the following expressions:

- continents such as Asia (H1)
- vehicles such as cars (H1)
- such continents as Africa (H2)
- such cars as Camry (H2)
- presidents, especially Bill Clinton (H3)
- vehicles, especially motor-bikes (H3)
- the Sears Tower and other sites in Chicago (H4)
- motor-bikes and other two-wheeled vehicles (H4)
- the Hilton hotel (D1)
- the hotel Hilton (D2)
- Omni, a hotel in the center of Los Angeles (A)
- The Omni is a nice hotel in the center of Los Angeles (C)
4.1.2 Labeling Algorithm

The whole process of labeling is divided into two main concepts: (i) Label may be present in the web page; (ii) Label may NOT be present in the web page. When label is available in the web page, we utilize those. When label is NOT available, we employ a number of sub processes such as Language pattern queries to search engines, consult web knowledge sources such as WikiPedia, make use of domain extraction ontologies, make use of user query variable etc. The step by step procedure to solve the labeling problem in our web data integration context are the following:

- Step1: Extract Concept Information from Input Web page (e.g from Web Form and Meta Tag) as well as from user SQL query variable
- Step2: Extract Concept Information from Output Result page (e.g in \(<TH>\), \(<THEAD>\) tags, use heuristics for generic data types i.e email, date, time, URL, gene, protein, phone etc)
- Step3: Formulate and execute speculative queries to web search engines (e.g Google, Yahoo and MSN)
- Step4: Prune and Rank Results (i.e compute Statistical Fingerprints)

**INPUT**

\[L(1\ldots m) : A set of labels\]
\[A(1\ldots n) : A set of anonymous attribute\]
Precondition \(m > n\)
\[P(1\ldots k) : A set of Pattern\]
\[NP(1\ldots l) : A set of generic numeric pattern (e.g$\$, year, dd – mm – yyyy)\]
\[V(1\ldots t, 1\ldots n) : Table Data Value set\]

**VARIABLE**

\[N : Number of Google hits\]
\[H(1\ldots n, 1\ldots m) : 2D array to store hit count\]
\(R \rightarrow 1\ldots t Random indexes between 1\ldots t, generate one time\)
Formulate Speculative query : \(L \times P \times V \rightarrow N\)

Our first approach utilizes page meta information and direct information given in a web form tags. Example: The following cut out shows meta information given on the site http://www.amazon.com

```html
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<meta name="description" content="Online shopping from the earth’s biggest selection of books, magazines, music, DVDs,"
```
Algorithm 1 LADS: Labeling Anonymous Data Set

for $i = 1$ to $n$
    $H[i][1..m] = 0$
    for each $r \in R$
        $instanceValue = V_r$
        for $j = 1$ to $m$
            $labelValue = L[j]$
            for $p = 1$ to $k$
                $QueryVar = labelValue + P[p] + instanceValue$
                $N \leftarrow \text{GoogleExecuteQuery}(QueryVar)$
                $H[i][j] += N$
            end for
        end for
    end for
end for

for $i = 1$ to $n$
    Let $z$ be index$(1..m)$ of highest count in $H[i][1..m]$
    $A[i] = L[z]$
end for

Example: The following screen shot shows the advanced book search option of amazon.com. The set of concepts that can be extracted from the form tag labels are 
\{Keywords, Author, Title, ISBN, Publisher, Subject\} Again, we wish to capture concepts from user query variable. An example is as follows:

\begin{verbatim}
SELECT Title, Rating
FROM amazon.com
WHERE Title like "Discrete Mathematics"
AND Rating=5;
\end{verbatim}
By analyzing the advanced search option of amazon.com site, we see that the 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.com 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 may only be interested to see records correspond to Rating 5. The above phenomena leads to what we so called local interface scheme inadequacy problem. Formally, we denote a local search interface schema (LIS) as $S_i = \{A_1, A_2, \ldots, A_k\}$, where each $A_j$ is an attribute. When a query is submitted against the search interface, the entities in the returned results also have a certain ”hidden” schema, denoted
as $S_e = \{a_1, a_2, \ldots, a_n\}$, where each $a_j (j = 1, \ldots, n)$ is an attribute to be discovered. Note that this "hidden schema" is NOT the same as Hidden attributes that are invisible on a query interface that have fixed values assigned to them in each query [65]. The schema of the retrieved data and the interface schema usually share a significant number of attributes. Therefore, if an attribute $a_i$ in the search results does have a matched attribute $A_t$ in the LIS, all the data units identified with $a_i$ can be labeled by the name of $A_t$. However it is often the case as in our SQL user query example above that $S_e$ is not entirely contained in $S_i$. In that case there will be no label in the search interface that can be assigned to the discovered data units of this attribute, then the only hints to label the attribute is to extract concept from the user SQL query variable. The set of concepts that can be extracted from user SQL query variable for the above example is $\{\text{Title}, \text{Rating}\}$.

Our final set of candidate labels $L$ are the union of all three sources: meta tag info, $M$, form tag label info, $F$ and user query variable, $Q$. Thus $L \subseteq (M \cup F \cup Q)$. The following figure depicts the overall scenario of the problem we are going to model and address.

![Diagram](image)

Figure 4.2: Labeling Process Steps

### 4.1.3 Probabilistic Labeling of Anonymous Data

An anonymous dataset is a structured collection of data in which descriptive labels for similar objects are missing, e.g. Table 4.1 shows a relational anonymous dataset about music. This dataset is a prototypical example of the output produced by most of the state-of-the-art wrappers for web data extraction. Note that the data has a well defined although semantically poor schema $R(A_1, A_2)$, also both the attributes have well defined domains ($A_1$ contains artist names and $A_2$ contains album names). The labeling problem, which we deal in this research is that of finding descriptive labels for an anonymous relational dataset. Labeling problem can be solved in two steps: Finding a good set of candidate labels, and finding the best matching between labels and the attributes. Different ways of finding candidate labels are possible. For example, the labels may be mined from the web pages with the data (e.g from web form label tag, meta tag etc) or the user
Table 4.1: An anonymous dataset about music, containing a single relation $R(A_1, A_2)$

<table>
<thead>
<tr>
<th>$R$</th>
<th>$A_1$</th>
<th>$A_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Miles Davis</td>
<td>Kind of Blue</td>
</tr>
<tr>
<td></td>
<td>John Coltrane</td>
<td>A Love Supreme</td>
</tr>
<tr>
<td></td>
<td>John Coltrane</td>
<td>My Favourite Things</td>
</tr>
</tbody>
</table>

may provide her own label (e.g. in the form of SQL query variable, map the query to site’s IS and RS). We propose fully automated and independent methods for the label selection and the label assignment tasks in this research.

4.1.4 Candidate Label Selection

Our label selection algorithm is based on the following assumptions. Given attribute $A_i$ of anonymous relations $R$, we assume that: (1) plausible labels for $A_i$ are likely to occur in Web documents that contain instances of $A_i$; (2) such labels are likely to appear close to those instances in such documents; (3) this close co-occurrence (pairwise mutual information, PMI) expresses the hyponymy relationship between the label and the instance; (4) this situation is frequent on the web, so that these pages are likely to be crawled by popular search engines.

The process consists of searching for documents with specific patterns, usually called Hearst patterns, expressing relationships among terms, particularly hyponymy. For instance, the pattern "$NP_1$ such as $(NP_2)^*$" can be used to find one or more instances $NP_2$ (for noun phrase) of a class $NP_1$. We formulate a query to a search engine using a Hearst pattern and search for nouns that appear close to the exact query phrase in the documents returned by the search engine. We note that the correct labeling heavily depends on the correct "sense of a word". An illustrative example is as follows: suppose we want to label a value "Sin". If we are working in genome domain, it can be labeled as gene, while working in trigonometric domain, it is a measure of an angle, in natural language, it refers to some misdeed and lastly as in Immigration system in Canada, it refers to Social Insurance Number. This example illustrates that we can solve labeling problem considering one domain at a time. Handling multiple domains at a time will lead to spurious result. This problem is known as Word Sense Disambiguation (WSD) problem, it is a separate fundamental research problem, we don’t deal it here in our research. We assume that each value will have one type. Our label would be pair wise disjoint, type/ concept wise. Example: Address is a concept. In the label field it should have one instance as addr. If multiple label instances are there, the algorithm will NOT be able to distinguish between those. Here are some example of conflicting labels:

- DOB, Date of Birth
- Present Address, Permanent Address
- Student, Pupil
- President, Vice President
- Artist, singer, performer
4.1.5 Affinity-Based Labeling

Let $R(A_1, A_2, \ldots, A_n)$ be a relation on $n$ anonymous attributes $A_1, \ldots, A_n$, where each $A_j$ is of domain $D_j$ and domains are pairwise disjoint. Assume we are given an instance of $R$ with $t$ tuples, and a set $L = \{l_1, \ldots, l_m\}$ with $m$ candidate labels ($m > n$). Our goal is to assign to each $A_j$ a label $l_i \in L$ which is the best descriptor for attribute $A_j$.

There are two challenges in finding good labels for anonymous data. First, we need a way of measuring how well a labeling $R \rightarrow L^n$ describes the domains of the attributes in $R$. Second, the cost of the labeling algorithm must not be too high. We note here that the labeling problem can be rephrased as finding a maximum-weight matching in a complete bipartite graph $G(V, E)$ where the vertex sets are the columns in $R$ and $L$, respectively, and the weight of each edge $(A_j, l_i)$ indicates how well $l_i$ describes $A_j$. The fastest algorithms for this problem run in low polynomial, but super-linear time on the size of the graph. In order to minimize the number of accesses to the search engine, we use a greedy labeling strategy: we find a label for each attribute in isolation, and once a label is assigned for an anonymous attribute, it is no longer considered as a candidate label for other attributes. Also, we take a probabilistic approach for estimating the goodness of a labeling. More precisely, we use $P(l_i | A_j)$, the probability of $l_i$ describing well attribute $A_j$ as the metric for evaluating a labeling of an anonymous dataset. By doing so, we account for the inherent uncertainty in how well a label describes a domain. For simplicity, we assume that the probability of a label being a good fit for an attribute is independent of other attributes.

**Computing Label-Attribute Affinities.** According to Baysian probability $P(l_i | A_j) = \frac{P(A_j, l_i) P(l_i)}{P(A_j)}$ Thus, we need to estimate $P(A_j | l_i)$ and $P(l_i)$ in order to compute the affinity between $l_i$ and $A_j$. $P(A_j)$ is a normalizing factor and can be ignored for all practical purposes. Intuitively, the prior probability $P(l_i)$ captures the users’ preference for the label $l_i$ regardless of its affinity with any of the attributes. We estimate the true affinity between labels and domains by submitting speculative queries to popular Web search engines. A speculative query is a statement saying that a given label $l_i$ is a good descriptor for attribute $A_j$. We use the number of documents that the search engines classify as relevant answers for that query to estimate the probabilities above. The intuition behind speculative queries is as follows. If label $l_i$ is a better match for attribute $A_j$ than label $l_k$, a Web document $D$ containing high quality information about an instance of $A_j$ is more likely to refer to $l_i$ than to $l_k$. The following two definitions will be helpful to make understand the use of speculative queries to find Label-Attribute Affinity.

**Definition 1.** The Document Count of a query expression $e$, denoted $DC(e)$, is the number of documents relevant to $e$ according to a given Web search engine.

**Definition 2.** Given an anonymous relation $R(A_1, \ldots, A_n)$ and a set of candidate labels $L = \{l_1, \ldots, l_m\}$, the Label-Attribute Affinity between $A_j$ and $l_i$, denoted $LAA(A_j, l_i)$, is defined as $LAA(A_j, l_i) = P(A_j | l_i) = \frac{1}{||A_j||} \sum_{x=1}^{||A_j||} \frac{DC(l_i \wedge v_x)}{\sum_{y=1}^{||L||} DC(l_y \wedge v_x)}$

where $[A_j]$ is the active domain of $A_j$, and $v_x \in [A_j]$. The active domain of an attribute is the set of distinct values for the attribute that are used in the actual database instance. We write $l_i \wedge v_x$ to denote the speculative query asserting that $l_i$ and value $v_x \in A_j$ are likely to appear together in a Web document.

\begin{itemize}
  \item Gender, Sex
\end{itemize}
Table 4.2: No. of answers to different kinds of speculative queries among different Search Engine

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>artist</td>
<td>Miles Davis</td>
<td>53700</td>
<td>78900</td>
<td>27500</td>
</tr>
<tr>
<td>title</td>
<td>Miles Davis</td>
<td>1930</td>
<td>3550</td>
<td>704</td>
</tr>
<tr>
<td>album</td>
<td>Miles Davis</td>
<td>15800</td>
<td>15600</td>
<td>4560</td>
</tr>
<tr>
<td>artist</td>
<td>John Coltrane</td>
<td>28700</td>
<td>44200</td>
<td>10100</td>
</tr>
<tr>
<td>title</td>
<td>John Coltrane</td>
<td>3400</td>
<td>1150</td>
<td>1410</td>
</tr>
<tr>
<td>album</td>
<td>John Coltrane</td>
<td>553</td>
<td>6180</td>
<td>1220</td>
</tr>
<tr>
<td>artist</td>
<td>Kind of Blue</td>
<td>102</td>
<td>104</td>
<td>73</td>
</tr>
<tr>
<td>title</td>
<td>Kind of Blue</td>
<td>715</td>
<td>697</td>
<td>323</td>
</tr>
<tr>
<td>album</td>
<td>Kind of Blue</td>
<td>15400</td>
<td>36000</td>
<td>7660</td>
</tr>
<tr>
<td>artist</td>
<td>A Love Supreme</td>
<td>7</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>title</td>
<td>A Love Supreme</td>
<td>814</td>
<td>465</td>
<td>274</td>
</tr>
<tr>
<td>album</td>
<td>A Love Supreme</td>
<td>10500</td>
<td>1150</td>
<td>3160</td>
</tr>
</tbody>
</table>

Table 4.3: LAA based on Different Search Engine result

<table>
<thead>
<tr>
<th></th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(A₁</td>
<td>artist)</td>
<td>0.8153</td>
<td>0.8311</td>
</tr>
<tr>
<td>P(A₁</td>
<td>title)</td>
<td>0.0655</td>
<td>0.029</td>
</tr>
<tr>
<td>P(A₁</td>
<td>album)</td>
<td>0.0194</td>
<td>0.1395</td>
</tr>
<tr>
<td>P(A₂</td>
<td>artist)</td>
<td>0.003</td>
<td>0.001</td>
</tr>
<tr>
<td>P(A₂</td>
<td>title)</td>
<td>0.0579</td>
<td>0.1534</td>
</tr>
<tr>
<td>P(A₂</td>
<td>album)</td>
<td>0.9358</td>
<td>0.8451</td>
</tr>
</tbody>
</table>

4.1.6 Speculative Labeling

This section describes our implementation of the labeling method and our strategy to formulate speculative queries. We illustrate the discussion using the dataset in Table 4.1 and candidate labels \( L = \{\text{artist, title, album}\} \)

Formulating Speculative Queries

Speculative query \( l_i \land v_x \) formulates the hypothesis that \( l_i \) is a class of objects of which \( v_x \) is a member. The first question that arises is how to formulate such a hypothesis using the keyword-based search paradigm currently in use in all major popular Web search engines. Table 4.2 compares the number of results of speculative queries using phrase approach, which uses speculative queries asking for documents containing a phrase formed by the label and the value (e.g. “artist John Coltrane”). All the queries were run on August 3, 2008. Table 4.3 shows the affinity between the labels and attributes, computed as in Definition 2, using the DC values in Table 4.2. We run more queries on movie domain anonymous dataset as shown in the following table 4.4: The following table data 4.5, 4.6, 4.7 was collected as of August 19, 2008, showing the number of hits that we got from different web search engine: The data in table 4.5 is used to label anonymous attribute \( A_5 \) of the table 4.4. There were 3 competing label for \( A_5 \), namely: \( \text{actor, director and directed} \)
Table 4.4: Sample Anonymous Dataset from Movie domain

<table>
<thead>
<tr>
<th></th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romance</td>
<td>5 Stars</td>
<td>1965</td>
<td>Doctor Zhivago</td>
<td>David Lean</td>
</tr>
<tr>
<td>Comedy</td>
<td>5 Stars</td>
<td>1936</td>
<td>Modern Times</td>
<td>Charles Chaplin</td>
</tr>
<tr>
<td>Action</td>
<td>4.5 Stars</td>
<td>1981</td>
<td>Raiders of the Lost Ark</td>
<td>Steven Spielberg</td>
</tr>
<tr>
<td>Epic</td>
<td>4 Stars</td>
<td>1960</td>
<td>Spartacus</td>
<td>Stanley Kubrick</td>
</tr>
</tbody>
</table>

Table 4.5: Hits count to annotate A5 Label

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor</td>
<td>David Lean</td>
<td>278</td>
<td>82</td>
<td>79</td>
</tr>
<tr>
<td>director</td>
<td>David Lean</td>
<td>61000</td>
<td>160000</td>
<td>43600</td>
</tr>
<tr>
<td>directed by</td>
<td>David Lean</td>
<td>39400</td>
<td>132000</td>
<td>23600</td>
</tr>
<tr>
<td>actor</td>
<td>Charles Chaplin</td>
<td>3520</td>
<td>2500</td>
<td>706</td>
</tr>
<tr>
<td>director</td>
<td>Charles Chaplin</td>
<td>15600</td>
<td>26100</td>
<td>9280</td>
</tr>
<tr>
<td>directed by</td>
<td>Charles Chaplin</td>
<td>11000</td>
<td>25800</td>
<td>9570</td>
</tr>
<tr>
<td>actor</td>
<td>Steven Spielberg</td>
<td>1940</td>
<td>860</td>
<td>3120</td>
</tr>
<tr>
<td>director</td>
<td>Steven Spielberg</td>
<td>368000</td>
<td>1270000</td>
<td>295000</td>
</tr>
<tr>
<td>directed by</td>
<td>Steven Spielberg</td>
<td>293000</td>
<td>799000</td>
<td>129000</td>
</tr>
<tr>
<td>actor</td>
<td>Stanley Kubrick</td>
<td>359</td>
<td>312</td>
<td>245</td>
</tr>
<tr>
<td>director</td>
<td>Stanley Kubrick</td>
<td>138000</td>
<td>373000</td>
<td>89700</td>
</tr>
<tr>
<td>directed by</td>
<td>Stanley Kubrick</td>
<td>61500</td>
<td>258000</td>
<td>51600</td>
</tr>
</tbody>
</table>

by. From the hits count, director dominates all other. Therefore we assign director to be the label for A5. The data in table 4.6 is used to label anonymous attribute A4 of the table 4.4. There were 3 competing label for A4, namely: film, title and movie. From the hits count, title loses the competition. Therefore we assign film/movie to be the label for A4.

Anonymous attribute A3 conforms to our generic, conventional data type year, so we assign year to be the label for A3

Anonymous attribute A2 falls in the category of repeating suffix (here in this case stars). Therefore we assign stars to be the label for A2. It can also be shown to be labeled as rating, we deliberately exclude the computation.

Last of all, our remaining anonymous attribute is A1. For this we use data from table 4.7 as follows: It can easily be seen that genre outperform other hits count. So we assign genre to be the label for A1. This completes our labeling anonymous dataset in table 4.4, all the attributes are semantically enriched with meaningful label.

From Table 4.3, we see that there is a ordered rank list of probabilities associated with each label to an anonymous attribute. Therefore we need to prune the result by what we so called Statistical Fingerprints. The mathematical and logical foundation of which is discussed in the following section.

37
Table 4.6: Hits count to annotate $A_4$ Label

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>film</td>
<td>Doctor Zhivago</td>
<td>4910</td>
<td>5040</td>
<td>2720</td>
</tr>
<tr>
<td>title</td>
<td>Doctor Zhivago</td>
<td>1160</td>
<td>1810</td>
<td>495</td>
</tr>
<tr>
<td>movie</td>
<td>Doctor Zhivago</td>
<td>4510</td>
<td>6300</td>
<td>1420</td>
</tr>
<tr>
<td>film</td>
<td>Modern Times</td>
<td>13500</td>
<td>34200</td>
<td>8030</td>
</tr>
<tr>
<td>title</td>
<td>Modern Times</td>
<td>2090</td>
<td>2350</td>
<td>986</td>
</tr>
<tr>
<td>movie</td>
<td>Modern Times</td>
<td>4800</td>
<td>10800</td>
<td>1850</td>
</tr>
<tr>
<td>film</td>
<td>Raiders of the Lost Ark</td>
<td>17900</td>
<td>44900</td>
<td>12300</td>
</tr>
<tr>
<td>title</td>
<td>Raiders of the Lost Ark</td>
<td>939</td>
<td>3550</td>
<td>706</td>
</tr>
<tr>
<td>movie</td>
<td>Raiders of the Lost Ark</td>
<td>26200</td>
<td>80300</td>
<td>17800</td>
</tr>
<tr>
<td>film</td>
<td>Spartacus</td>
<td>10800</td>
<td>28100</td>
<td>6210</td>
</tr>
<tr>
<td>title</td>
<td>Spartacus</td>
<td>1280</td>
<td>2490</td>
<td>798</td>
</tr>
<tr>
<td>movie</td>
<td>Spartacus</td>
<td>6810</td>
<td>19200</td>
<td>5570</td>
</tr>
</tbody>
</table>

Table 4.7: Hits count to annotate $A_1$ Label

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>genre</td>
<td>Romance</td>
<td>537000</td>
<td>174000</td>
<td>295000</td>
</tr>
<tr>
<td>director</td>
<td>Romance</td>
<td>736</td>
<td>1610</td>
<td>2080</td>
</tr>
<tr>
<td>film</td>
<td>Romance</td>
<td>65100</td>
<td>146000</td>
<td>30600</td>
</tr>
<tr>
<td>genre</td>
<td>Comedy</td>
<td>755000</td>
<td>5990000</td>
<td>3570000</td>
</tr>
<tr>
<td>director</td>
<td>Comedy</td>
<td>7400</td>
<td>28800</td>
<td>23700</td>
</tr>
<tr>
<td>film</td>
<td>Comedy</td>
<td>711000</td>
<td>2040000</td>
<td>407000</td>
</tr>
<tr>
<td>genre</td>
<td>Action</td>
<td>4980000</td>
<td>16000000</td>
<td>6500000</td>
</tr>
<tr>
<td>director</td>
<td>Action</td>
<td>82000</td>
<td>219000</td>
<td>47900</td>
</tr>
<tr>
<td>film</td>
<td>Action</td>
<td>1270000</td>
<td>2620000</td>
<td>361000</td>
</tr>
<tr>
<td>genre</td>
<td>Epic</td>
<td>43900</td>
<td>52400</td>
<td>16500</td>
</tr>
<tr>
<td>director</td>
<td>Epic</td>
<td>4910</td>
<td>17100</td>
<td>6700</td>
</tr>
<tr>
<td>film</td>
<td>Epic</td>
<td>87900</td>
<td>145000</td>
<td>26700</td>
</tr>
</tbody>
</table>
Statistical Fingerprints

In the previous section, we employed Hearst pattern to formulate speculative queries to Web search engines. In our approach, rather than actually downloading web pages for further processing, we just take the number of web pages in which a certain pattern appears as an indicator for the strength of the pattern. Given a candidate entity we want to classify with regard to an existing ontology, we instantiate the above patterns with each concept from the given ontology. For each pattern instance, we queried Google, Yahoo and MSN to get the number of documents that contain it. The function ‘count’ models this query.

\[
\text{count} : E \times C \times P \rightarrow N
\]

Thereby, \(E\), \(C\) and \(P\) stand for the set of all entities to be classified, for the concepts from a given ontology and for a set of pattern schema, respectively. Thus, \(\text{count}(e, c, p)\) returns the number of hits of pattern of the pattern schema \(p\) instantiated with the entity \(e\) and the concept \(c\). Further we define the sum over all the patterns conveying a certain relation \(r\):

\[
\text{count}_r(e, c) = \sum_{p \in P_r} \text{count}(e, c, p)
\]

where \(P_r\) is the set of pattern schemas denoting a certain relation \(r\). Now we formally define the statistical fingerprint of an entity \(e\) with respect to a relation \(r\) and a set of concepts \(C\):

\[
SF(e, r, C) := \{(c, n) | c \in C \land n = \text{count}_r(e, c)\}
\]

Further, instead of considering the complete statistical fingerprints, we consider views of these such as defined by the following formula. The first formula defines a view of the statistical fingerprint which only contains the concept with maximal number of hits.

\[
SF_{\text{max}}(e, r, C) := \{(c, n) | c := \text{argmax}_{c' \in C} \text{count}_r(e, c') \land n = \text{count}_r(e, c)\}
\]

Further, we extend this to consider the top \(m\) concepts with maximal count:

\[
SF_m(e, r, C) := \{(c, n) | C = \{c_1, c_2, \ldots, c_{|C|}\} \land \text{count}_r(e, c_1) \leq \ldots \leq \text{count}_r(e, c_{|C|}) \land c \in \{c_1, \ldots, c_m\} \land n = \text{count}_r(e, c)\}
\]

(if \(m \leq |C|\)) Finally, we also consider a view only taking into account those concepts having hits over a certain threshold \(\theta\), a value of 100 seems reasonable as per the experimental result reported in the literature.

\[
SF_\theta(e, r, C) := \{(c, n) | \text{count}_r(e, c) \geq \theta \land n = \text{count}_r(e, c)\}
\]

We can now combine these views by set operations. For example, we yield the set of the \(m\) top concepts having hits over a threshold \(\theta\) as follows:

\[
SF_{m, \theta}(e, r, C) = SF_m(e, r, C) \cap SF_\theta(e, r, C)
\]

We report here some typical Statistical Fingerprint (SF) computed for some attribute from Watch domain anonymous dataset as in table 4.8. The queries were run on August 21, 2008.

The corresponding Statistical Fingerprints (SF) are shown in fig 4.3 and 4.4
Table 4.8: Sample Anonymous Dataset from Watch domain

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Armani</td>
<td>AR5447</td>
<td>Ladies</td>
<td>Stainless steel bracelet</td>
<td>$195.00</td>
<td></td>
</tr>
<tr>
<td>Seiko</td>
<td>SDWG32</td>
<td>Men</td>
<td>Stainless steel Butterfly clasp</td>
<td>$400.00</td>
<td></td>
</tr>
<tr>
<td>Longines</td>
<td>L51580966</td>
<td>Petite</td>
<td>Stainless steel bracelet</td>
<td>$1,700.00</td>
<td></td>
</tr>
<tr>
<td>Casio</td>
<td>DW5600E1V</td>
<td>Men</td>
<td>Plastic, black</td>
<td>$70.00</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.3: Statistical Fingerprints for Armani and Longines

From the table data and statistical fingerprint, it is clear that Armani and Longines are instances of *brand*. Confusion is arising when to annotate a less popular attribute value such as *L51580966*. *L51580966* is a member of the anonymous attribute *A2* to which *AR5447* also a member. We could not obtain any hits for *AR5447* from any of the web search engine using phrase approach. Therefore we resort to word approach to web search engine (e.g. "band", "L51580966") for this less popular attribute value. We get the following results, shown in table 4.10: As can be seen from the statistical fingerprints, there are as many as 4 potential candidate to be the label for *L51580966*. *So only the best 1st and 2nd count is not enough.*
<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>band</td>
<td>Armani 520</td>
<td>391</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>brand</td>
<td>Armani 32200</td>
<td>490000</td>
<td>32000</td>
<td></td>
</tr>
<tr>
<td>category</td>
<td>Armani 1140</td>
<td>24800</td>
<td>15700</td>
<td></td>
</tr>
<tr>
<td>condition</td>
<td>Armani 221</td>
<td>181</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>Armani 2510</td>
<td>1650</td>
<td>720</td>
<td></td>
</tr>
<tr>
<td>display</td>
<td>Armani 560</td>
<td>170</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>gender</td>
<td>Armani 22</td>
<td>33</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>features</td>
<td>Armani 642</td>
<td>642</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>material</td>
<td>Armani 208</td>
<td>245</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>movement</td>
<td>Armani 769</td>
<td>753</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>Armani 7920</td>
<td>27200</td>
<td>1710</td>
<td></td>
</tr>
<tr>
<td>price</td>
<td>Armani 8340</td>
<td>17500</td>
<td>2170</td>
<td></td>
</tr>
<tr>
<td>price range</td>
<td>Armani 7</td>
<td>11</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>savings amount</td>
<td>Armani 0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>Armani 559</td>
<td>664</td>
<td>414</td>
<td></td>
</tr>
<tr>
<td>style</td>
<td>Armani 7820</td>
<td>34900</td>
<td>3050</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>Armani 3720</td>
<td>8250</td>
<td>478</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Armani 1450</td>
<td>1850</td>
<td>363</td>
<td></td>
</tr>
<tr>
<td>band</td>
<td>Longines 1090</td>
<td>937</td>
<td>455</td>
<td></td>
</tr>
<tr>
<td>brand</td>
<td>Longines 15200</td>
<td>77900</td>
<td>5620</td>
<td></td>
</tr>
<tr>
<td>category</td>
<td>Longines 220</td>
<td>258</td>
<td>132</td>
<td></td>
</tr>
<tr>
<td>condition</td>
<td>Longines 116</td>
<td>94</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>Longines 500</td>
<td>458</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>display</td>
<td>Longines 78</td>
<td>53</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>gender</td>
<td>Longines 5</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>features</td>
<td>Longines 246</td>
<td>143</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>material</td>
<td>Longines 51</td>
<td>18</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>movement</td>
<td>Longines 866</td>
<td>1140</td>
<td>242</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>Longines 2400</td>
<td>1350</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>price</td>
<td>Longines 1700</td>
<td>1290</td>
<td>501</td>
<td></td>
</tr>
<tr>
<td>price range</td>
<td>Longines 1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>savings amount</td>
<td>Longines 0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>Longines 202</td>
<td>132</td>
<td>132</td>
<td></td>
</tr>
<tr>
<td>style</td>
<td>Longines 442</td>
<td>819</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>Longines 349</td>
<td>974</td>
<td>132</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Longines 55</td>
<td>59</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.4: Statistical Fingerprints for Longines and L51580966

Table 4.10: Hits count to annotate $A_2$ Label, using word approach

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
<th>Google</th>
<th>Yahoo</th>
<th>MSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>band</td>
<td>L51580966</td>
<td>163</td>
<td>289</td>
<td>12</td>
</tr>
<tr>
<td>brand</td>
<td>L51580966</td>
<td>179</td>
<td>430</td>
<td>48</td>
</tr>
<tr>
<td>category</td>
<td>L51580966</td>
<td>31</td>
<td>155</td>
<td>40</td>
</tr>
<tr>
<td>display</td>
<td>L51580966</td>
<td>224</td>
<td>242</td>
<td>29</td>
</tr>
<tr>
<td>gender</td>
<td>L51580966</td>
<td>69</td>
<td>107</td>
<td>10</td>
</tr>
<tr>
<td>features</td>
<td>L51580966</td>
<td>89</td>
<td>197</td>
<td>32</td>
</tr>
<tr>
<td>material</td>
<td>L51580966</td>
<td>116</td>
<td>171</td>
<td>32</td>
</tr>
<tr>
<td>movement</td>
<td>L51580966</td>
<td>351</td>
<td>515</td>
<td>36</td>
</tr>
<tr>
<td>model</td>
<td>L51580966</td>
<td>138</td>
<td>415</td>
<td>27</td>
</tr>
<tr>
<td>size</td>
<td>L51580966</td>
<td>241</td>
<td>474</td>
<td>23</td>
</tr>
<tr>
<td>style</td>
<td>L51580966</td>
<td>89</td>
<td>205</td>
<td>9</td>
</tr>
<tr>
<td>type</td>
<td>L51580966</td>
<td>43</td>
<td>109</td>
<td>10</td>
</tr>
</tbody>
</table>
Chapter 5

Proposed Work and Possible Future Work

The high level architecture of the system is shown in Figure 5.1. Here we implement the system as a layered architecture consist of four layers [57], as depicted in the following Figure.

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 and 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 column name identification algorithm *LADS*. It accesses the world-wide web through the Internet (Java URL and URLConnection class), fetches HTML documents and converts them into relational 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 will the proposed Wrapper 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. 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.

In this research proposal we have presented an implementation of a procedural and logical approach to web resource capability discovery/description. 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 [60, 63]. In this connection we have identified web form to be the gateway. We map user query variable to site, then in response to web form, we assume a tabular or XML output. By our Intelligent Wrapper, we convert the repeated structure into relational model [74], we can identify attributes and values in the table, then we can apply normal database SQL like queries on it.

The main contribution of this research is that we have proposed some novel algorithms: *LADS* for column name identification, algorithm to map user query to site using variant of Bipartite Graph and 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. We use MySQL as our underlying relational DB as it outperformed other DB [4] and the application module uses Java Database Connectivity (JDBC). To the best of our knowledge, such an integrated solution is not available anywhere, although some other systems can only extract/export region of interest web data. Ours is an integrated solution.

We wish to build a full Web Data Integration System from Biological domain consisting of a few
test sites such as:

- http://www.flybase.org
- http://atlas.med.harvard.edu/cgi-bin/alignace.pl

**Background.** David Embley et al pioneered the Web Data Extraction based on extraction ontologies. In order for their system to work, they also need some *seed* tuples, instances of the data. We only need ontology, a set of labels. We don’t need any seed instances of the data as we will formulate speculative query to web search engine. Our system is novel in the sense that we accommodate labels from the user query variable as well.

**Proposed work.** To complete my dissertation research, we will apply our techniques of Labeling anonymous dataset for Web Data Integration in scientific workflows. To achieve this, the following steps are required:

- Agree on a list of possible labels for a domain. We proposed to extract labels from web page as well as from user query variable. What we have noticed is that there may be a lot of redundant or misleading label information. In other words, how to prune a set of solid candidate labels is a huge research problem on its own.

- Agree on a set of exhaustive language pattern to complement Hearst pattern. We proposed a number of them.

- Limitations of our approach will be identified and insights to the requirements for a generic column name identification system will be reported.

**Possible future work.**

The possible future work includes:

- Label column name based on Google snippets only, which are short excerpts from the web page that show a bit of the context of the query term. An example is as follows: suppose we want to label *Camry*, which is a car made by Toyota. We just need to issue a query to Google "Camry is a". Extract the Google snippet to mine label for Camry.

- Label column name based on combination of Google hits and Dictionary Extraction.

- We have tested our hypothesis on 5-10 different domains. We wish to test these on broader aspects like 50-100 different domains.
Chapter 6

Conclusions

In order to build a large scale information integration system for the hidden web, the cost of manually locating information sources and manually obtaining source descriptions will not be acceptable. Therefore the following problem is worth studying: automatically obtaining descriptions about a given webbase’s content, such as schema or semantics of the contained data. Without good solution to the problem, the cost of discovering sources and obtaining source descriptions will become the "bottleneck" in building large-scale information integration systems. Speculative queries on popular domains (e.g books) return a much larger number of results compared to other less popular ones (e.g watches). This raises questions regarding the confidence in the labeling produced by our algorithm. We are interested in characterizing formally what would be an acceptable threshold for stopping the execution of speculative queries, also given that our algorithm samples from the anonymous dataset, we want to study how resilient it is to sampling bias. Search engine hits count depend on the popularity of the label as well as the value. What we have found is that the hits count is not deterministic, but it does NOT affect the result of our labeling algorithm. Query with or without quote produces different number of hits, but the decision of the algorithm never changes. Here in this preliminary thesis proposal, we have reported our experimental results from about 5-10 domains.

Again, we wish to introduce more pattern queries in addition to Hearst (e.g only phrase approach is not enough for less popular attribute value i.e AR5447, we need to use word approach as well), thereby produce more confidence in statistical fingerprints. As future work, we plan to include an ontology into the system [31], which can help the label assignment if the data domain of the web site is known. Our main contributions include:

- Developed an Algorithm LADS for Labeling Anonymous Data Set
- The proposed algorithm works fairly well in different domain
- Previous work as reported in the literature consider only the Hyponym/ Hypernym relationship (isa). We can extend the pattern to take care of the other ontological relationship such as synonym, part of etc. We are proposing the following lexico-syntactic pattern to complement Hearst pattern: Synonymous to; Part of; Belongs to; Also known as; A kind of. We propose to introduce these more generic pattern to complement Hearst Pattern.
Acknowledgements

I would like to thank my advisor Professor Hasan M. Jamil, Professors Farshad Fotouhi, Shiyong Lu, Lori Pile and Geoffrey S. Nathan for their guidance and support for this research.
Bibliography


