1-1-2011

Post Processing Wrapper Generated Tables For Labeling Anonymous Datasets

Emdad Ahmed
Wayne State University

POST PROCESSING WRAPPER GENERATED TABLES FOR LABELING ANONYMOUS DATASETS

by

EMDAD AHMED

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2011

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor **Date**

_____

_____

_____

_____

DEDICATION

To my parents and family with love. Specially to my late father, who dreamed my higher education abroad.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep and sincere thanks and gratitude to my advisor, Dr. Hasan M. Jamil. His passionate guidance, spacious research vision, tremendous support and extreme patience, his depth and breadth of knowledge have facilitated me to become a researcher. His unending encouragement, persistence and positive way of thinking have helped me to make the impossible possible. I enjoyed our numerous discussions and brainstorming sessions, and I am very thankful for his constant dedication and availability to me. His sincere, kind and sound advice has helped me to grow as a researcher, educator and individual.

Very warm thanks to Dr. Farshad Fotouhi, whose precious advice, experience and support have been always available to me during my five years spent in the department. His example of great leadership as the department chair has taught me many valuable qualities. I am extremely grateful to him for providing me the opportunity to teach and co-teach many undergraduate and graduate courses and for his kind accommodation of my teaching preferences.

Many, many thanks to Dr. Shiyong Lu for his constructive comments and suggestions on my research and his participation in our early efforts on using Taverna to experiment with Web access as Scientific Workflow, started as a course project in his database research seminar. I enjoyed his teaching of graduate courses in databases and learned a lot from his very successful teaching style. Thanks for being such a great teacher. I also would like to thank him for his advice on my job search and career start-up.

I am very thankful to Dr. Geoffrey S. Nathan to serve as external committee member of my dissertation committee.

I am very grateful to Dr. Hasan M. Jamil, Dr. Shiyong Lu, Dr. Farshad Fotouhi, and Dr. Geoffrey S. Nathan for serving on my dissertation committee and providing wonderful recommendation letters.

I am very thankful to Dr. Mustafa Atay, Dr. Hasan M. Jamil, Dr. Loren Schwiebert, Dr. Nathan Fisher, Dr. Farshad Fotouhi and Fernando Martincic who supervised my teaching assignments, and the following faculty from whom I have taken courses: Dr. Shiyong Lu, Dr. Hasan M. Jamil, Dr. Frank Stomp, Dr. Sorin Draghichi, Dr. Andrian Marcus, Dr. Farshad Fotouhi, Dr. John Liu and Dr. Eivan Avrutsky.

I am very grateful to Dr. Lori Pile and Marla Spain for giving me valuable feedback on my work on *Gene Regulatory Networks*. I was able to publish two IEEE papers out of it.

I would like to thank my fellow students for their encouragement and friendship: Dr. Shazzad Hosain, Anupam Bhattacharjee, Aminul Islam, Shafkat Amin, Kazi Zakia Sultana, Shahriyar Hossain, Saikat Dey.

I would like to thank other fellow students for their constant moral support for me: Salahuddin M. Aziz, Sharrukh Zaman, Yousof Gawasmeh, Saied Haiderian, Munirul Islam, Indranil Palit, Masud Ahmed, Farhana Dewan and the list continues to my memory.

I would also like to thank the following staff for the great service and support in these past few years: Deb Mazur, Judy Lechvar, Rachel Gillett, Alfred Glenn, Matt Orr, Gladys P. Maxwell, Jeff Fleming and Derrick White.

Conor Shaw-Draves of writing center of Wayne State University deserves special thanks and appreciation for the dissertation proofreading.

Finally, my *special thanks and appreciation* to my parents; my wife Kaniz Fatima; and my daughters Ardeena and Ariyana for their love, encouragement and constant supports throughout my life. Thanks for always being there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In this chapter first we present the notions of Web Wrappers and the post processing part of the wrapper, i.e., how to annotate or label the extracted tabular Web data and our strategy for marshalling Web resources into conceptual back end data. Next, our research motivation, challenges, goals and contributions are stated. Finally, an organization of the rest of the dissertation is outlined.

1.1 Web Wrapper

Data extraction from HTML pages is performed by software modules, usually called wrappers. A wrapper identifies and extracts relevant pieces of text inside a Web page, and recognizes them in a more structured format. A wrapper is a set of data extraction rules that converts semi-structured or unstructured data (such as XML and HTML data) into structured data (such as table and views in relational databases). Note that $\mathcal{DB}$ is used for structured data and $\mathcal{IR}, \mathcal{IE}$ are used for unstructured data.

A wrapper can be seen as a procedure that is designed for extracting the content of a particular information source and delivering the content of interest in a self describing representation. In the database community, a wrapper is a software component that converts data and queries from one model to another. In the Web environment, its purpose should be to convert information implicitly stored as an HTML document into information explicitly stored as a data structure for further processing. A wrapper for a Web source accepts queries about information in the pages of that source, fetches relevant pages from the source, extracts the requested information and returns the result. It consists of a set of extraction rules and the code required to apply these rules and is specific to one source. To extract information from several independent sources, a library of wrappers are needed.

A wrapper is an interface that (a) conveys requests from one system to a second system, (b) conveys replies from the second system to the first system, and (c) performs any necessary

lexical/ logical mapping. The other layer of a data integration system that does not exist in a traditional system is the wrapper layer. Unlike a traditional query execution engine that communicates with a local storage manager to fetch the data, the query execution plan in a data integration system must obtain data from remote sources. A brief survey of Web data extraction tools can be found in [48].

1.2 Web Data Extraction and Labeling

Data extraction from Web pages has been an active research topic since 1997. In the 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 have been 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 interactions and dynamic changes on Web sites are the obstacles that make Web data extraction difficult in real-life application scenarios [35].

Many Web documents contain an abundance of recognizable constants that together describe the essence of a document's content. For these kinds of data-rich, multiple-record documents (e.g., advertisements, movie reviews, weather reports, travel information, spots summaries, financial statements, etc.) we can apply a conceptual-modeling approach to extract and structure data automatically. We can automatically produce a database schema and recognizers for constants and keywords, and then invoke routines to recognize and extract data from unstructured Web documents and structure it according to the generated database schema, which can later be used in declarative ad hoc queries [27, 29–31].

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 have 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 tabular data annotation. Information Extraction ($\mathcal{IE}$) and labeling/annotation are two separate processes. Note that we are not working on wrapper generation at all. The wrapper generation technology is mature enough and a lot of work has been done in this field. But to the best of our knowledge, very little work has been done in terms of column name identification and data

annotation, which is a classification process. We will show that our labeling process yields better results on structured data such as wrapper generated Web page tables.

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 [70]. However, the quantity of data sources made available and their significant increase explain the need for non-materialized access to Web data [47].

There is a high demand for collecting data of interest from multiple Web databases. For example, a comparison-shopping system (e.g., shopping.com, pricegrabber.com, pricelist.com, ebay.com etc.) 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. In this research we wish to automate the labeling of the data generated from dynamic Web pages for scalable later use. Our work is further motivated by recent initiative of Google Base beta version for structured query over Web data [41].

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 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 research problem, and it represents a step toward the automatic extraction and manipulation of Web data [8]. 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 [51].

To minimize user effort in an Information Retrieval ($\mathcal{IR}$), $\mathcal{IE}$ 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 work has been done with regard to Web data extraction and wrapper generation. But most of the work exports the Web data as *anonymous* datasets, without assigning meaningful labels. To the best of our knowledge, there are only two *generic* methods for the automatic labeling of anonymous datasets: DeLa (Data Extraction and Label Assignment) and the *Labeler* 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 comes from, allowing the user to provide her own set of labels.

We propose a novel and a highly effective method for automatically labeling anonymous datasets 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 datasets can be materialized into a suitable relational Database ($\mathcal{DB}$) [3]. 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 hit counts 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 datasets. The Web 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 i.e., (*Hearst*) patterns 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 [4,6]. In our previous work [7,16–20,28,32,69], we have implemented a number of prototype systems such as WebFusion, PickUp, FastWrap, OntoBuilder, OntoMatch, etc., that are much more scalable approaches toward Web data integration.

1.3 Problem Statement/Description

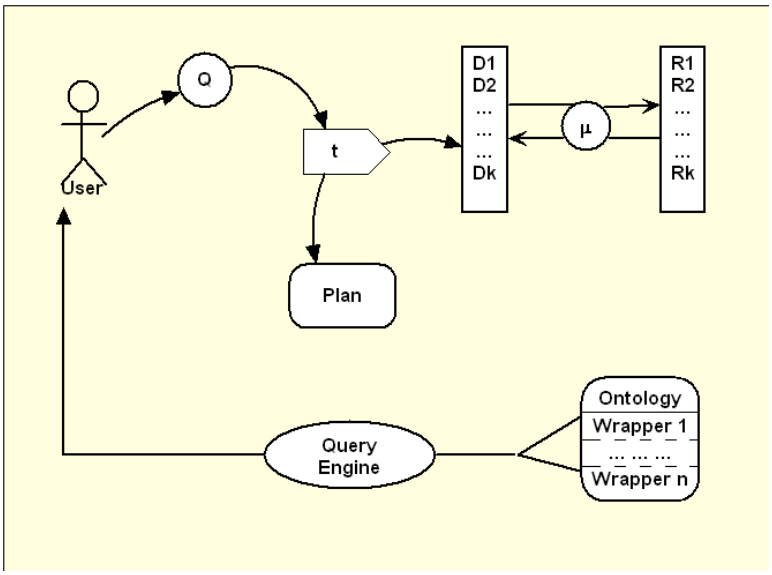

Figure 1.1: Resource Discovery/Description Mechanism

Figure 1.1 shows our overall envisioned system, resource capability discovery/description management system. Labeling component is the core of the system. In response to the user query submitted in 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. Resource Definition Statements (RDS) will be used to state what the system should expect from the Web resources. Thus our *resource capability discovery/description* will be focused toward handling Web forms and representing table schema. The rest of the report deals with the above two ideas.

Research hypothesis: Given a candidate set of labels (using σ function) and anonymous datasets (using ω function), can Web search engines such as Google, Yahoo, MSN be used as μ function to assign labels for the anonymous datasets? This research supports this hypothesis.

1.4 Use of Web Search Engine

In general the approach that we have taken to assign column label is to use the *hit counts* from Web search engines to disambiguate column labeling. In the literature the method has been termed 'self annotating Web' [21, 22], that is Web pages are used to annotate Web pages. For this approach, we can draw an analogy between Computer Network and Web Database. We had begun our journey from modern telecom network (connection oriented service), then we developed Internet Protocol (IP) (connectionless service), again we are in need of a telephone like network for *Quality of Service* (QoS). Similar cycle has been observed while labeling anonymous datasets. We had begun our journey from raw, unstructured text, then we developed structured text as $\mathcal{DB}$ (during the 1970's), again we are in need of raw, unstructured text (Web page text) for *Labeling*.

1.5 Research Motivation, Challenges, Goals and Contributions

Our research focuses on issues related to Web information retrieval and Web information integration. In particular, we are focusing on categorizing and classifying data collections into meaningful groups. This direction of research is essential in the field of autonomous information integration and data aggregation where autonomous systems collect information from Internet resources having no proper labels. In such situations, query systems fail to identify required information and cause the production of wrong responses. Our focus in this research is on generating useful labels for groups of data items to facilitate successful query computation. We try to do post processing, categorize items in a column that better describes the objects. In the absence of ontology, instead we use linguistic patterns. In the absence of

an existing ontology in a domain how to come up with terminology that are representative terms of the domain? BioFlow is a language to address Web data integration in an ad hoc manner. BioFlow deals with Web mashups and screen scraping techniques. We want to give a database abstraction for the Web, treat each Web page as relation. Relations have column names, Web tables may or may NOT have column names. Given a Web table, how to label the columns with different terminology? By solving our approach, the Web will become a synthetic extension of traditional database systems. This *dissertation goal* is to address all the above challenges from both theoretical and practical perspectives with the main focus on a generic, correct and efficient solution to the problem of labeling anonymous datasets. We bridge the gap between the *Terminology Layer* and *Data Layer* via the formalization of our proposed solution viz. Labeling Anonymous Datasets (Lads).

This dissertation makes the following *research contributions*:

- In this dissertation, we research the problem of labeling anonymous datasets. Our main focus is on a generic, correct and efficient solution to the problem of labeling anonymous datasets. We first formalize the problem as part of Deep Web Database Abstraction model

- We review the existing work on label assignment and propose to make it fully automatic

- We propose and define a novel label assignment problem

- We propose a specific method, *Web search engine based annotator*, to solve the problem without the support of domain ontology. In our proposed method, domain ontologies are not required any more. Specifically we bridge the gap between two orthogonal research viz. *wrapper generation* and *label extraction* for value added services such as online comparison shopping.

- We run experiments to test the proposed methods and show that the proposed methods are *effective*

- We categorize tabular Web data into three types: disjoint set column (DSC), repeated prefix/suffix column (RPS) and numeric column (NUM)

- We design a couple of ***efficient Labeling algorithms*** to facilitate the implementation of new operator (i.e., *link and combine*) [38] in relational databases: (1) LADS (2) NLADS (3) GLADS (4) LADSComplete [2]

- We make a procedural implementation of our approach in a Java framework

- Our algorithm LADS [5], [2] is guaranteed to work for disjoint set columns

1.6 Organization of this Dissertation

The remaining chapters of the dissertation are organized as follows: Chapter 2 reviews the researches on Web Wrapper and data annotation/labeling that are most closely related to our work; Chapter 3 presents our implementation on column name identification; Chapter 4 focuses on results, discussions and implementation; Finally, Chapter 5 concludes the dissertation and lists some of the remaining interesting research problems.

CHAPTER 2
RELATED WORK

2.1 Web Wrapper

Considerable research has been done in the area of Web Data Extraction viz. wrapper generation. In this chapter we will discuss some of the state-of-the-art works in wrapper generation, then confine ourselves to reviewing the research that is most closely related to the work we have done here. Wrapper Generation function ω for Web data can be stated as follows [48].

Definition 2.1.1 (Wrapper Generation) Given a Web page $\mathcal{S}$ containing a set of implicit objects, determine a mapping ω that populates a data repository $\mathcal{R}$ with the objects in $\mathcal{S}$, i.e., $\omega : S \mapsto R$. The mapping ω must also be capable of recognizing and extracting data from any other page $\mathcal{S}'$ *similar* to $\mathcal{S}$. $\diamondsuit$

In [67], the authors present a novel method called Information Extraction Knowledge Adaptation (IEKA) to solve the wrapper adaptation problem. They have analyzed the problem by identifying two kinds of features, namely *site-invariant* and *site-dependent* features.

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. In Data Extraction and Label Assignment (DeLa) [66], 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, DeLa employed four heuristics, but it is not very clear how the heuristics in DeLa are combined. There are many common cases where DeLa system cannot label data fields. For example, the DeLa heuristics miss a common case where a data field does not have a label (for example, the job title field on most job sites is a heading with no prompt or label). Also, the DeLa heuristic would be confused by sites with multiple fields of the same data type (for example, the *bedrooms, bathrooms*, and *maximum occupancy* fields

in vacation rentals) [15]. Arlotta et al. in the RoadRunner project have [8] developed several heuristics for labeling, however their system will fail in cases when descriptive labels are missing adjacent to data values. Visual Perception-based Extraction of Records (ViPER) mainly developed two heuristics: *inter-column* label assignment heuristic and *inner-column* label assignment heuristic to assign column label to data items [61]. These two heuristics are good to annotate a column where a repeated prefix or a repeated suffix will be found, those will be used to label the column. Automatic Data Extraction and Labeling (Adel) [49] system can automatically extract records from Web sites and semantically label the fields or mapping them to a schema. There are some shortcomings for the Adel system, such as: Adel is not able to process forms, instead it manually retrieves list pages and provides them to the system. 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 rules such as "sedan 4dr" and "4dr sedan" refer to the same concept. Our system can uniformly label those as "body." [52] proposed a multi annotator approach to tackle the annotation problem with each six basic annotator to exploit a different type of features. One of their approaches uses *Common Knowledge annotator*. Using our approach, we eliminate the need for externally supplied knowledge base as we use Web search engines as a collective knowledge source. All these approaches can NOT solve the problem of "local interface schema inadequacy problem" (some attributes in the results are not entirely contained in the Web form interface). [58] presents two algorithms: (i) Structural analysis of HTML form, and (ii) Confirming field annotations with probing. Our labeling method heavily depends on the correct extraction of Web form labels for which we also refer to work in HiWE [56] and labelEx [53].

2.2 Labeler Component of RoadRunner

The problem the RoadRunner project [25] solved may be stated as follows: "Given a set of sample HTML pages belonging to the same class, find the nested type of the source datasets and extract the source datasets from which the pages have been generated." Since Web pages are designed to be presented on a browser to a human user, usually labels and values are *visually close* to each other. Therefore, first the Labeler computes the coordinates of the bounding boxes of every label and data-value in a given sample page. Then it tries to find

the optimal association label/data-value by analyzing their spatial relationships. Arlotta et al. [8] have developed several heuristics to establish the correct associations:

- labels and values are close to each other
- usually a label is aligned vertically, horizontally or centrally 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 for the extracted attributes, which are initially anonymous.

The algorithm implemented by LABELER is illustrated in Figure 2.1: it takes as input a wrapper and a set of sample pages; as output, it produces a set of label/variant associations for the given wrapper. Initially it chooses an arbitrary order for the set of labels $L = \{l_i, i = 1 \dots N_l\}$ and for the set of variants $V = \{v_j, j = 1 \dots N_v\}$ of the wrapper. Then it calls the function LABELS to compute a set of label/variant associations (l_i/v_j) on each of the given samples.

LABELS tries to exploit the *spatial relationships* between labels and data-values in the graphical rendering of one Web page as displayed in Web browser. It computes an $N_l \times N_v$ matrix M of *scores*, that contains positive real numbers such that $M[i, j]$ is a measure of the "goodness" of l_i as label for v_j; as smaller are the scores as better are the associations. Some associations are immediately discarded without further evaluation by setting the corresponding score in M to $+\infty$. Namely (l_i/v_j) can be discarded if any of the following conditions hold: (i) l_i is below or to the right of v_j; (ii) the distance between l_i and v_j is greater than D_{max} pixels; (iii) l_i is placed on the diagonal of v_j; (iv) there exists another variant/invariant which is located between l_i and v_j. The first three conditions reflect the fact that a label is usually placed above or to the left of the corresponding data-value, never too far from it. The last condition considers that anything between a data-value and its label would keep the reader from intuitively associating them. Given a pair (l/v) and the corresponding bounding-boxes on a page s, the score function is defined as follows:

$SCORE(l, v, s) = DISTANCE(l, v, s) \times sin(2 \times ALIGNMENT(l, v, s))$. In the following,

Algorithm LABELER
Parameters : D_{max}, the maximum allowed distance from a variant to its label;
α, a threshold for the SCORE function;
Input : a wrapper w, a set of samples $S = \{s_1, \ldots, s_n\}$;
Output: a set $\{(l/v)\}$ of associations;
begin
Let $L = \{l_i, i = 1, \ldots, N_l\}, V = \{v_j, j = 1, \ldots, N_v\}$
be respectively the labels and the variants in w;
Let A_k **be** LABELS(L, V, s_k), $k = 1, \ldots, n$;
Let A **be** the set of associations (l/v) such that l is associated only to v
and v is associated only to l in $\bigcup_{k=1,\ldots,n} A_k$;
return A;
a set $\{(l_i/v_j)\}$ LABELS(a set of labels L, a set of variants V, a sample s)
begin
Let M be an $N_l \times N_v$ matrix such that
$M[i, j] = +\infty, if\ validPosition(l_i, v_j, s)$ **or** $inbetween(l_i, v_j, L, V, s)$ **or** $score(l_i, v_j, s) \geq \alpha D_{max}$
$= score(l_i, v_j, s)\, otherwise$
Let A **be** the empty set;
do begin
add $(l_{i_{min}}/v_{j_{min}})$ to A such that $M[i_{min}, j_{min}]$ is the minimum of M
set $M[i_{min}, h]$ and $M[k, j_{min}]$ to $+\infty$, $h = 1, \ldots, N_l$, $k = 1, \ldots, N_v$;
while $(\exists M[i, j] \neq +\infty)$;
return A;
end
boolean validPosition (label l, variant v, sample s)
return (in the graphical rendering of s, l is placed either above or to the left of v
and their distance is less than D_{max});
boolean inbetween (label L, variant V, sample s)
return (there exists either a label or a value whose bounding-box
in the graphical rendering of s is between those of l and v);
$\Re^+$ distance(label l, variant v, sample s)
return (the minimum distance between bounding boxes of l and v in the rendering of s);
$\Re^+$ score(label l, variant v, sample s)
return $SCORE(l, v, s) = DISTANCE(l, v, s) \times sin(2 \times ALIGNMENT(l, v, s))$

Figure 2.1: The Labeling Algorithm of RoadRunner

we discuss some of the limitations of the section 2.2. The approach needs that textual labels describing the meaning of the extracted fields are present in the Web page. However, many data are published on Web pages leaving their meaning implicit. It is worth saying that labels may be present as images and not as texts as we have seen in book and movie rating/ranking information. In fact, often pages do not include explicit labels for those data whose semantics can be clearly understood from the context (e.g., the title of a book in a page describing details about that book). In our research, we have addressed that problem

by introducing a new paradigm called *Web Search Engine Based Annotator*. Specifically, we could label the unlabeled data-values in the anonymous datasets by looking for labels in the Web search results.

2.3 PANKOW and C-PANKOW

PANKOW (Pattern-based ANnotation through Knowledge On the Web) uses statistical and pattern-matching techniques to automatically discover relevant concepts in the document. In particular, PANKOW generates instances of *lexico-syntactic* patterns indicating a certain semantic or ontological relation and counts their occurrences in the WWW using the Google^{TM} API. In the following, we briefly discuss the general process of PANKOW, consisting of three steps:

Input: A set of entities (instances or concepts) to be classified with regard to an ontology.

Step 1: The system iterates through the set of entities to be classified and generates instances of patterns, one for each concept in the ontology. For example, the instance '*South Africa*' and the concepts `Country` and `Hotel` are composed using a pattern schema of a set of patterns and resulting in pattern instances like '*South Africa is a country*' and '*South Africa is a hotel*' or '*countries such as South Africa*' and '*hotels such as South Africa*'.

Result 1: Set of pattern instances

Step 2: Then, Google^{TM} is queried for the pattern instances through its Web service API. The API delivers as its results

Result 2: The counts for each pattern instance

Step 3: The system sums up the query results to a total for each concept

Result 3: The statistical fingerprint for each entity, i.e., the results of aggregating for each entity, the number of Google hit counts for all pattern instances conveying the relation of interest.

Given an unknown instance or concept on a certain Web page, patterns respectively indicating an *instance-of* or *subconcept* relation are instantiated for the new instance or concept and each concept in the target ontology. Finally, given the statistical fingerprint of the instance or concept, PANKOW follows a *principle of disambiguation by maximal evidence* thus assigning the instance or concept to that concept in the statistical fingerprint.

C-PANKOW (Context-driven PANKOW) [21] alleviates several shortcomings of PANKOW. *First*, by downloading abstracts and processing them off-line, C-PANKOW avoids the generation of a large number of linguistic patterns and a correspondingly large number of Google queries. *Second*, by linguistically analyzing and normalizing the downloaded abstracts, C-PANKOW increases the coverage of the pattern matching mechanism and overcomes the several limitations of the earlier pattern generation process. *Third*, C-PANKOW uses the annotation context in order to distinguish the significance of a pattern match for the given annotation task. In the following, we present the C-PANKOW algorithm. The complexity of C-PANKOW is linear in the number of instances and hence in the size of the document. The number of queries sent to the Google API is constant for each instance to be annotated.

Algorithm 1 C−PANKOW: Context-driven PANKOW

```
C − PANKOW (document d)
/* recognize all the instances in input document */
I = recognizeInstances(d)
for each i ∈ I do
  for each (p, c) ∈ P do
    /* download the n first Google abstracts matching the exact query c(i) */
    Abstracts = downloadGoogleAbstracts(c(i), n));
    for each a in Abstracts do
      /* calculate the similarity between the document and the Google abstract a */
      sim = calculateSimilarity(a, d);
      if (sim > t) then
        if (p.matches(a)) then
            c = p.getConcept();
            Res[c] = Res[c] + sim;
        end if
      end if
    end for
  end for
  annotate(i, maxarg_c Res[c]);
end for
```

2.4 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, [66] 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 set the stage for more fully automatic data annotation of Web sites.

2.5 ODE

[63] presents Ontology-Assisted Data Extraction, a complete Data Extraction workflow system. The algorithm labelAssignment shown below is used to find the label sequence with the largest probability.

Algorithm 2 labelAssignment()

1: *Assign labels for* d_1, *find top* k *labels with largest probability, set* s_{1j}, $1 \leq j \leq k$ *accordingly*
2: **for** $i = 2$ *to* n **do**
3: **for** $j = 1$ *to* k **do**
4: *Assign labels for* d_i, *given* $s_{(i-1)j}$ *as previous label sequence, and*
5: *append each label to* s_{ij} *to make a new sequence*
6: **end for**
7: *Find* k *label sequences with the largest probability from the newly generated*
8: *label sequence and set* s_{ij} $1 \leq j \leq k$, *correspondingly*
9: **end for**
10: *Label the data value using the label sequence* s_{n1}, *which has the largest probability*

2.6 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 [61].

2.6.1 Inter Column Label Assignment

[61] have 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 tests 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.

2.6.2 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 appears as "Save: $10.00 (13%)." The fix token here (save:), which is not numeric type, is removed from individual rows and added as label.

In addition to the above two heuristics, [61] also advocates 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" appears only in the "Mega pixels" field, we can split "Canon X300 8MP" into two text segments, thereby adding more semantics on the labeled data.

2.7 Multi Annotator

[52] have 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. [52] have defined six 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. *First*, 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. *Second*, it uses both the local interface schema (LIS) as well as an integrated interface schema (IIS) 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 illustrate how the use of the integrated interface schema can help alleviate the *local interface schema inadequacy problem* and the *inconsistent label problem*.

2.8 Comparison with other Annotation Tools

In the following, we compare our work with some other tools that have been reported in the literature. [36] addresses the issue of semantic annotation using *horizontal* and *vertical* contexts. Horizontal context look for information left to and right to the targeted instance. Our label assignment is based on left horizontal context only. This approach is sufficient specially for DSC columns. We have found that right horizontal context may be appropriate for numeric column (example from Hotel domain: 5 BR (BR refer to Bed Room)). [68] reports extracting data from lists and grouping them by rows and columns. They perform list data extraction in two procedures: separator selection and clustering based list extraction. Clusters indicate columns of the list. Our work post process the clusters and assign suitable label for the columns. [37] reports fully automatic Web data extraction tool ViPER. They have introduced the notion of *user vocabulary* that has been inspired by the idea of social bookmarking. Their annotation/labeling process is manual, i.e., user has to indicate which tag will belong to which column. Using our method the labeling process can be made fully automatic. [45] report MSAA (Multi-Source Automatic Annotation) using 4 types of validate queries: "C : instance"; "C = instance"; "C (instance)" and "C instance". The last validate query is the same as ours "L V" pattern. We have reduced the number of validate queries from four types to one and still our solution is very promising. As our target is free text Web pages, we believe that label value co-occur as "L V", i.e., in free text Web page this is the most dominant pattern.

2.9 Misc. Related Work

[27, 31] have suggested a different approach to the problem of schema matching, one which may work better for the heterogeneous HTML tables encountered on the Web. Embley et al. have transformed the table location problem and the schema matching problem into an extraction problem by making use of so-called *extraction ontologies* and inferred the semantic correspondence between a source table and a target schema. [54, 55] have shown an approach TARTAR for automatic table data annotation.

[50] report five heuristics for *Local Label Assignment*. Out of five heuristics, four heuristics are the same as DeLa heuristics, one new heuristic they introduced is as follows: *search for labels in source code*, basically to look for a label in the Cascading Style Sheet (CSS) code. But there is no guideline as how far from data value we will look for a label in the CSS code, the heuristic seems not to be very useful. Their notion of *Global Label Assignment* is similar in vision as ours, i.e., look for labels in other data sources.

[11–14] in a bid to uncover the relational Web have presented WebTables. In the project they extracted 14.1 billion HTML tables from several billion page portion of Google's general purpose Web crawl and estimated that 154 million of these tables contain high-quality relational-style data. According to their finding, the number of columns are in the range $2, \ldots, 9$, which comes from more than 55% of the tables. We will make use of this range figure in the complexity analysis of our algorithm Lads in Section 3.8. Related recent published works in our research group are as follows: [9, 10, 18, 28, 39, 40, 44, 62].

CHAPTER 3
COLUMN NAME IDENTIFICATION

This chapter discusses our main contribution for the thesis, autonomous label assignment for wrapper generated tables. We have developed an algorithm, LADS, which can holistically assign label for tabular Web document. We classify Web tables into three categories: disjoint set column (DSC), repeated prefix/suffix column (RPS) and numeric column (NUM). Our algorithm, LADS, yields very promising results for labeling disjoint set columns [5], [2].

3.1 Problem Statement

We want to give a database abstraction for the Web and treat each Web page as relation. Relation has column names but Web table may or may NOT have column names. Given a Web table, how to label the columns with different terminology?

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

Definition 3.1.1 (QC) A Query Condition (QC) is a 3-tuple: $\{\mathcal{L}, \Xi, \mathcal{V}\}$, where $\mathcal{L}$, Ξ and $\mathcal{V}$ are set of labels, relational operators and instance data values respectively. Ξ is any relational operator such as $=, \leq, \geq, \neq$ etc. $\diamondsuit$

Each Query Condition can be modeled using SQL syntax as:
SELECT $\{attr_{r1}, attr_{r2}, \ldots, attr_{rm}\}$

FROM DB

WHERE $attr_{q1} = val_{q1}, attr_{q2} = val_{q2}, \ldots, attr_{qn} = val_{qn}$

where val_{qi} is the corresponding attribute value filled into the query form. We model the dynamic Web site as $S \subseteq Q \times R$, where Q is the query interface schema and can be represented as $Q \subseteq F \times P$ and R is the result schema, can be represented as $R \subseteq L \times V$ [46]. The semantics of the above definition are as follows: we have a set of Web form labels $\mathcal{F}$ and a set of corresponding parameters $\mathcal{P}$. In response to the query submitted in Web form, we will have tabular data which will have a set of values $\mathcal{V}$. The label $\mathcal{L}$ of the values $\mathcal{V}$ may or may NOT be present in the Web page. The following Figure 3.1 depicts the overall scenario of the problem that we are going to model and address. Our model assumes that in response to

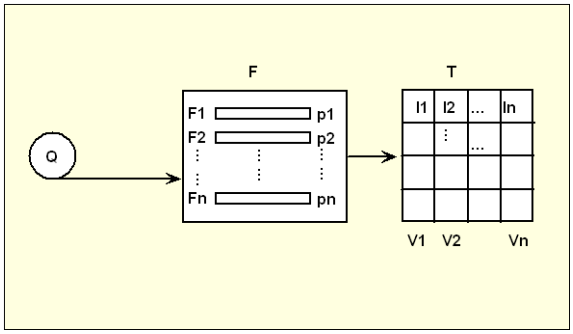

Figure 3.1: Deep Web Database Abstraction Model

user query submitted in Web form, we will have tabular Web data. Actually according to a literature survey, the depth of a document is about five, i.e., in order to find actual tabular Web data, we have to navigate about five links/forms.

Suppose a user wants to purchase computer network switch. In order to illustrate our model, we present the following tables: 3.1, 3.2 and 3.3 from comparison shopping sites www.shopping.com, www.pricegrabber.com and www.ebay.com. Due to space limitations, we choose only to show the tables as anonymous datasets. Readers should be convinced that the tables have been extracted from hidden Web by filling in Web form from the respective Web sites. Using our method we can efficiently label the columns A_1 and A_2 as *brand* and *model* respectively.

Table 3.1: Anonymous datasets about computer network switch

A_1	A_2	A_3	A_4	A_5	A_6
D-Link	DGS-2208	8-port	10/100/1000	4.5 stars	$38.79
Linksys	EG005W	5-Port	10/100/1000	4 stars	$38.99
Cisco	WSC2960G48TCL	48-port	1000		$3090.00

Table 3.2: Anonymous datasets about computer network switch, src: shopping.com

A_1	A_2	A_3	A_4	A_5	A_6
D-Link	DGS-2208	8-port	10/100/1000	3 stars	$30 - $80
Linksys	EG008W	8-Port	10/100/1000	3.5 stars	$60 - $100
Cisco	WSC296024TTL	24-port	1000		$500 -$1,922

Table 3.3: Anonymous datasets about computer network switch, src: ebay.com

A_1	A_2	A_3	A_4	A_5	A_6
D-Link	DGS-2208	8-port	10/100/1000	$58.31	25d 23h 51m
Linksys	EG008W	8-Port	10/100/1000	$54.99	9d 5h 1m
Cisco	WSC296024TTL	24-port	1000	$2739.99	

3.2 Motivating Example

The problem of column name identification degenerates to finding labels extracted from Web form labels as well as from user SQL query variables, then associating the extracted labels to the attributes in anonymous datasets extracted by Web wrappers. Once the labels have been identified, then assign the labels to the tabular data to assign column label. The problem is how to relate a label to the data value instances of an output Web page table? In other words, given a set of data values (homogeneous), how to come up with a suitable name/label for the data values? 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 examples: given $\{Apple, Orange, Banana\}$, assign label of the data as *fruit*. One more example is as follows: given the set containing $\{India, Pakistan, USA, UK\}$, then it has to be labeled with the category *country* name.

In response to user query $\mathcal{Q}$ submitted in the Web form $\mathcal{F}$, we will have tabular data $\mathcal{T}$. Semantic enrichment of the tabular data will be made semi-automatically and through human knowledge. One of the major fundamental research problem is how to find $\mathcal{L}$, given $\mathcal{Q}, \mathcal{F}, \mathcal{P}$ and $\mathcal{V}$. Lets consider the following motivating example.

Example 3.2.1 (Motivating Example 1) Consider the following user SQL query for the site S1, i.e., http://www.amazon.com

```
SELECT Title, Rating
FROM Site S1
WHERE Title like "Discrete Mathematics"
AND Rating=5;
```

♢

amazon.com site shows results pages, one example as Discrete Mathematics by Kenneth Rosen, but there is no label as title, which is understood by user from the context.

3.3 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. The distinguishing between static structure (i.e., labels are invariant) and dynamic content (i.e., data values are variant) is done by wrapping tools such as RoadRunner, PickUp, FastWrap. The assignment of a suitable label is an open problem, which we will solve by using a series of steps including Web knowledge. Both the sets, the label set $\mathcal{L}$ (for example BAMM datasets [42]) as well as the value set $\mathcal{V}$ (for example RoadRunner datasets [43]) are known – the task of the label assignment function μ is to identify correspondence between $l_i \in \mathcal{L}$ and $v_j \in \mathcal{V}$. A knowledge source can be applied in this scenario as a 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 case are Web 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. The main idea herein is to approximate semantics by considering information about the statistical distribution of certain syntactic structures over the Web. The ontological instance in question is then annotated semantically according to a *principle*

of maximal evidence, i.e., with the label having the largest number of hits. Let's assume that

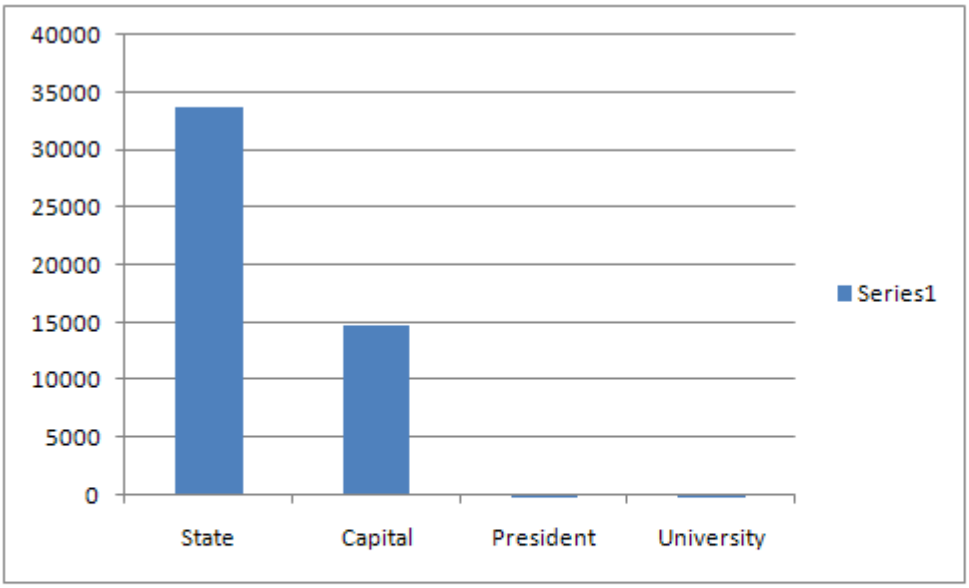

Figure 3.2: Statistical distribution of 'is a' patterns for Washington

the string 'Washington' appears in a Web table and we have no idea about how to annotate it. Figure 3.2 shows the Google hits for the following four expressions: *Washington is a State, Washington is a Capital, Washington is a University, Washington is a President.* The best k candidate labels are then presented to the user to choose the label that fits. Intuitively, given this figure 3.2, we would naturally tend to annotate *Washington* as a state as it seems to be its main meaning on the Web. This idea can be traced back in the seminal work Pointwise Mutual Information for Information Retrieval (PMI-IR) by Turney (2001) [65]. We have done some experiments with Web search engines such as Google, Yahoo and MSN. The details are discussed in the section *Affinity Based Speculative Labeling.*

Let's consider another example. Suppose we have to label/annotate 'Michael Jordan' to one of the following professional categories: $\{BasketballPlayer, Footballer, Politician\}$. We

Table 3.4: Possible labels for different Domain

Domain	Manually selected labels
Book	Title, Author, Price, ISBN, Publisher, Publication Date
Automobile	Make, Model, Price, Year, Mileage, Color
Music	Artist, Title, Album
Movie	Actor, Director, DirectedBy, Film, Movie, Title
Watch	Band, Brand, Display, Gender, Model, Price
Political	President, VP, Party, Senator, Governor
Demographic	City, County, Country, Population, Capital
Athlete	Name, Gender, DOB, Country, Discipline
Research Publication	Title, Author, Date, Journal, Conference, Publisher, Pages, Volume
Digital Camera	Brand, Model, Series, Color, Type, Image Sensor, Megapixel, Zoom
Online Social Network	Name, Email, Education, Employment, Gender, Age, Country, City, Hometown, Zip
Product Information	Brand, Model, Merchant, Type, Attribute

can issue a number of search queries as follows: Michael Jordan (Basketball Player), Michael Jordan (Footballer), Michael Jordan (Politician). This is exactly the *disambiguation and representation strategy used in WikiPedia*. So our approach is not a brand new one, but we are tailoring it to label columns on the fly to facilitate Web data integration.

3.3.1 Example of Candidate Set of Labels

Each site will have a predetermined set of labels, which represent the search capabilities in a site (in our case those will be extracted from Web form labels as well as from user SQL query variables), those will be used to label the anonymous datasets. We consider the set of potential labels as controlled vocabulary for the domain of interest. It is like an ontology that will provide a common vocabulary to support the sharing and reuse of knowledge, which has schema only, not any instances of data. We consider the labels as a set of lexical tokens or terms. Also note that we are going to label a *single Web table* at a time. Labels are useful for building concept taxonomies (like *age* which can be *infant, adolescent, old* etc). Table 3.4 shows the candidate set of labels by which we have done our experiment. Our laborious method of column labeling will help to build an ontology, which is useful in intelligent information integration [1], later can help users to write queries. We will discuss these issues further in the section *Candidate Label Selection*.

3.3.2 Language Patterns

The rationale for using language patterns is that certain English text fragments indicate semantic relationships between terms. First, Hearst has applied regular expression patterns indicating *lexico-syntactic* patterns [34] for extraction of semantic relations from the text. Her underlying idea is very simple and can be easily applied to the Web to derive relational information from the Web. Normal regular expressions capture recurring expressions and map the results of the matching expression to a semantic structure, such as taxonomic relations between concepts. Cimiano and Staab [22] 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 labeling context are the following:

P1 <L> <V>

H1 <L>s such as <V |L'>

H2 such <L>s as <V |L'>

H3 <L>s, (especially | including) <V |L'>

H4 <V |L'> (and | or) other <L>s

D1 the <V |L>

D2 the $< L|V >$

A <V |L'>, a <L>

C <V |L'> is a <L>

"L V" patterns have previously been used in OntoBuilder [32] to find labels in Web forms. The Idea was that the annotation of objects either precede them or are above them as in tables, and never below or behind. And we are exploiting that observation. The above patterns would match the following expressions:

artist Miles Davis (P1)

artist such as Miles Davis (H1)

album such as Kind of Blue (H1)

such albums as A Love Supreme (H2)

artists, especially Miles Davis (H3)

John Coltrane *and other artists* (H4)

the Kind of Blue album (D1)

the album Kind of Blue (D2)

3.3.3 Labeling Algorithm

Before presenting the algorithm, the following definition of *Speculative Query* will be helpful to make understand how we pose queries to Web search engines.

Definition 3.3.1 (Speculative Query) A speculative query consists of 3 parts: $\{\mathcal{L}, \mathcal{P}, \mathcal{V}\}$, where $\mathcal{L}$, $\mathcal{P}$ and V are set of labels, patterns and instance data values respectively. $\diamondsuit$

In order to overcome the limitation of keyword based search paradigm, we formulate speculative queries to Web search engines as exact phrase matching. Example: "artist Miles Davis" is a speculative query.

Assume we are given wrapper generated anonymous datasets. The step by step procedure to solve the labeling problem in our context is as follows:

- Step1: Extract label from Web form (refers to work in HiWE [56] and labelEx [53]) as well as from user SQL query variable (select clause)

- Step2: Formulate and execute speculative queries to Web search engines (*e.g., Google, Yahoo and MSN*)

- Step3: Prune and rank results (*i.e., compute statistical fingerprints*) [22]

The following screen shot figure 3.3 shows the advanced book search option of amazon.com. The set of labels that can be extracted from the Web form are:
$\{Keywords, Author, Title, ISBN, Publisher, Subject\}$
Also, we wish to capture labels from user SQL query variable. An example is as follows:

```
SELECT Title, Rating
FROM amazon.com
```

Algorithm 3 LADS: Labeling Anonymous Datasets

INPUT
$\mathbf{L}(1 \ldots m)$: *A set of labels*
$\mathbf{A}(1 \ldots n)$: *A set of anonymous attributes*
Precondition $m > n$
$\mathbf{P}(1 \ldots k)$: *A set of patterns*
$\mathbf{V}(1 \ldots t, 1 \ldots n)$: *Tabular Data Value*
VARIABLE
$\mathbf{N}$: *Google hit counts*
$\mathbf{H}(1 \ldots n, 1 \ldots m)$: *2D array to store hit counts*
$R \rightarrow 1 \ldots t$ *Random indexes between* $1 \ldots t$ *generate one time*
Formulate Speculative query : $L \times P \times V \rightarrow N$
for $i = 1$ to n **do**
  $H[i][1..m] = 0$
  for *each* $r \in R$ **do**
    $iValue = V_{ri}$
    for $j = 1$ to m **do**
      $lValue = L[j]$
      for $p = 1$ to k **do**
        $QueryVar = lValue + P[p] + iValue$
        $N \leftarrow GoogleExecuteQuery(QueryVar)$
        $H[i][j] += N$
      end for
    end for
  end for
end for
for $i = 1$ to n **do**
  Let z *be index*$(1..m)$ *of highest count in* $H[i][1..m]$
  $A[i] = L[z]$
end for

```
WHERE Title like "Discrete Mathematics"
AND Rating=5;
```

By analyzing the advanced search option of amazon.com site, we see that the *Title* attribute can be mapped to query Interface Schema (IS), whereas the *Rating* attribute can be mapped to Result Schema (RS). For the above SQL query example with the query Interface that are available in the amazon.com site, user query can be satisfied only for the *Title* attribute, NOT for the *Rating* attribute. Consequently, the user will see a lot of tuples containing all the *Ratings* returned in response to query, whereas she may only be interested to see records corresponds to Rating 5. The above phenomena leads to what we called *local interface schema*

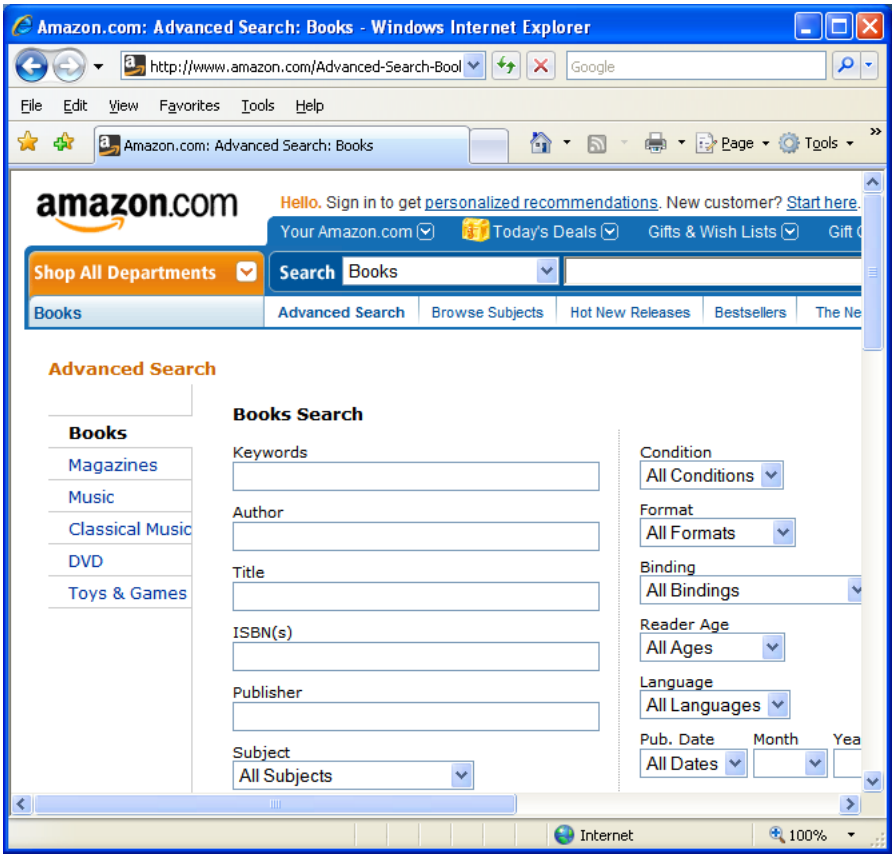

Figure 3.3: Web Form Interface

inadequacy problem, i.e., for the above example, amazon.com site can NOT take *Rating* as an input parameter to be submitted in the Web form, whereas from book search result page, Rating can be found. The following definition of Local Interface Schema (LIS) will be helpful to clarify the concept.

Definition 3.3.2 (LIS) Local Interface Schema (LIS) is a set of attributes. For a Web database, the query form often contains some attributes of the underline database. ♢

Formally, we denote a local search interface schema (**LIS**) as $S_i = \{F_1, F_2, \ldots, F_k\}$, where each F_j is a Web form label attribute. When a query is submitted against the search interface, the entities in the returned results also do have a certain "hidden" schema, denoted as $S_e = \{f_1, f_2, \ldots, f_n\}$, where each $f_j (j = 1, \ldots, n)$ 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* [59]. The schema of the retrieved data

and the interface schema usually share a significant number of attributes. Therefore, if an attribute f_t in the search results does have a matched attribute F_t in the LIS, all the data units identified with f_t can be labeled by the name of F_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. Therefore we are in need of IIS and AMT which are defined as follows.

Definition 3.3.3 (IIS) Integrated Interface Schema (IIS). For Web databases belonging to a given domain, IIS contains all the information of each LIS. That is through schema mapping, we can integrate each LIS into a holistic IS. ♢

Definition 3.3.4 (AMT) Attribute Map Table (AMT) Let X be the set of attributes in LIS, and Y be the set of attributes in IIS. Then AMT $\subseteq X \times Y$ such that a condition θ holds, is called an Attribute Map Table. ♢

Intuitively, AMT contains all mapping relationships between IIS and LIS.

We have manually computed candidate set of labels for 4 domains: Book, Automobile, Movie and Music (BAMM) from UIUC Web data integration repositories [42]. We have shown the set of labels in the table 3.5. The datasets have been crafted from around 50 deep Web sources. The candidate set of labels for the above mentioned 4 DOMAIN can be thought to be robust and satisfies most user queries. Again, the user is free to pose queries to site using her own set of labels. It is our argument that what the user will choose as input candidate labels are nothing but a variant of what the SYSTEM has in its knowledge base (i.e., *label library* [64]), i.e., the user supplied labels can be some ontological variation of the SYSTEM's label: e.g., isA, synonym and part-of relationship. Our assumption for the above is also supported by *schema dictionary* that defines the mapping relationship between each column in instance lib with the labels in schema dictionary [50]. However in the absence of IIS and AMT, there will be no label in the search interface that can be assigned to the discovered data units of this *Rating* attribute, then the only hints to label the attribute is to extract label from the user SQL query variable. The set of labels that can be extracted from user SQL query variable for the above example is $\{Title, Rating\}$.

Table 3.5: Manually computed set of labels, source: BAMM datasets

Domain	Manually computed labels
Book	Title, (Author, First Name, Last Name), Subject, Price, (Publish, Publisher), Category, (Format, Binding), Publication Date, ISBN, Keyword
Automobile	Make, Model, Price, Year, Mileage, Color, (Class, Type, Style, Category), (Zip Code, Area, State)
Movie	Title, (Actor, Artist), Director, Format, (Type, Genre, Category), (Star, Rating), (Cast/Crew, People), Price, Studio, Keyword
Music	Artist, Album, (Song, Title), Style, Soundtrack, Band, Genre, Label, Catalog number, Category, Keyword, Format

Our label selection function σ will prepare the final candidate set of labels $\mathcal{L}$ by taking the union of all the sources: Web form label tag $\mathcal{F}$ and user query variable $\mathcal{Q}$. Thus $\mathcal{L} \subseteq \mathcal{F} \cup Q_s$. Sub index s refer to user SQL query variable in *select* clause.

3.3.4 Probabilistic Labeling of Anonymous Datasets

An anonymous datasets are a structured collection of data in which descriptive labels for similar objects are missing, e.g., table 3.6 shows anonymous datasets about music, collected from *www.allmusic.com*. These datasets are 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 $R(A_1, A_2, A_3, A_4)$, also all the anonymous attributes A_j have well-defined domains (e.g., A_1 contains *artist* names, A_2 contains *album* names, A_3 contains *year* and A_4 contains *user rating*). We assume that the domains D_j, i.e., *semantic type* of the data value instances of A_j are pairwise disjoint i.e $\forall_{i,j}\ D_i \cap D_j = \phi$. Based on the above ideas, we provide the following definitions of Unannotated Table (UT) and Disjoint Set Column (DSC).

Definition 3.3.5 (UT) Unannotated Table (UT) is a relation $R(A_1, A_2, \ldots, A_n)$ where the descriptive labels for similar objects in a column are missing. ♢

Table 3.6 is an example of an Unannotated Table.

Definition 3.3.6 (DSC) Disjoint Set Column (DSC) in an Unannotated Table $R(A_1, A_2, \ldots, A_n)$, each anonymous attribute A_j is of domain (i.e., semantic type) D_j such that $\forall_{i,j}\ D_i \cap D_j = \phi$.
♢

Example in table 3.6, A_1 contains *artist* names, A_2 contains *album* names, they are Disjoint Set Column (DSC). We categorize columns into three types: Disjoint Set Column (DSC), Repeated Prefix/Suffix Column (RPS) and Numeric Column (NUM). Below we formally define the RPS and NUM.

Definition 3.3.7 (RPS) Repeated Prefix/Suffix Column (RPS) in an Unannotated Table, $R(A_1, A_2, \ldots, A_n)$, if all the instance data values in an anonymous attribute A_j contains the same prefix or same suffix, then the column is said to be Repeated Prefix/Suffix Column (RPS). $\diamondsuit$

Example in table 3.6, A_4 contains *stars*, it is Repeated Prefix/Suffix Column (RPS)

Definition 3.3.8 (NUM) Numeric Column (NUM) in an Unannotated Table, $R(A_1, A_2, \ldots, A_n)$, if all the instance data values in an anonymous attribute A_j contains more digits $0, 1, \ldots, 9$ than characters i.e., $a, b, \ldots, z$, then the column is said to be a Numeric Column (NUM). $\diamondsuit$

Example in table 3.6, A_3 contains *1959*, it is Numeric Column (NUM).
To reduce the computational complexity, we employ a random *sampling function* ψ, which given an *UT* T, returns a subset of it T', $\psi : T \mapsto T', where\ T' \subset T$. In our experiment, the function ψ returns random records. We choose to do our experiment with 1, 3, 5, 7, 9 number of random records T' from T. In order to label anonymous datasets $R(A_1, A_2, \ldots, A_n)$, we are in need of a candidate set of labels $\mathcal{L}$. Then we apply our label assignment function μ, which takes as parameter T' and a candidate set of labels $\mathcal{L}$ and the function will assign label for $A_1, A_2, \ldots, A_n$, i.e $\mu : (T', \mathcal{L}) \mapsto A_1, A_2, \ldots, A_n$, where $\forall_{1 \leq i \leq n}\ A_i \in \mathcal{L}$.

The labeling problem which we deal in this research is that of assigning descriptive labels for anonymous datasets. In doing so, we employ our ω, σ, ψ and μ functions. The function μ uses Web search engines *hit counts* to measure the probabilities which we will discuss more in detail in the section *Affinity Based Speculative Labeling*. Labeling problem can be solved in two steps: finding a good candidate set of labels, i.e., σ function and finding the best matching between labels and the anonymous attributes, i.e., μ function.

Table 3.6: An anonymous datasets about music, containing a single relation R

R	A_1	A_2	A_3	A_4
	Miles Davis	Kind of Blue	1959	5 stars
	John Coltrane	A Love Supreme	1964	5 stars
	John Coltrane	My Favourite Things	1960	4.5 stars

3.3.5 Affinity Based Speculative 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 (i.e., semantic type) 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 set of 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 datasets. *First*, we need a way of measuring how well a labeling $\mathcal{R} \rightarrow L^n$ describes the domains of the attributes in $\mathcal{R}$. *Second*, the cost of the labeling algorithm must not be too high. We note here in passing that the labeling problem can be rephrased to finding a *maximum-weight matching in a complete bipartite graph* $G(V, E)$ where the vertex sets are the columns in $\mathcal{R}$ and $\mathcal{L}$ respectively and the weight of each edge (A_j, l_i) indicates how well l_i describes A_j.

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 assume $P(l_i)$ to be 1 in this research. More formally, we have established in line with the probabilistic foundations [57], [26] that $P(l_i|A_j) \propto P(A_j|l_i)$. We estimate the true affinity between labels and values 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 Web search engines classify as relevant answers for that query to estimate the probabilities above. *We have used Web search engine's hit counts based on first time success, meaning if the search engine returns some hit counts value we use it, otherwise sometimes a Web search engine asks questions that the query did not produce any results, whether the user*

wants to formulate query without quote etc. For those cases, we assume the hit counts to be zero.

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 $\mathcal{D}$ 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 clarify the use of speculative queries to find Label Attribute Affinity.

Definition 3.3.9 (Document Count (DC)) 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 3.3.10 (LAA) Given an unannotated table $R(A_1, \ldots, A_n)$ and a candidate set of labels $L = \{l_1, \ldots, l_m\}$, the Label Attribute Affinity between A_j and l_i, denoted $\mathrm{LAA}(A_j, l_i)$, is defined as

$$LAA(A_j, l_i) = P(A_j|l_i) = \frac{1}{|A_j|} \sum_{x=1}^{|A_j|} \frac{DC(l_i \wedge v_x)}{\sum_{y=1}^{m} DC(l_y \wedge v_x)}$$

♢

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 label l_i and value $v_x \in A_j$ are likely to appear together in a Web document.

This section describes our implementation of the labeling method and our strategy to formulate speculative queries. We illustrate the discussion using the anonymous datasets in table 3.6 and a candidate set of labels $L = \{artist, title, album\}$. Speculative query $l_i \wedge v_x$ 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. Our answer is that we use speculative query to Web search engines. Table 3.7 compares the number of results of speculative queries using the *phrase* approach, which uses speculative queries asking for documents containing a phrase formed by label and value (e.g., "artist John Coltrane"). Table 3.8 shows the affinity between the labels and anonymous attributes, computed as in definition

Table 3.7: No. of answers to speculative queries among different search engines

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

Table 3.8: LAA Based on Different Search Engine Result

Probability	Google	Yahoo	MSN
P(A_1| artist)	0.8153	0.8311	0.8163
P(A_1| title)	0.0655	0.029	0.0658
P(A_1| album)	0.0194	0.1395	0.1174
P(A_2| artist)	0.003	0.001	0.0108
P(A_2| title)	0.0579	0.1534	0.0593
P(A_2| album)	0.9358	0.8451	0.9296

LAA, using the *DC* values in table 3.7. From table 3.8, we label anonymous attribute A_1 to be *artist* (with 81% probabillity) and A_2 to be *album* (with 93% probability) (*probabilities computed by Google hit counts*).

From table 3.8, we see that there is an ordered rank list of probabilities associated with each label l_i to an anonymous attribute A_j. Therefore we need to prune the results by what we so called *Statistical Fingerprints*. The details of mathematical and logical foundation of *Statistical Fingerprints* are discussed in the following section.

3.4 Statistical Fingerprints

In the previous section, we have employed Hearst's 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 have queried Google, Yahoo and MSN to

get the number of documents that contain it. The function 'count' models this query.

$$count : L \times P \times V \rightarrow \mathbb{N}$$

Thereby, $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{P}$ stand for the set of all entities to be classified, for the labels from a given ontology and for a set of pattern schema, respectively. Thus, count(l, p, v) returns the number of hits of pattern of the pattern schema p instantiated with the entity v and the label l. Further we define the sum over all the patterns conveying a certain relation r:

$$count_r(v, l) = \sum_{p \in P_r} count(l, p, v)$$

where P_r is the set of pattern schemas denoting a certain relation r. Now we formally define the statistical fingerprint of an entity v with respect to a relation r and a set of labels L:

$$SF(v, r, L) := \{(l, n) | l \in L \wedge n = count_r(v, l)\}$$

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.

$SF_{max}(v, r, L) := \{(l, n) | l := argmax_{l' \in L} count_r(v, l') \wedge n = count_r(v, l)\}$

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

$SF_m(v, r, L) := \{(l, n) L = \{l_1, l_2, \ldots, l_{|L|}\} \wedge$

$$count_r(v, l_1) \leq \ldots \leq count_r(v, l_{|L|}) \wedge$$

$$l \in \{l_1, \ldots, l_m\} \wedge n = count_r(v, l)\}$$

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

$SF_\theta(v, r, L) := \{(l, n) | count_r(v, l) \geq \theta \wedge n = count_r(v, l)\}$

Table 3.9: Sample anonymous datasets from Watch domain, source wristwatch.com

A_1	A_2	A_3	A_4	A_5
Armani	AR5447	Ladies	Stainless steel bracelet	$195.00
Seiko	SDWG32	Men	Stainless steel Butterfly clasp	$400.00
Longines	L51580966	Petite	Stainless steel bracelet	$1,700.00
Casio	DW5600E1V	Men	Plastic, black	$70.00

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 θ as follows:

$$SF_{m,\theta}(v, r, L) = SF_m(v, r, L) \cap SF_\theta(v, r, L)$$

We report here some typical Statistical Fingerprint (SF) computed for some attributes from Watch domain anonymous datasets as shown in table 3.9 (source: www.wristwatch.com). The corresponding Statistical Fingerprints (SF) are shown in figure 3.4 and 3.5. From

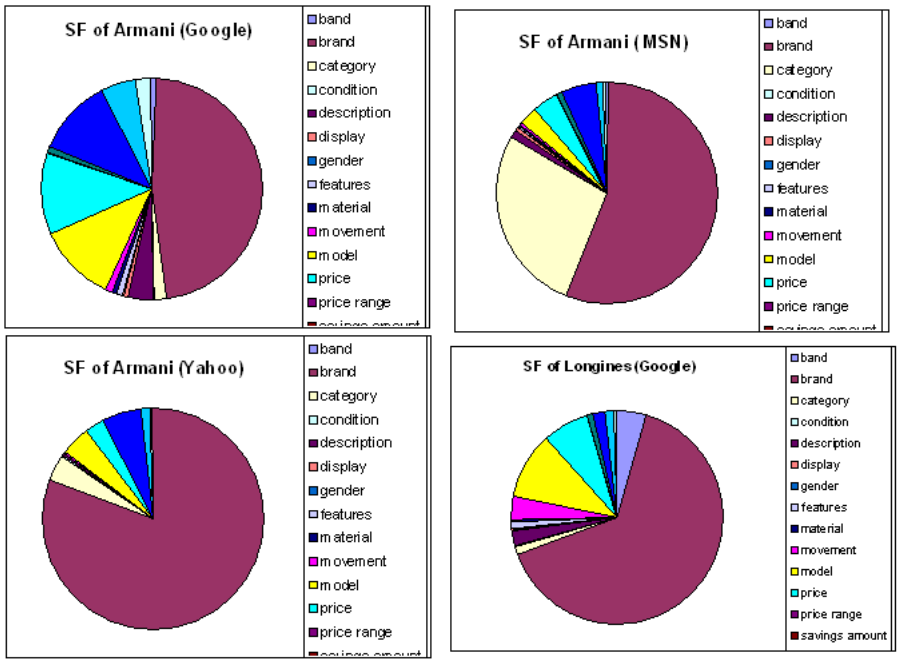

Figure 3.4: Statistical Fingerprints for Armani and Longines

Table 3.10: Hits count to annotate A_1 Label

Label	**Value**	**Google**	**Yahoo**	**MSN**
band	Armani	520	391	155
brand	*Armani*	*32200*	*490000*	*32000*
category	Armani	1140	24800	15700
condition	Armani	221	181	63
description	Armani	2510	1650	720
display	Armani	560	170	192
gender	Armani	22	33	25
features	Armani	642	642	259
material	Armani	208	245	121
movement	Armani	769	753	86
model	Armani	7920	27200	1710
price	Armani	8340	17500	2170
price range	Armani	7	11	4
savings amount	Armani	0	0	0
size	Armani	559	664	414
style	Armani	7820	34900	3050
title	Armani	3720	8250	478
type	Armani	1450	1850	363
band	Longines	1090	937	455
brand	*Longines*	*15200*	*77900*	*5620*
category	Longines	220	258	132
condition	Longines	116	94	74
description	Longines	500	458	178
display	Longines	78	53	63
gender	Longines	5	6	1
features	Longines	246	143	109
material	Longines	51	18	25
movement	Longines	866	1140	242
model	Longines	2400	1350	184
price	Longines	1700	1290	501
price range	Longines	1	0	0
savings amount	Longines	0	0	0
size	Longines	202	132	132
style	Longines	442	819	195
title	Longines	349	974	132
type	Longines	55	59	27

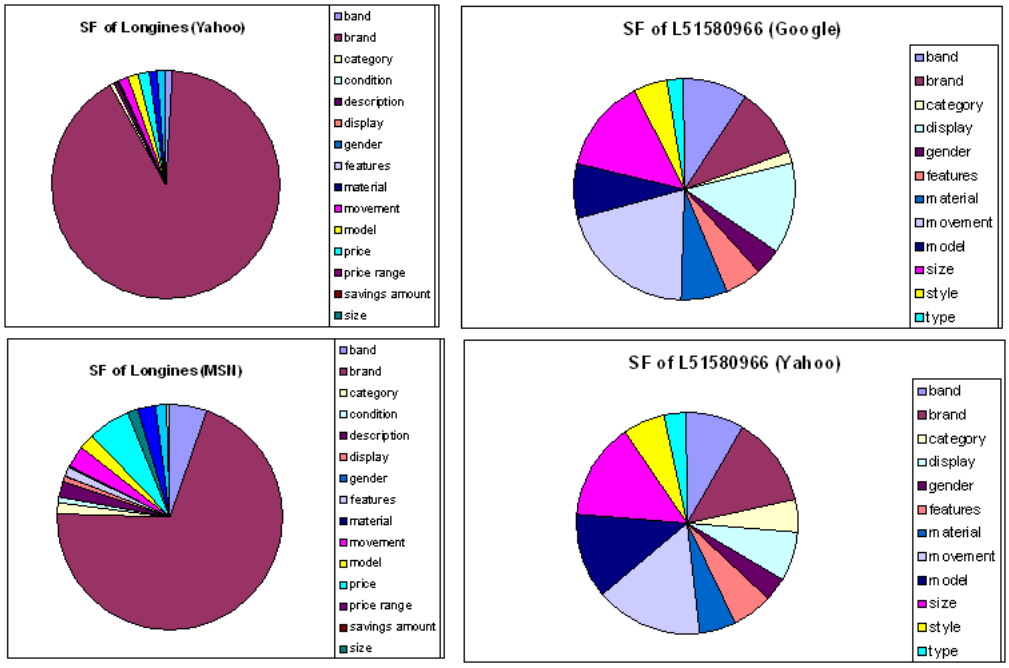

Figure 3.5: Statistical Fingerprints for Longines and L51580966

the table 3.10 data and statistical fingerprint, it is clear that *Armani* and *Longines* are instances of *brand*. Confusion arises when to annotate a less popular attribute value such as *L51580966*. L51580966 is a member of the anonymous attribute A_2 to which *AR5447* is also a member. We could not obtain any hits for *AR5447* from any of the Web search engine using the phrase approach. Therefore we resort to word approach to Web search engine (e.g., "band", "L51580966") for this less popular attribute values. We get the following results, shown in table 3.11. As can be seen from the statistical fingerprints, there are as many as 4 potential candidates to be the label for *L51580966. So only the highest 1st and 2nd count is not enough.*

3.5 Minimal Set of Patterns Required to Disambiguate Column Labeling

In this section, we will show that $P1$ pattern i.e., "L V" label followed by value pattern alone is enough to disambiguate column labeling. We notice that 10 patterns can be used

Table 3.11: Hits count to annotate A_2 Label, using word approach

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

to formulate speculative queries to Web search engines. Our experimental observation is that only one pattern, viz. "L V" $P1$ pattern is enough for column assignment. We call it *Discriminative power of Phrase Pattern* (P1). We are benefited by the *law of diminishing utility of pattern*. We will prove the observation by experimental results. Before going in detail, out of 10 possible patterns, we choose only 5 of them as follows:

L V (P1)

L such as V (H1)

such L as V (H2)

L including V (H3)

L specially V (H3')

We have applied the above 5 patterns in Music domain datasets as shown in table 3.6 with a candidate set of labels { artist, title, album }. We have used array of patterns consistently as follows:

$P1$ (1)

$P1 + H1$ (2)

$P1 + H1 + H2$ (3)

$P1 + H1 + H2 + H3$ (4)

$P1 + H1 + H2 + H3 + H3'$ (5)

Table 3.12 shows the hit counts that we have got from Google. From the table 3.12 data, we compute the probabilities, defined as LAA, the values are shown in tables 3.13 and 3.14.

From the probabilities as shown in table 3.13 and 3.14, we plot the curve as shown in

Table 3.12: Experimental results for different patterns

Label	**Value**	5	4	3	2	1
artist	Miles Davis	13777	13777	13777	13656	13500
title	Miles Davis	3031	3031	3031	3031	3030
album	Miles Davis	7523	7523	7523	7523	7520
artist	John Coltrane	13310	13310	13308	13307	13300
title	John Coltrane	1680	1680	1680	1680	1680
album	John Coltrane	6750	6750	6750	6750	6750
artist	Kind of Blue	1060	1060	1060	1060	1060
title	Kind of Blue	523	523	523	523	523
album	Kind of Blue	14306	14306	14306	14306	14300
artist	A Love Supreme	6790	6790	6790	6790	6790
title	A Love Supreme	615	615	615	615	615
album	A Love Supreme	5680	5680	5680	5680	5680

Table 3.13: Probabilities for Anonymous Attribute A_1

No. of Pattern	P(A_1| artist)	P(A_1| title)	P(A_1| album)
1	0.5866	0.101	0.3116
2	0.588	0.1	0.3106
3	0.59	0.1	0.309
4	0.59	0.1	0.31
5	0.59	0.1	0.31

Table 3.14: Probabilities for Anonymous Attribute A_2

No. of Pattern	P(A_2| artist)	P(A_2| title)	P(A_2| album)
1	0.2928	0.0399	0.667
2	0.2928	0.0399	0.667
3	0.2928	0.0399	0.667
4	0.2928	0.0399	0.667
5	0.2928	0.0399	0.667

Figure 3.6 and 3.7 respectively. From the plot as shown in Figure 3.6 and 3.7, it is clear

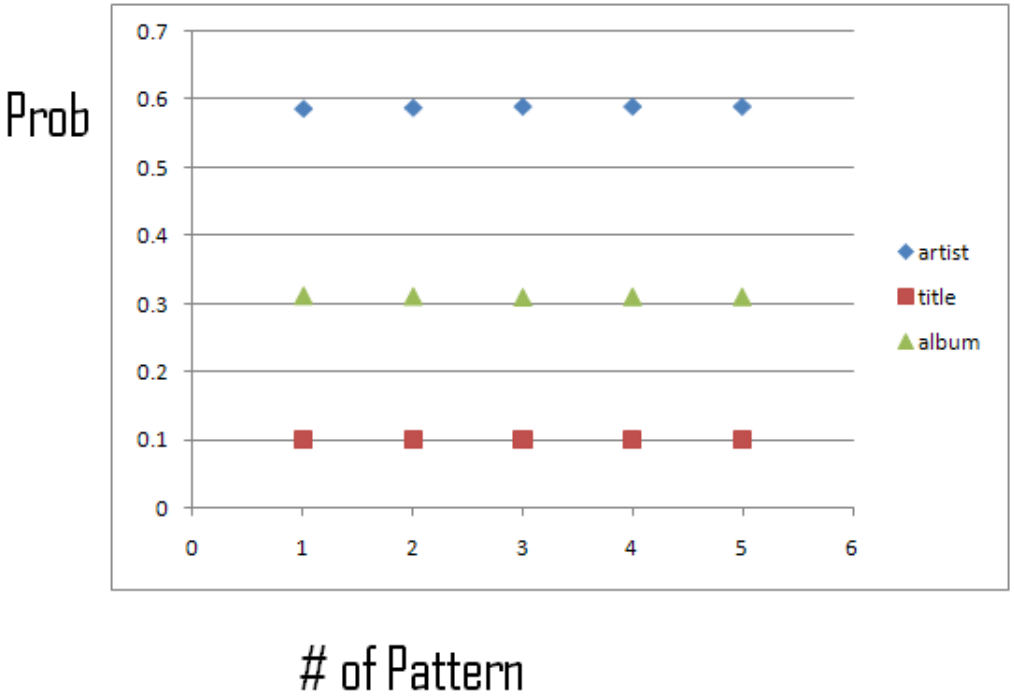

Figure 3.6: Probability of A_1 for different pattern

that only P1 pattern "L V" is enough to disambiguate column labeling. Mathematically, we formulate minimal pattern finding problem as follows: we have a set of patterns $\mathcal{P}$, a set of anonymous attributes $\mathcal{A}$ and a candidate set of labels $\mathcal{L}$. We write $\mathcal{P} \xrightarrow{L} \mathcal{A}$ to denote that $\mathcal{P}$ can determine $\mathcal{A}$ from a candidate set of labels $\mathcal{L}$. Then in each iteration, as we remove one $p_i \in \mathcal{P}$ and want to show that $\forall p \in \{\mathcal{P} - p_i\} \xrightarrow{L} \mathcal{A}$ should iteratively hold as long as p_i is redundant. The remaining last set of p_i is the minimal set of patterns. Surprisingly, the solution seems to be as finding key in RDBMS: reduce left hand side, reduce right hand side and reduce redundancy of Functional Dependency (FD) [60]. In our case, we only need to reduce left hand side (i.e., patterns).

The above graph (Figure 3.6 and 3.7) also demonstrates that *Hearst's pattern actually does NOT appear much in Web document exactly.* We have noticed a huge hits difference for queries with or without quote. It is our observation that *Hearst's pattern may appear in Web documents when we pose queries without quote.* But for our labeling context, we need to pose

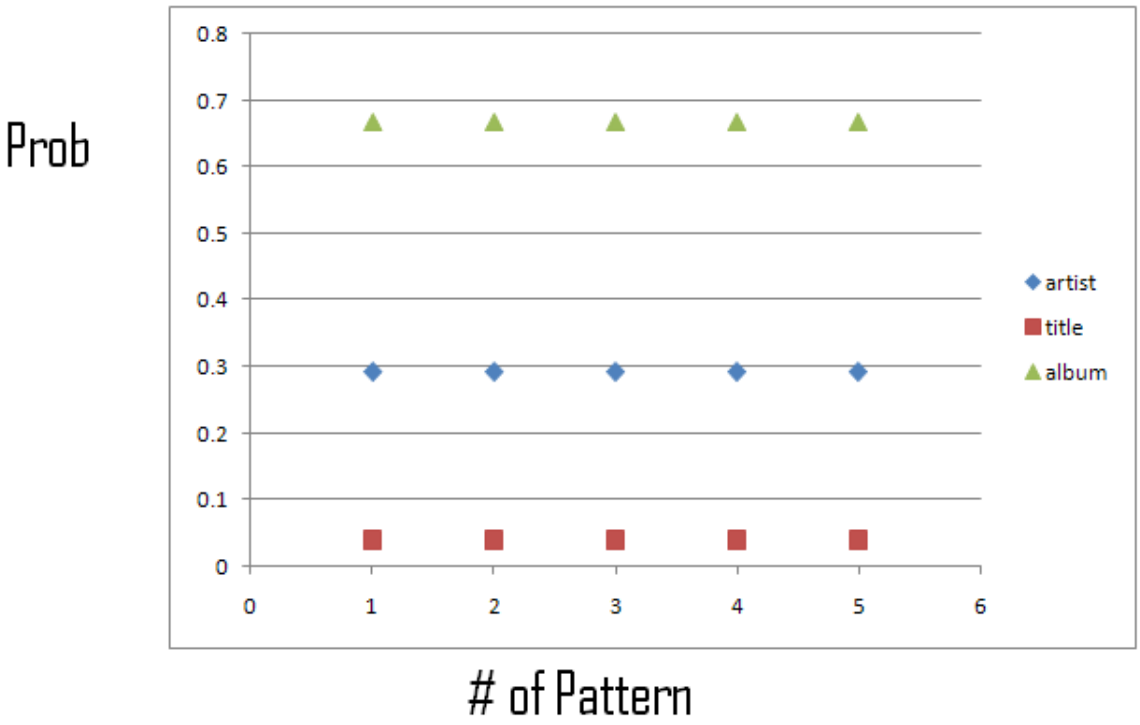

Figure 3.7: Probability of A_2 for different pattern

queries with quote for exact phrase matching, consider the following two queries: "`artist Miles Davis`" vs "`LA is 200 Miles away from California Davis`". If we query without quote, both the queries will produce hits for **Miles Davis**. Initially we thought to extend the Hearst pattern to take care of the other ontological relationship such as synonym, part-of etc. We were proposing the following *lexico-syntactic pattern* to complement Hearst pattern: Synonymous to; Part of; Belongs to; Also known as; A kind of. But as our experimental results do NOT yield sufficient hits, we give up the idea that the other ontological patterns will appear much in Web documents.

3.6 Proof by PMI

In the previous section, we have shown the feasibility that "L V" alone is enough to disambiguate column labeling. Now we give alternative proof for the same by the solid statistical method, PMI. Pointwise Mutual Information (PMI) is a measure of association based on information theory that compares the probability of observing two items together with the

probability of observing two items independently. PMI-IR was defined by Turney (2001) [65]. We have found that Web-based PMI statistics can be *effective* for disambiguating column labeling. We re-define PMI(L,V) into our labeling context as follows.

Definition 3.6.1 (PMI(L,V)) Pointwise Mutual Information between a label $\mathcal{L}$ and instance data value $\mathcal{V}$ denoted as PMI(L,V) is the number of hits for a query that combines the label $\mathcal{L}$ and instance data value $\mathcal{V}$ divided by the hits for the instance data value $\mathcal{V}$ alone.

$$PMI(L,V) = \frac{Hits(''L+V'')}{Hits(''V'')}$$

$\diamondsuit$

The '+' in the numerator denotes string concatenation, conform to URL encoded string in the Web search engine. The above definition can be viewed as the probability that "L+V" will be found on the Web page that contains V. We have applied PMI(L,V) in our labeling context to estimate which label $\mathcal{L}$ has the highest affinity with instance data value $\mathcal{V}$. The experiment was done with the anonymous datasets from music domain (with candidate set of labels: artist, title and album) as shown in Table 3.6 on the search engine *bing* (formerly MSN and live Search). The results are shown as follows:

Hits("Miles Davies") = 7450000

Hits("artist Miles Davis") = 2950000

Hits("title Miles Davis") = 274000

Hits("album Miles Davis") = 1220000

PMI("artist", "Miles Davis") = **0.3959**

PMI("title", "Miles Davis") = 0.0367

PMI("album", "Miles Davis") = 0.163

Conclusion: Highest PMI observed for "Miles Davis" is with "artist", therefore we label "Miles Davis" to be *artist*.

Hits("Kind of Blue") = 1110000

Hits("artist Kind of Blue") = 12300

Hits("title Kind of Blue") = 9030

Hits("album Kind of Blue") = 152000

PMI("artist", "Kind of Blue") = 0.011

PMI("title", "Kind of Blue") = 0.0081

PMI("album", "Kind of Blue") = **0.1369**

Conclusion: Highest PMI observed for "Kind of Blue" is with "album", therefore we label "Kind of Blue" to be *album*. Clearly PMI(L, V) $\neq$ PMI(V, L).

3.7 Proof from Probabilistic Model of Weakly Annotated Data

The previous two sections were dedicated for proving that "L V" pattern is sufficient to disambiguate column labeling. We make use of probability measure defined as Label Attribute Affinity (LAA) as well as proved by PMI(L,V). In this section, we will be discussing theoretical proof outline for the same, thanks to [33]. Hasan Davulcu et al. in their system of Baysian model for improving Weakly Annotated Data (WAD), defined *annotations* correspond to ontological role assignments such as *Concept, Attribute, Value* or *Noise*. Our labeling problem correspond to role assignment as *Attribute only*, i.e., relate label $\mathcal{L}$ with instance data value $\mathcal{V}$.

In this section, we will first present the notations used for the formal description of the model. We will formally define the problem of re-annotating the labels of a Web page and present the probabilistic algorithm. Next we will define the missing attribute inference problem and propose a solution. The notation used for formalization is given as follows:

- The set of all labels in the domain is denoted as $\mathcal{L}$.
- The **ontological roles** $\mathcal{R}$ is the set of *Concept, Attribute, Value* or *Noise*. Formally $\mathcal{R} = \{C, A, V, N\}$.
- A **term** is a pair $\langle l, r \rangle$ composed of a label l and a role $r \in \mathcal{R}$. In other words, terms are tagged labels in the Web pages. Each label in a web page is assumed to be tagged with only one of the given ontological roles above.

- In this setting, we consider all the labels in each Web page are tagged with roles, hence we define **Web page** to be vector of its terms. Formally, assuming m labels in the Web page $\mathcal{W}$; $\mathcal{W} = \{\langle l_1, r_1\rangle, \langle l_2, r_2\rangle, \ldots, \langle l_m, r_m\rangle\}$.

- The relational graph $\mathcal{G}$ is a weighted undirected graph where the nodes are the terms in the domain, and the weights on the edges represent the *association strength* between the terms.

- In our framework, the **context** of a label $l \in \mathcal{L}$ in a Web page $\mathcal{W}$ is the web page $\mathcal{W}$ itself.

The nodes $\mathcal{G}$ denote the labels with their ontological roles and the edges denote the association strengths between the annotated labels. Node weights are initialized as the counts of the corresponding terms and the edge weights are the counts of the corresponding edges in the document collection. Formally, w_{ij} which is the weight between the terms i and j is initialized as the number of times the edge (i, j) appeared in the entire domain. Similarly, w_i represents the weight of the node i and initialized as the occurrence of the corresponding term in the domain, i.e., term count. Note that the edges are undirected since association strength between labels is a bi-directional measure.

3.7.1 Label Role Inference

The role of a label depends on its *context*. This context of a label is intuitively defined to be its own Web page. The problem of role assignment for each label can now be formally defined as follows.

Definition 3.7.1 (Role Assignment) Given a Web page $\mathcal{W}$, the probability of a term $\langle l, r\rangle$ where $l \in \mathcal{L}$ and $r \in \mathcal{R}$ is $P(\langle l, r\rangle | \mathcal{W})$. $\diamondsuit$

This corresponds to the probability of the classification of l as r to be correct. Then, the role with the maximum probability will be the role assignment for the particular label l that is,

$$argmax_r P(\langle l, r\rangle | \mathcal{W})$$

For simplicity we use the *naive assumption* which states that,

Assumption 1. *All the terms in $\mathcal{G}$ are independent from each other but the given term $\langle l, r\rangle$*

During the role assignment probability calculation of a term, since we would like to utilize only the label's context we also assume,

Assumption 2. *The prior probabilities of all the roles of a label l are uniform.*

Now with the above assumptions, we can state the following theorem.

Theorem 3.7.2 *Let $\mathcal{W} = \{t_1, t_2, \ldots, t_m\}$. Then,*

$$argmax_r P(\langle l, r\rangle \,|\mathcal{W}) = argmax_r \prod_{i=1}^{m} P(\langle l, r\rangle \,|t_i)$$

Proof: Let $t = \langle l, r\rangle$. By Bayes's rule,

$$P(t|\mathcal{W}) = P(t|t_1, t_2, \ldots, t_m) = \frac{P(t_1, t_2, \ldots, t_m|t)P(t)}{P(t_1, t_2, \ldots, t_m)}$$

Using the independence assumption,

$$= \frac{\prod_{i=1}^{m} P(t_i|t)P(t)}{\prod_{i=1}^{m} P(t_i)}$$

Again using Bayes's rule,

$$\frac{\prod_{i=1}^{m} P(t|t_i)P(t_i)}{P(t)^m} \cdot \frac{P(t)}{\prod_{i=1}^{m} P(t_i)} = \frac{\prod_{i=1}^{m} P(t|t_i)}{P(t)^{m-1}}$$

By assumption 2, $P(t)^{m-1}$ will be constant. That is,

$$argmax_r P(t|\mathcal{W}) = argmax_r \prod_{i=1}^{m} P(t|t_i).$$

□

3.7.2 Missing Attribute Inference

In WAD, most of the attribute labels are missing, especially in the non-technical domains and non-template driven Web sites. Discovering missing relations is one of the crucial tasks during automated meta-data extraction. Our probabilistic model can also be tailored to infer some of the missing attributes.

Suppose two related entities have a missing attribute in a Web page. The first entity may be either a concept or an instance of a concept whereas the second one may be a value or a set of values.

Definition 3.7.3 (Missing Attribute Inference) Given two related entities e_1 and e_2 in a Web page, the probability of a label $l \in \mathcal{L}$ to be the attribute between them is $P(\langle l, \mathcal{A} \rangle | \mathcal{S})$ where $\mathcal{S} = e_1 \cup e_2$. $\diamondsuit$

Thus, the missing attribute can be inferred by the following formula,

$$argmax_{l \in \mathcal{L}} P(\langle l, \mathcal{A} \rangle | \mathcal{S})$$

And, with the same assumptions described above,

Theorem 3.7.4 *Let e_1 and e_2 be two entities and $\mathcal{S} = e_1 \cup e_2$. Then,*

$$argmax_{l \in \mathcal{L}} P(\langle l, \mathcal{A} \rangle | \mathcal{S}) = argmax_{l \in \mathcal{L}} \prod_{t \in \mathcal{S}} P(\langle l, \mathcal{A} \rangle | t)$$

Proof: Follows from the same methodology in the proof of the previous Theorem. $\square$

3.8 Complexity Analysis of LADS

$m = Candidate\ set\ of\ labels,\ L_1 \ldots L_m$

$n = Number\ of\ anonymous\ attributes,\ A_1 \ldots A_n$

$k = Number\ of\ patterns,\ p_1 \ldots p_k$

$r = Number\ of\ random\ tuples\ t_1 \ldots t_r$

The complexity of LADS is $\theta(mnkr)$, which can be reduced to linear time in the order of

supplied candidate set of labels, m. The details are as follows: the number of pattern k can be at most 10. We have proved that "L V" i.e., label followed by value pattern alone is sufficient to disambiguate column labeling. So k is small constant, then the complexity reduces to $\theta(mnr)$. Our experiment shows that the number of random tuples required for LADS to correctly assign label is only $3, \ldots, 9$. So r can be replaced with a small constant. Then the complexity becomes $\mathcal{O}(mn)$. Again, according to literature review (ref BAMM dataset at UIUC, http://metaquerier.cs.uiuc.edu/repository/datasets/bamm), the number of anonymous attributes in a Web table is in the range of $6, \ldots, 10$. So n can be thought of as small constant, then the complexity of LADS reduces to $\mathcal{O}(m)$, i.e., the complexity of our algorithm LADS is linear in the number of supplied candidate set of labels, m. Our experimental results show that it takes only a couple of minutes to holistically label all the DSC columns of a reasonable size of real life Web table. The cost is actually the number of queries submitted to Web search engine. The cost represents the network latency for all the queries. From the above analysis, maximum number of queries submitted to Web search engines is $10 * 9 * m * 10 = 900m$. This figure even reduces to $90m$ when only "L V" pattern is used. This complexity measure is a *quantitative proof* which shows why our work is better. We also note that there are two variants of LADS: Naive LADS (NLADS) and Greedy LADS (GLADS) (for which once a label is assigned, it is no more considered as a candidate for subsequent anonymous attributes). The complexity of NLADS is $\mathcal{O}(mn)$, whereas the cost of operation of GLADS is $mn - \frac{n(n-1)}{2}$, yields a performance improvement of $\frac{n(n-1)}{2}$, this is possible because of our disjoint set column (DSC) assumption.

3.9 Proof of Lemma

In this section, we present two lemma pertinent to our column labeling strategy.

Lemma 3.9.1 *Given a candidate set of labels $\{l_1, l_2, \ldots, l_m\}$ and a disjoint set of anonymous attributes $\{A_1, A_2, \ldots, A_n\}$ where $m > n$, the label assignment algorithm* LADS *has the Greedy Choice Property [23].*

Lemma 3.9.2 *Algorithm* LADS *is not only greedy but also optimal such that the order of choosing anonymous attributes to be labeled in each iteration does not matter.*

Table 3.15: Greedy Labeling

Domain	**Candidate set of labels**	**No. of attribute**	**Matches**	**Mismatches**
Movie	Actor, Director, Genre, Rating, Title, Film	4	3	1
Music	Artist, Title, Album	2	2	0
Political	President, Governor, Senator, Vice President	3	2	1
Synthetic	Toy, Furniture, Electronics, Clothing	3	3	0
Watch	Band, Brand, Gender, Display, Model, Category	4	3	1
Automobile	Model, Color, Make, State, Type, Class , Style, Category	4	3	1

In the following we will prove the above two lemma by experimental observation on anonymous datasets from a number of domains. Before proving the above two lemma, we wish to digress a little bit about the operational modalities. In order to prove the above two lemma by experimental observation, we will have a candidate set of labels, m, $\{l_1, l_2, \ldots, l_m\}$ and anonymous datasets n, $\{A_1, A_2, \ldots, A_n\}$ and we need that $m > n$, i.e., we have more labels than to assign to anonymous attributes. Greedy label assignment can take place in two different ways: (i) choose a label $l_i, 1 \leq i \leq m$, then assign the label l_i to one of the anonymous attribute $A_j, 1 \leq j \leq n$ (ii) choose an anonymous attribute $A_j, 1 \leq j \leq n$ and label the attribute with one $l_i, 1 \leq i \leq m$. Method (i) should NOT be followed because we may be exhausted with assigning labels for all the anonymous attributes, whereas more appropriate labels can be there to assign (as $m > n$). Method (ii) is appropriate and we choose to follow this method, in each iteration we want to label an anonymous attribute A_j with a label l_i which best describes the semantic meaning of the instance data values correspond to A_j. But once we assign label l_i for some anonymous attribute A_j, we no longer consider the label l_i for the subsequent anonymous attributes A_j any more. Our two lemma will guarantee that by doing so, we are only speeding up the processing without losing any information.

Now we present table 3.15, showing a number of domains by which we have done our experiment to prove the above two lemma. The probabilistic labeling results for these domains are shown in the next chapter.

Our greedy labeling algorithm has produced the correct labels for all the above domains. Thus we prove lemma 1. We define the *correct labels* to be those which *matches* between human assigned labels to that of machine generated labels. Now we prove the second lemma. Note that given n anonymous attributes, there are $n!$ possible permutations of the anonymous attributes. Therefore the worst case complexity to prove lemma 2 is $\mathcal{O}(n!)$. However it is sufficient to prove the second lemma by considering two orders only: *forward* and *backward*. Therefore we choose to run our algorithm, LADS, using *forward* and *backward* orders only. Forward order corresponds to choosing the anonymous attributes starting from $1, 2, \ldots, n$, whereas backward order corresponds to do the reverse, $n, n-1, \ldots, 2, 1$. For both the cases, we have found that our algorithm, LADS, has produced the same set of labels for the anonymous attributes. Thus we prove lemma 2. Now we present the following Theorem.

Theorem 3.9.3 *Algorithm* LADS *is guaranteed to work for DSC column.*

Proof First we show that our algorithm, LADS, is able to assign labels for all the anonymous datasets. This step of the proof is trivial. Given a candidate set of m labels $l_1, l_2, \ldots, l_m$ and an anonymous datasets n, $A_1, A_2, \ldots, A_n$, where $m > n$ the algorithm, LADS, terminates in time complexity of $900m$ with assigning label for all the anonymous attributes $A_1, A_2, \ldots, A_n$. Now we make use of our disjoint set column assumption and using Lemma 1 and 2, we prove the above Theorem. (***proved***)

Note that we have proved phrase pattern "L V" i.e., label followed by value pattern alone provides *very good* result to disambiguate column labeling. However, there are failure cases as well, *"L V" pattern alone sometimes can't disambiguate column labeling when labels themselves are confusing.* For those cases, we may augment "L V" pattern with additional *Hearst pattern* in a bid to disambiguate column labeling. In this section we present one such case as follows. For a candidate set of labels $\{title, director, actor, genre\}$ and anonymous datasets $\{''Vertigo'','' A.Hitchcock'','' J.Stewart'','' Thriller''\}$,
"Thriller" was assigned *title* to be the label, whereas its correct label is *genre*. Mathematically, let $\{l_1, l_2\}$ be two confusing labels for an instance data value V. Let $l_1\ V \rightarrow N_1$ and $l_2\ V \rightarrow N_2$ and $N_1 \approx N_2$ (similar count, so can't decide which label l_1 or l_2 is more appropriate for V).

Then let $p_1, p_2, \ldots, p_k$ be a set of *Hearst patterns*. Then we have $\sum_{i=1}^{k} l_1 p_i V \rightarrow N_1'$ and $\sum_{i=1}^{k} l_2 p_i V \rightarrow N_2'$. Then there will be three possible outcomes: $N_1' > N_2'$, $N_1' < N_2'$ and $N_1' \approx N_2'$. For the first two cases, introduction of additional *Hearst pattern* yields positive result to disambiguate column labeling. For the third case, after introducing *Hearst pattern*, we still can't disambiguate which label l_1 or l_2 is more appropriate for V. Then we can disambiguate label for V based on *majority* of the Hearst patterns that yield hits for V. Example: already mentioned, for a candidate set of confusing labels $\{title, genre\}$ and instance data value $\{Thriller\}$, we could disambiguate $\{Thriller\}$ by utilizing third case to be *genre*. We have another example for the above from Political Domain. For instance values such as $\{Republican, Democrat\}$ and a candidate set of labels $\{VicePresident,\ Party\}$, the instance data values were incorrectly labeled as "Vice President" when we only used "L V" pattern alone. Then we introduced additional *Hearst pattern* and could disambiguate (by using the first two cases of our argument) the data value instances of $\{Republican, Democrat\}$ to be "Party".

We also note that our algorithm, LADS, can only label numeric data as NUM. Finding exhaustive type of numeric pattern itself is a fundamental research problem of its own. For example it is easy for regular expression based wrapper to label a value like 123-456-7890 as phone number, but it is extremely difficult to know whether the number is home-phone or office-phone. We did NOT find much work which aims at labeling numeric data.

3.10 Error Estimation Due to Model Violation

So far we have assumed that the columns are pairwise disjoint, there will be no repeating value instances in multiple columns. We have shown in our subjective evaluation that under such assumptions, the performance of LADS is very good. However, in many real life scenarios, it is possible to have multiple columns where some value instances will repeat. In this section, we formally define such a problem and at the end we will argue that the performance of our algorithm, LADS, in general, and our probability measure in particular, are able to handle the unexpected situations as well.

The question we are trying to answer in this section can be formulated as follows: what is the effect of a single value instance v which appears in multiple columns? Mathematically,

$\exists v | v \in A_i$ *and* $v \in A_j$ *for some*(i, j) where A_i is of domain D_i and A_j is of domain D_j and $D_i \cap D_j \neq \phi$. Example: suppose we have a set of values $\{Hotmail,\ Gmail,\ Eudora,\ Pine\}$ and another set of values $\{Oak,\ Apple,\ Blueberry,\ Pine\}$ and we have to label the datasets with two candidate set of labels $\{Email\ Software,\ Tree\}$. Note that *Pine* appears in both the columns. In the first datasets, *Pine* is Email Software, whereas in the second datasets, *Pine* represent Tree. We have got more hits for "Tree Pine" (55,800) than "Email Software Pine" (13,100). But this result has a negligible effect on labeling $\{Hotmail,\ Gmail,\ Eudora,\ Pine\}$, the datasets will be labeled as "Email Software." *General intuition for solution for this case:* the other homogeneous set of values in the respective column will *compensate* for the violating values' effect.

Our naive assumption for the algorithm, LADS, was that "L V" will co-occur in Web documents as one-to-one mapping, that is one label produces hits with one data value instance. One label produces hits with multiple homogeneous data value instances as well. In real life application, we find the relationship between label and value to be $M : N$, many-to-many. We decompose $M : N$ relationship into two relationship $M : 1$ and $1 : N$. For the $M : 1$ relationship, we have multiple labels that can produce almost equal hits for a single data value instance. Those multiple labels happen to be synonymous. In the Figure 3.5 of statistical fingerprint from watch domain, we have shown that there can be as many as 4 potential labels for some anonymous attributes. We have also shown another example earlier, two labels $\{title, genre\}$ produce almost equal hits for the same instance data value $\{Thriller\}$. Errors are introduced because of the confusing labels, $\{title, genre\}$. In the earlier section, we have shown how to handle that type of error. In this section, we will present one more unusual labeling use case where a single value instance produces almost equal hits with multiple labels. Example from Political Domain, v = George H. W. Bush, D_i is "President" and D_j is "Vice President". The labels "President" and "Vice President" are NOT synonymous (VP is subclass of President), whereas a single data value instance such as George H. W. Bush is very confusing to be either "President" or "Vice President." We treat a single data value instance as *violating* if the value instance may appear in multiple columns. We will estimate the labeling error produced under such event and will show that as the number of *violating*

value instance increases, the labeling accuracy decreases still the performance of our labeling method is acceptable.

In this section we wish to quantify the error introduced by labeling algorithm in general and our probabilistic measure in particular. We will demonstrate how the error behaves, what is the effect of degree of repeat violation in column labeling. For n anonymous attributes, maximum possible repeats will be in $n-1$ columns, and there are $\frac{n(n-1)}{2}$ possible permutations of pairs of columns. So some value instance repeat in a pair of columns can have an adverse effect on the labeling performance. Here we present our experimental observation from Political Domain. The experiment was performed during January 2010 using the search engine, Google. We have performed our experiment with random number of tuples 1, 3, 5, 7, 9, 11 and 13 on the anonymous datasets from Political Domain with a candidate set of labels $\{President,\ VicePresident,\ Party,\ Governor,\ Senator\}$

We choose to vary the violating tuples to be $0, 1, \ldots, 6$. Where 0 means there is no repeating value instance in multiple columns, an ideal case and for that case our algorithm LADS produces the best possible result. We ran the algorithm seven times, then a total of 14 labels were assigned. The best result was when all the 14 labels were correctly assigned by LADS as well as by our probability measure. Due to *violating tuples*, errors are introduced and the number of correctly labeled attributes will be less than 14. We plot the performance as number of violating tuples $0, 1, \ldots, 6$ vs number of correctly assigned labels $5, \ldots, 8$. To prove our intuition, we provide the following Figure 3.8. As can be seen from the plot Figure 3.8, the number of violating tuples have negligible effect on the labeling performance. Thus we prove our intuition that the other homogeneous set of values in the respective column will compensate for the violating values' effect. The plot also reveals that our decision based on hit counts measure is almost comparable to probabilistic measure, whereas probabilistic measure always outperform hit counts measure.

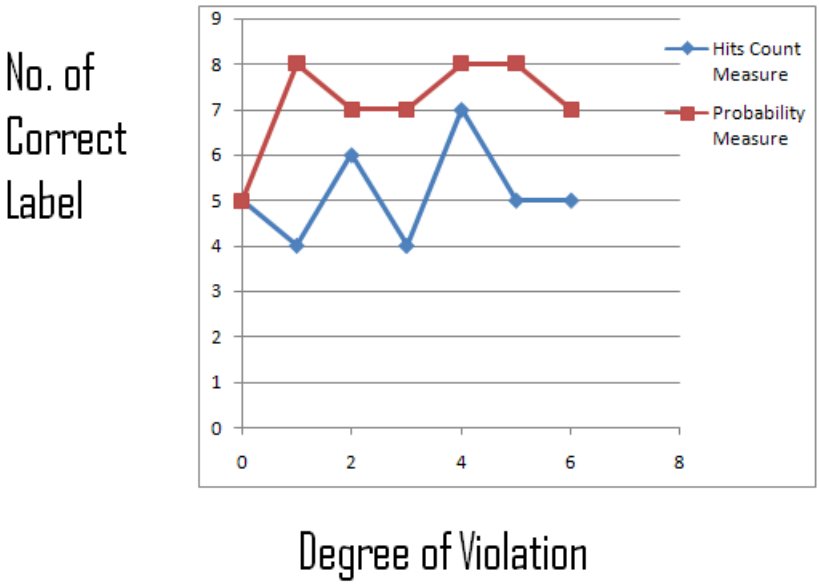

Figure 3.8: Degree of Violation vs No. of correct label

CHAPTER 4

RESULTS, DISCUSSIONS AND IMPLEMENTATION

4.1 Sample Result of Labeling

Here we provide some results of our experiment from a number of domains. The correct results are highlighted in the tables. The figures show that there are sufficient gaps between 1st and 2nd highest probabilities for each of the anonymous attributes. Therefore it will make it easier for the knowledge engineer to take final decision.

4.2 Correction Factor for Annotation using Algorithm LADS

Suppose we have two competing labels l_i and l_k for the anonymous attribute A_j. Assume that l_i is more appropriate for A_j than l_k. Then labeling error will be produced only if the following situation happens: $\exists k \ |1 \leq k \leq m, k \neq i, \ LAA(A_j, l_i) < LAA(A_j, l_k)$. For violating tuples, the fix will be to remove the tuples from the datasets and re-run LADS for the final annotation. Note that we are NOT permanently removing the tuples from datasets, only removing temporarily for the sake of annotation. Mathematically, for an anonymous attribute A_j, there exist two labels l_i and l_k and both the labels are equally likely for A_j, i.e., $LAA(A_j, l_i) \cong LAA(A_j, l_k)$. The same condition holds for another anonymous attribute A_h, i.e., $LAA(A_h, l_i) \cong LAA(A_h, l_k)$, where A_j is of domain D_j and A_h is of domain D_h and $D_j \cap D_h \neq \phi$. We consider two labels l_i and l_k are equally likely for some anonymous attribute A_j only if $|LAA(A_j, l_i) - LAA(A_j, l_k)| < \tau$, where τ is some threshold set by the user.

Table 4.1: Anonymous Datasets from Synthetic Domain

Candidate Set of Labels: Toy, Furniture, Electronics, Clothing, Footwear			
A_1	A_2	A_3	A_4
chair	baseball	pants	Boots
desk	doll	coat	Running shoes
bed	teddy bear	hat	Sandals
couch	skipping rope	dress	Dress shoes
table	jigsaw puzzle	scarf	Slippers

Table 4.2: Probabilistic Labeling on Synthetic Domain

Probability	**Value**
P(A_1|Toy)	0.18680023277160368
P(A_1|Furniture)	**0.7145956260276547**
P(A_1|Electronics)	0.029082023998840772
P(A_1|Clothing)	0.05282508377170027
P(A_1|Footwear)	0.016697033430200604
P(A_2|Toy)	**0.7856416057854673**
P(A_2|Furniture)	0.0495453859701936
P(A_2|Electronics)	0.007439431509867136
P(A_2|Clothing)	0.11982097602779276
P(A_2|Footwear)	0.03755260070667921
P(A_3|Toy)	0.06915115539539118
P(A_3|Furniture)	0.14361584940104846
P(A_3|Electronics)	0.06365242335290656
P(A_3|Clothing)	**0.6541838859981988**
P(A_3|Footwear)	0.06939668585245508
P(A_4|Toy)	0.058395838744636976
P(A_4|Furniture)	0.16843331931519773
P(A_4|Electronics)	0.012452289314594783
P(A_4|Clothing)	0.21077426646678044
P(A_4|Footwear)	**0.5499442861587901**

Table 4.3: Anonymous Datasets from Movie Domain

Candidate Set of Labels: actor, director, genre, rating, title, film			
A_1	A_2	A_3	A_4
Romance	5 stars	Doctor Zhivago	David Lean
Comedy	5 stars	Modern Times	Charles Chaplin
Epic	4 stars	Spartacus	Stanley Kubrick

Table 4.4: Probabilistic Labeling on Movie Domain

Probability	**Value**
P(A_1|actor)	0.01050362098284492
P(A_1|director)	0.024025618639729986
P(A_1|genre)	**0.4160762807492963**
P(A_1|rating)	0.009746013437688435
P(A_1|title)	0.47537845109174676
P(A_1|film)	0.06427001509869361
P(A_2|actor)	1.6276724951379707E-5
P(A_2|director)	1.1554376278477301E-4
P(A_2|genre)	1.0010384920768609E-4
P(A_2|rating)	**0.9860672999798817**
P(A_2|title)	8.338102771607169E-4
P(A_2|film)	0.012866965406013666
P(A_3|actor)	0.05386384761704929
P(A_3|director)	0.02432337090485912
P(A_3|genre)	0.0014365165801628781
P(A_3|rating)	0.005617358860632456
P(A_3|title)	0.1660278023266915
P(A_3|film)	**0.7487311037106048**
P(A_4|actor)	0.1184600671962024
P(A_4|director)	**0.37285591165363713**
P(A_4|genre)	0.014373627723665558
P(A_4|rating)	6.056309142375138E-5
P(A_4|title)	0.020598287011500845
P(A_4|film)	0.47365154332357035

Table 4.5: Anonymous Datasets from Watch Domain

Candidate Set of Labels: band, brand, gender, display, model, category			
A_1	A_2	A_3	A_4
Armani	AR5447	Ladies	Stainless steel bracelet
Seiko	SDWG32	Men	Stainless steel Butterfly clasp
Longines	L51580966	Petite	Stainless steel bracelet
Casio	DW5600E1V	Men	Plastic, black

Table 4.6: Probabilistic Labeling on Watch Domain

Probability	**Value**
P(A_1|band)	0.17364618052554065
P(A_1|brand)	**0.4186672639444297**
P(A_1|gender)	0.0017503056861785285
P(A_1|display)	0.05852546045227429
P(A_1|model)	0.23341411772341383
P(A_1|category)	0.11399667166816302
P(A_2|band)	0.2784638422436793
P(A_2|brand)	0.23767171000712742
P(A_2|gender)	0.09145032406886205
P(A_2|display)	0.17757755008800985
P(A_2|model)	**0.12151625632163594**
P(A_2|category)	0.0933203172706854
P(A_3|band)	0.18116412291464817
P(A_3|brand)	0.21697937098025694
P(A_3|gender)	**0.21602793724269945**
P(A_3|display)	0.06032548775942292
P(A_3|model)	0.20017775509127134
P(A_3|category)	0.12532532601170118
P(A_4|band)	**0.4303055531054701**
P(A_4|brand)	0.042720662092972865
P(A_4|gender)	0.042549810484077896
P(A_4|display)	0.10156887357786115
P(A_4|model)	0.12163063682956912
P(A_4|category)	0.26122446391004894

Table 4.7: Anonymous Datasets from Automobile Domain

Candidate Set of Labels: Model, Color, Make, State, Type, Class, Style, Category			
A_1	A_2	A_3	A_4
Acura	Sedan	2001 TL	by Acura
Acura	Coupe	2001 CL	by Acura
Audi	Sedan	2001 A4	by Audi
Audi	Coupe	2001 TT	by Audi
Audi	Convertible	2001 TT	by Audi
Audi	Wagon	2001 A4	by Audi
Ford	Sedan	2001 focus	by Ford
Cadillac	Sedan	2001 De Ville	by Cadillac
Audi	Coupe	2001 Eldorado	by Cadillac
Chrysler	Sedan	2001 300M	by Chrysler
Volvo	Sedan	2001 S40	by Volvo
Volvo	Coupe	2001 C70	by Volvo
Volvo	Convertible	2001 C70	by Volvo
Volvo	Wagon	2001 V70	by Volvo
Mercedes	Sedan	2001 E-class	by Mercedes
Mercedes	Coupe	2001 CLK-Class	by Mercedes
Mercedes	Convertible	2001 CLK-Class	by Mercedes
Mercedes	Wagon	2001 E-Class	by Mercedes
Saab	Sedan	2001 9-5	by Saab
Saab	Convertible	2001 9-3	by Saab
Jeep	Sport Utility Vehicle	2001 Grand Cherokee	by Jeep

Table 4.8: Probabilistic Labeling on Automobile Domain

Probability	**Value**
P(A_1|Model)	0.17856458732839947
P(A_1|Color)	0.06229903512610037
P(A_1|Make)	**0.17301966183079753**
P(A_1|State)	0.03220097993005406
P(A_1|Type)	0.13558027868873046
P(A_1|Class)	0.1342228534908299
P(A_1|Style)	0.0664358156905886
P(A_1|Category)	0.21767678791449954
P(A_2|Model)	0.1169118291655958
P(A_2|Color)	0.012067562953282398
P(A_2|Make)	0.03859657887688751
P(A_2|State)	0.008962084802220398
P(A_2|Type)	**0.24473705125524475**
P(A_2|Class)	0.19361598493627272
P(A_2|Style)	0.15404105370251026
P(A_2|Category)	0.23106785430798615
P(A_3|Model)	0.06432092125807717
P(A_3|Color)	0.07894651770461009
P(A_3|Make)	0.2040282821846578
P(A_3|State)	0.18950651079518155
P(A_3|Type)	**0.2679302527044949**
P(A_3|Class)	0.10559359949792047
P(A_3|Style)	0.06864669955773
P(A_3|Category)	0.021027216297328163
P(A_4|Model)	**0.5152461106473416**
P(A_4|Color)	2.0031245885784426E-4
P(A_4|Make)	0.003095548756520256
P(A_4|State)	0.11679784125409677
P(A_4|Type)	0.0468075074606059
P(A_4|Class)	0.0742606665372805
P(A_4|Style)	0.006033999264181826
P(A_4|Category)	0.23755801362111534

Table 4.9: Anonymous Datasets from Political Domain

Candidate Set of Labels: President, Governor, Senator, Vice President	
A_1	A_2
Barack Obama	Joseph Biden
George W. Bush	Richard Cheney
Bill Clinton	Al Gore
Jimmy Carter	Walter Mondale
Herbert C. Hoover	Charles Curtis
Woodrow Wilson	Thomas Marshall
William Taft	James Sherman
Grover Cleveland	Adlai Stevenson
Benjamin Harrison	Levi Morton
Grover Cleveland	Thomas Hendriks
James Garfield	Chester Arthur
Rutherford Hayes	William Wheeler
Ulysses S. Grant	Schuyler Colfax
James Buchanan	John Breckinridge
Franklin Pierce	William King
James K. Polk	George Dallas
Martin van Buren	Richard Johnson
Franklin Delano Roosevelt	Henry Wallace
William McKinley	Garret Hobart
George H. W. Bush	James Danforth Quayle
Ronald Wilson Reagan	George H. W. Bush
Gerald R. Ford	Nelson Rockefeller
Richard Milhous Nixon	Gerald R. Ford
Lyndon Johnson	Hubert Humphrey
John Fitzgerald Kennedy	Lyndon Johnson
Dwight David Eisenhower	Richard Milhous Nixon
Harry S. Truman	Alben Barkley
Franklin Delano Roosevelt	Harry S. Truman
Calvin Coolidge	Charles Dawes
Warren Harding	Calvin Coolidge

Table 4.10: Probabilistic Labeling on Political Domain

Probability	**Value**
P(A_1|President)	**0.5873377317403179**
P(A_1|Governor)	0.11672661319199679
P(A_1|Senator)	0.10532475574409776
P(A_1|Vice President)	0.19061089932358743
P(A_2|President)	0.45374480230286524
P(A_2|Governor)	0.17627956967075098
P(A_2|Senator)	0.07667668205290089
P(A_2|Vice President)	**0.2932989459734829**

4.3 The Overall Algorithm

Figure 3.8 shows that *probabilistic measure* will result in higher labeling accuracy than that of *hit counts measure*. Therefore we accommodate the probabilistic measure as well as additional Hearst pattern to compensate for the violating tuples effect. Here we briefly outline the overall labeling algorithm.

1. Accumulate hit counts for all L and a random subset of V

2. Compute LAA based on the hit counts

3. Annotate columns A_j with label l_i for which $LAA(A_j, l_i)$ is the highest and there is some label l_k for which $LAA(A_j, l_k)$ is the second highest and $|LAA(A_j, l_i) - LAA(A_j, l_k)| > \tau$

4. If some column A_j results in being labeled with a set of equally likely labels, introduce additional Hearst pattern to gather hit counts and choose the final label for the column A_j based on the criteria mentioned above

5. If a pair of columns result in being labeled with the same label due to violating tuples, remove those violating tuples from both the columns, then re-run LADS for final annotation

Here we present the complete algorithm LADSCOMPLETE which not only take care of DSC column, but also do a good job for RPS and NUM columns.

4.4 Experiment with Large Datasets

We have shown our approach based on small scale datasets and found the results to be favorable. Our data size wasn't that large in evaluation. In order to do experiments in large scale, one need special permission from search engine like Google. Otherwise Google may block the automated user query. That is one of the reasons why our results are based on small scale datasets. Information Extraction (IE) benchmarks in this area are not well developed. We have used the BAMM datasets from UIUC Web data integration repositories as well as from RoadRunner Datasets. BAMM datasets contains "query schemas" each of

Table 4.11: Anonymous Datasets from Music Domain

Artist, Title, Album, Band, Genre, Styles, Keyboard, Strings, Winds, Moods, Composer, Vocal, Percussion, Brass, Author, Rating, Label, Verse, Chorus							
A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
Miles Jaye	Aint That Good News	Kind of in Love	ABBA	Jazz	Euro Pop	Celeste	Bass
Miles B. Davis	Balm in Gilead	A Kind of Magic	AC-DC	Classical	Contemporary	Clavich	Cello
Myles C. Davis	By and By	A Kind of Bliss	Aerosmith	R&B	Pop/Rock	Harmonium	Vina
Dave Miles	Couldnt Hear Nobody Pray	Kind of Ballad	The Beatles	Pop/Rock	Rock	Harpsichord	Dulcimer
Jimmie Davis	De Gospel Train	Kind of Beast	Black Eyed Peas	Folk	Am Pop	Organ	Erhu
Xavier Rudd	Deep River	Kind of Better	Backstreet Boys	Latin		Piano	Guitar
Giles Davies	Jacobs Ladder	Kind of Miles	Bon Jovi	Blues		Keyboards	Sitar
Miles Davis	Down By the River	Kind of Smile	Bond	New Age		Electric Piano	Sarod
Lady Gaga	Down By the Riverside	The Kind of Place	Coldplay	Country		Accordion	Harp
David Myles	Every time I feel the Spirit	A Kind of Courage	Culture Club	Rap			Koto
David Keenan	Free at Last	Kind of Brother	Cypress Hill				Lute
David Myles	Git On Board Little Children	Kind of Silence	The Cranberries				Lyre
David Mills	Its Me	Kind of Girl	The Cheetah Girls				Banjo
David Myles	Great Day	A Kind of Solution	Dire Straits				Violin
Jesse Davis	I Couldnt Hear Nobody Pray	Kind of Like One	Def Leppard				Viola
Miles Baster	Ill Be There In The Morning	Kind of Like You	Destinys Child				Viol
Cile Davis		Kind of Lonesome	The Doors				
Danny Davis		The Kind of Love	Dixie Chicks				
Larry Davis		A Love Supreme	Disturbed				
Miles Visman		Supreme Love Gods	Deep Purple				
John Coates Jr.		Alif: Love Supreme	Evanescence				
Johnny Coles		Love Serenade	Escape the Fate				
John Conlee		Love Survives	Eagles				
John Coltrane		Love Snuck Up	Guns N Roses				
Johnny Costa		Love Scenes	Gorillaz				
John Constable		Kind of Blues	Hi 5				
Johnny Clarke		A Kind of Clue	Hollywood Undead				
Johnny Adams		A Kind of Cry Blues	II Divo				
Quantic		Kind of a Blur	Iron Maiden				
John Dowland		A Kind of Love In	Incubus				
John Cale		Kind of Summer	Inna				
Johnny Clegg		Kind of Gentle	Jonas Brothers				
John Lanchbery		A Kind of Suckle	Jagged Edge				
Johnnie Allan		A Kind of Closure	KoRn				
John Hollander		Kind of Rude	Limp Bizkit				
Jack Dangers		Kind of Like	Lynyrd Skynyrd				
Will Downing		Kind of Like a Bee	Metallica				
Johnny Carver		Kind of Brew	MC Magic				
Johnny Doran		Kind of Bill	Marilyn Manson				
John Fogerty		A Kind of Life	Mudvayne				
John Hartford		The Kindom of Love	Modern Talking				
Inna		Kind of Trouble	*NSYNC				
Jack Jones		Kind of Brittle	Oasis				
MC Magic		Kind of a Lie	OutKast				

which is a set of query attributes, extracted from Web query interfaces in 4 domains: Books, Automobiles, Movies, Music Records. Each domain consists of about 50 sources. We have shown an approach, Web Search Engine Based Annotator, to holistically assign column labels based on small scale datasets. In this section, we present results from music domain consist of 19 candidate set of labels and anonymous datasets of about 100 tuples. *Note that the datasets are simulated, meaning that a particular row of data do NOT represent a relation, although we assure that the data in a particular column is homogeneous.* Now we present more results of our experiment with large datasets collected from FIFA 2010 world cup tournament. There were 32 teams comprises of 32 coaches and 737 players. Note that *player* and *coach* represents possibility of DSC violation as those who are coaches, they were also player sometimes. As expected, the datasets of 32 coaches were labeled by our algorithm as 50% of chance being coach and 50% of chance being player. For the player datasets of 737 records, we randmonly

Table 4.12: Probabilistic Labeling on Music Domain

Label	$P(A_1)$	$P(A_2)$	$P(A_3)$	$P(A_4)$	$P(A_5)$	$P(A_6)$	$P(A_7)$	$P(A_8)$
Artist	0.1335	0.0284	0.0569	0.2233	0.0502	0.0096	0.0515	0.0515
Title	0.0160	0.0947	0.0239	0.0486	0.0500	0.0115	0.0547	0.0543
Album	0.1089	0.1581	0.0430	0.1291	0.0517	0.0468	0.0576	0.0576
Band	0.0353	0.0436	0.0483	0.2171	0.0593	0.0435	0.0647	0.0509
Genre	0.0380	0.0232	0.0271	0.0051	0.4465	0.4927	0.0286	0.0348
Styles	0.1009	0.1458	0.1350	0.0017	0.0385	0.0290	0.004	0.0047
Keyboard	0.0124	0.0570	0.0248	0.0182	0.0160	0.0013	0.1513	0.0742
Strings	0.0380	0.0152	0.0647	0.0500	0.0090	0.0055	0.1584	0.0896
Winds	0.1097	0.0389	0.1073	0.0616	0.0172	0.0695	0.0205	0.0427
Moods	0.0991	0.0717	0.0588	0.0650	0.0217	0.0110	0.0211	0.2199
Composer	0.0552	0.0394	0.0432	0.0132	0.0045	0.0069	0.0190	0.0124
Vocal	0.0208	0.0129	0.0254	0.0086	0.0860	0.0288	0.0837	0.0614
Percussion	0.0204	0.0419	0.0188	0.0373	0.0203	0.0067	0.0857	0.1554
Brass	0.0570	0.0236	0.0220	0.0086	0.0148	0.0434	0.0290	0.0181
Author	0.0128	0.0780	0.1318	0.0090	0.0362	0.0655	0.1142	0.0145
Rating	0.0215	0.0238	0.0567	0.0013	0.0126	0.0534	9.66E-4	0.0017
Label	0.0212	0.0630	0.0405	0.0186	0.0542	0.0682	0.0127	0.0234
Verse	0.0716	0.0168	0.0475	0.0587	0.0033	0.0032	7.40E-4	0.0236
Chorus	0.0268	0.0232	0.0231	0.0240	0.0069	0.0026	0.0401	0.0084

choose 1%, 2%, 5%,10%,15%, 20% and 25% records and run our algorithm with a candidate set of labels Player, Coach. The results are shown in the following Figure 4.1. It shows that randomly choosing 1% of records is good enough to correctly label anonymous datasets.

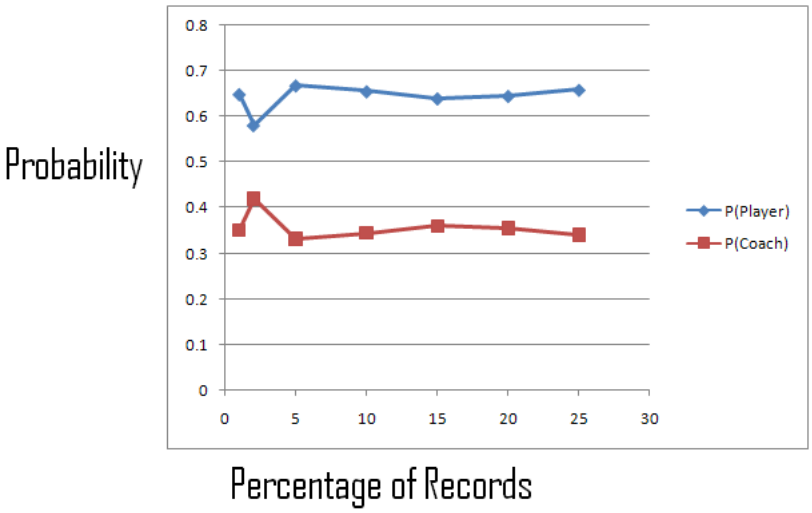

Figure 4.1: Probabilisitc Labeling on FIFA Datasets

4.5 Non-ideal Datasets and Impact of DSC Violation

Suppose we have table with 4 attributes (A_1, A_2, A_3, A_4), and 20 rows as shown in the following table. Lets choose the columns to be disjoint. The datasets are simulated, meaning a particular row do NOT represent a relation. We annotate them using LADS, results are as

Table 4.13: Anonymous Datasets from Movie Domain

Candidate Set of Labels: Actor, Director, Film, Movie, Genres, Types			
A_1	A_2	A_3	A_4
Kevin Costner	Roland Emmerich	All Quiet on the Western Front	Romance
Mel Gibson	Steven Spielberg	The Battle of Midway	Reality Show
Tom Cruise	Alfred Hitchcock	Back to the Future Part III	Spy Film
Omar Sharif	Stanley Kubrick	Terminator 2	Action
James Stewart	Roman Polanski	Casino Royale	Animation
Sean Connery	David Lean	The Longest Hundred Miles	Comedy
Robert De Niro	James Cameron	Shaft in Africa	Documentary
Leonardo DiCaprio	Tim Burton	Robin Hood	Drama
Tom Hanks	M. Night Shyamalan	The Patriot	Family
Eddie Murphy	Hayao Miyazaki	Jurassic Park	Science Fiction
Jim Carrey	Kevin Smith	The Sixth Sense	Horror
Arnold Schwarzenegger	John Ford	Chasing Amy	Suspense
John Travolta	Frank Capra	The Gold Rush	Western
Bruce Willis	Howard Hawks	The Godfather	Mystery
Brad Pitt	Cecil B. DeMille	Star Wars	Historical
Sylvester Stallone	William Wyler	Die Hard	War
Michael Douglas	George Cukor	Ben-Hur	Adventure
Pierce Brosnan	Woody Allen	Titanic	Culture and Society
Michael J. Fox	D.W. Griffith	Saving Private Ryan	Fantasy
Will Smith	Charles Chaplin	Vertigo	Sports Drama

Table 4.14: Probabilistic Labeling on Movie Domain

Label	$P(A_1)$	$P(A_2)$	$P(A_3)$	$P(A_4)$
Actor	0.5454	0.0366	0.0466	0.0499
Director	0.1499	0.7274	0.0281	0.0680
Film	0.1030	0.1156	0.2468	0.3947
Movie	0.1940	0.0547	0.3940	0.2075
Genres	2.27E-4	1.36E-4	0.1048	0.1572
Types	0.0073	0.0654	0.1794	0.1223

follows: Then let us copy 10% of column A_1 values into A_2. Annotate the table again. Then do it for 20%, 50%, 75% and 100%. The question is what should the annotator do? Obviously, as the overlap approaches 100% the annotation of A_1 and A_2 will approach identity, results as shown in the following table. Should the annotator catch this discrepancy and reduce the weights of the annotations for both A_1 and A_2 in some sound quantitative way to reflect that it cannot assign full faith in the annotation? How? Or should we just leave it to the user to figure it out. As can be seen from the table, when there is no repeat, the difference of

Table 4.15: Probabilistic Labeling of A_2 with Different % of Repeat from A_1

Probability	0%	10%	20%	50%	75%	100%
P(A_2|Actor)	0.0366	0.0930	0.1191	0.2179	0.2463	0.2973
P(A_2|Director)	0.7274	0.7001	0.6574	0.5585	0.4985	0.4548
P(A_2|Film)	0.1156	0.1059	0.1044	0.1066	0.1072	0.1081
P(A_2|Movie)	0.0547	0.0564	0.0722	0.0796	0.1157	0.1116
P(A_2|Genres)	1.36E-4	1.43E-4	1.36E-4	1.10E-4	1.19E-4	1.00E-4
P(A_2|Types)	0.0654	0.0442	0.0464	0.0371	0.0319	0.0279

probability between Director and Actor is 0.6908, for 10% repeat it is 0.6071, the difference monotonically decreases for 100% repeat as 0.1575. This implies that for this case a threshold value τ of 0.16 is good enough to take the final decision. If we assume that all the given m candidate set of labels are equally likely, then the threshold τ will be $\frac{1}{m}$. Our result supports this as well, for the above case we have 6 candidate set of labels and $\frac{1}{6} \approx 0.1575$.

Our approach seem to suggests latter, and basically saying LADS does NOT do it, but one can ignore it for small violations as has been shown in the previous chapter. What about the case when overlaps are present in all the columns? Then the problem becomes NP-hard problem. We only could figure out a model in this regard as a pair-wise column name annotation problem in section Correction Factor for Annotation using Algorithm LADS. Our future work will investigate more in this regard.

4.6 Implementation

We have used Google, Yahoo and MSN wrapper. We also make use of Google AJAX Search API. There are search engines which offer limited queries permitted per day, in that case we can make use of Yahoo! Search BOSS (Build your Own Search Service) API (it offers unlimited query per day). Here we show the screen shot of sample labeling on synthetic datasets:

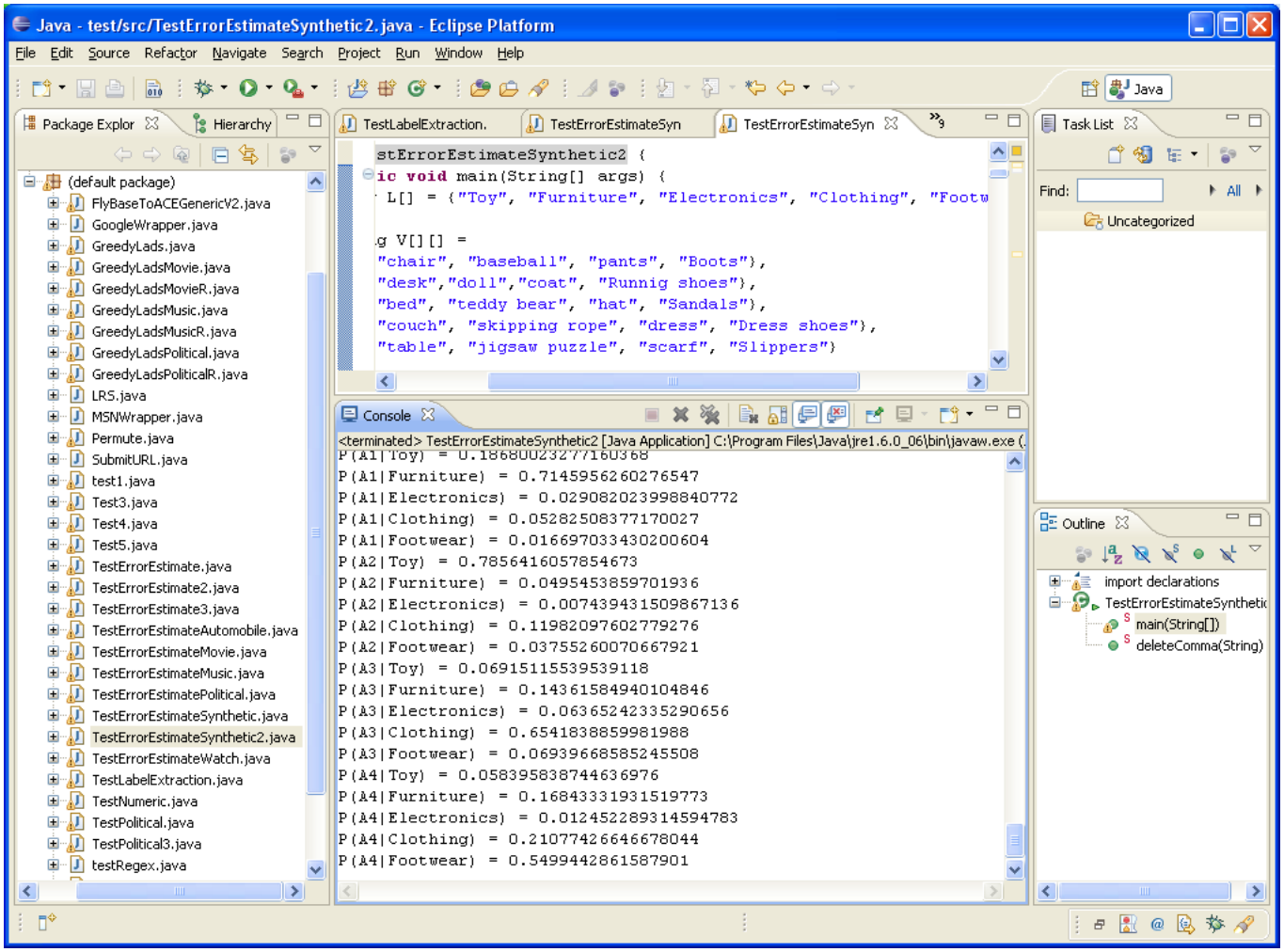

Figure 4.2: Screen shot of sample labeling on Synthetic Datasets

Algorithm 4 LADSComplete: Labeling Anonymous Datasets Complete

INPUT
$\mathbf{L}(1 \ldots m)$: *A set of labels*
$\mathbf{A}(1 \ldots n)$: *A set of anonymous attributes*
Precondition $m > n$
$\mathbf{P}(1 \ldots k)$: *A set of patterns*
$\mathbf{NP}(1 \ldots l)$: *A set of generic numeric patterns*
(*e.g* \$, *year*, $dd - mm - yyyy$)
$\mathbf{V}(1 \ldots t, 1 \ldots n)$: *Tabular Data Value*
s : *Odd Number of Seach Engine i.e* 1, 3, 5 *etc*
Search Engine Name : *e.g Google*, *Yahoo*, *MSN*
VARIABLE
$\mathbf{N}(1 \ldots s)$: *Search Engine hit counts*
$\mathbf{H}(1 \ldots n, 1 \ldots m, 1 \ldots s)$: $3D$ *array to store hit counts*
$R \rightarrow 1 \ldots t$ *Random indexes between* $1 \ldots t$ *generate one time*
Formulate Speculative query : $L \times P \times V \rightarrow N$
for $i = 1$ to n **do**
  Data Preprocessing
  determine column type to be DSC, *RPS*, *NUM*
  if *column* A_i *is of type NUM* **then**
    Label A_i *to be one from NP*
  else if *column* A_i *is of type RPS* **then**
    Remove RPS from A_i *and Label it with RPS*
  else
    $H[i][1..m][1..s] = 0$
    for *each* $r \in R$ **do**
      $iValue = V_{ri}$
      for $j = 1$ to m **do**
        $lValue = L[j]$
        for $p = 1$ to k **do**
          $qVar = lValue + P[p] + iValue$
          for $q = 1$ to s **do**
            $N[q] \leftarrow ExecQuery(qVar, SearchEngine_q)$
            $H[i][j][q] += N[q]$
          end for
        end for
      end for
    end for
    Let z be the majority index $(1..m)(1..s)$ *of highest*
    count in $H[i][1..m][1..s]$
    $A[i] = L[z]$
  end if
end for

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

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 research problem is worth studying: *automatically obtaining descriptions about a given Webbase's content, such as schema or semantics of the contained data*. Without a 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. Annotator based on Web Search Engine utilizes the information (*hit counts*) provided by popular web search engines such as Google, Yahoo and MSN. Search engine hit counts depend on the popularity of the label as well as the value. Most recent events have more hits than historic past events. We have got more hits for "President George W. Bush" (8,600,000) than for "President George Washington" (297,000). Query with or without quotes produces a different number of hit counts, but the decision of the algorithm never changes. Here in this research, we have reported our experimental results from about 10-15 domains which show that our approach is *effective* and *efficient*.

In this research we have studied the problem of automatic data annotation and described a *low-cost approach* for holistically labeling anonymous datasets. We have tested our approach on *small scale* datasets and found that the results are favorable. This method is meant to provide an incremental step toward the larger goals of Web Data Integration. Our approach, viz. Web Search Engine Based Annotator is complementary to the other work in relation to labeling anonymous datasets.

In this dissertation, we have researched the problem of labeling anonymous datasets. Our main focus was on a generic, correct and efficient solution to the problem of labeling anonymous datasets. First of all, we have formalized the problem as part of deep Web database

abstraction model. Finally, we have designed *efficient* labeling algorithm, LADS, LADSCOMPLETE. In summary, our main contributions are:

- We review the existing work on label assignment and propose to make it fully automatic
- We propose and define a novel label assignment problem
- We propose specific method, *Web search engine based annotator*, to solve the problem
- We run experiments to test the proposed methods and show that the proposed methods are *effective* and *efficient*
- We have shown that "L V", i.e., phrase pattern is enough to disambiguate column labeling. We have shown the feasibility of the pattern by experimental results as well as by solid statistical method PMI
- We make a procedural implementation of our approach in a Java framework

For future work, we plan to extensively evaluate the quality of the label assignment algorithm. In the following, we list some interesting research directions for future work:

- Label column name based on search engine 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 search engine "Camry is a." Extract the search engine snippets to mine label for Camry.
- Label column name based on combination of search engine hits and dictionary extraction.
- Label Column Name based on combination of general knowledge source and domain specific knowledge Source (e.g., CooksRecipe.com, bookfinder.com etc.).
- We have tested our hypothesis on 10-15 different domains. We wish to test these on broader aspects like 50-100 different domains.

- Labeling anonymous datasets needs to be thought of as part of a general framework for the understanding of the hidden Web that would include service discovery, semantic analysis of the relations between input and output concepts of a service and indexing and high-level querying of semantically analyzed services.

- When DSC assumption does not hold, we may need to consider multiple columns at a time. Unfortunately, multiple column label assignment seems to be NP-hard problem. Future work can investigate more in this regard.

Our main contributions include:

- Developed algorithm LADS and LADSCOMPLETE for Labeling Anonymous Datasets

- The proposed algorithm works fairly well for wrapper generated tables for Labeling Anonymous Datasets

We have shown a new paradigm, *Web Search Engine Based Annotator*, to overcome the labeling/annotation problem. Labeling /Semantic Annotation still has a long way to go, but it will go a long way as the demand is immense.

BIBLIOGRAPHY

[1] Emdad Ahmed. Use of ontologies in software engineering. In *17th International Conference on Software Engineering and Data Engineering (SEDE)*, pages 145–150. ISCA, Los Angeles, USA, June 30 - July 2 2008.

[2] Emdad Ahmed. Achieving classification and clustering in one shot - lesson learned from labeling anonymous datasets. In *4th International Conference on Semantic Computing (ICSC)*, pages 228–231. IEEE, Carnegie Mellon University, USA, September 22-24 2010.

[3] Emdad Ahmed, K. M. Ibrahim Asif, and Miftahur Rahman. Performance analysis of data intensive web application (diwa) - a case study. In *7th International Conference on Computer and Information Technology (ICCIT)*, pages 226–231. BRAC University, December 26-28 2004.

[4] Emdad Ahmed and Hasan M. Jamil. A survey on bioinformatics data and service integration using ontology and declarative workflow query language. In *PhD Qualifying Survey Report*, pages 1–67. Department of Computer Science, Wayne State University, USA, March 2007.

[5] Emdad Ahmed and Hasan M. Jamil. Post processing wrapper generated tables for labeling anonymous datasets. In *11th International Workshop on Web Information and Data Management, co-located with CIKM 2009*, pages 63–66. ACM, WIDM09, Hong Kong, China, November 2 2009.

[6] Emdad Ahmed and Hasan M. Jamil. Resource capability discovery and description management system - labeling anonymous dataset for web data integration. In *PhD Prospectus Technical Report*, pages 1–54. Department of Computer Science, Wayne State University, USA, February 2009.

[7] Mohammad Shafkat Amin and Hasan M. Jamil. Fastwrap: An efficient wrapper for tabular data extraction from the web. In *International Conference on Information Reuse and Integration*, pages 1–8. IEEE, Las Vegas, August 2009.

[8] Luigi Arlotta, Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Automatic annotation of data extracted from large web sites. In *International Workshop on the Web and Databases*, pages 1–6. WebDB, June 12-13 2003.

[9] Anupam Bhattacharjee, Aminul Islam, Mohammad Shafkat Amin, Shahriyar Hossain, Shazzad Hosain, and Hasan M. Jamil. Lifedb: An autonomous system for semantic integration of life science data on hidden web. In *Semantic Web Applications and Tools for Life Sciences*, pages 1–1. SWAT4LS, Scotland, UK, November 2008.

[10] Anupam Bhattacharjee, Aminul Islam, Mohammad Shafkat Amin, Shahriyar Hossain, Shazzad Hosain, Hasan M. Jamil, and Leonard Lipovich. On-the-fly integration and ad hoc querying of life sciences databases using lifedb. In *20th International Conference on Database and Expert Systems Applications*, pages 1–8. DEXA, Linz, Austria, August 2009.

[11] Michael J. Cafarella, Alon Halevy, and Zhe Daisy Wang. Uncovering the relational web. In *11th International Workshop on Web and Databases (WebDB)*, pages 1–6. ACM, June 13 2008.

[12] Michael J. Cafarella, Alon Halevy, and Zhe Daisy Wang. Webtables: Exploring the power of tables on the web. In *VLDB*, pages 1–12. ACM, August 2008.

[13] Michael J. Cafarella, Jayant Madhavan, and Alon Halevy. Web-scale extraction of structured data. *SIGMOD Record*, 37(4):55–61, December 2008.

[14] Michael John Cafarella. *Extracting and Managing Structured Web Data*. PhD thesis, University of Washington, 2009.

[15] Andrew Carlson and Charles Schafer. Bootstrapping information extraction from semi structured web pages. In *ECML*, pages 1–16. ACM, December 2008.

[16] Liangyou Chen and Hasan M. Jamil. Supporting remote user defined functions in heterogeneous biological databases. In *International Symposium on Bio-Informatics and Biomedical Engineering (BIBE)*, pages 144–152. IEEE, November 2001.

[17] Liangyou Chen and Hasan M. Jamil. On using remote user defined functions as wrappers for biological database interoperability. *International Journal of Coopeartive Information Systems (CoopIS)*, 12(2):161–195, June 2003.

[18] Liangyou Chen and Hasan M. Jamil. *AD HOC INTEGRATION AND QUERYING OF HETEROGENEOUS ONLINE DISTRIBUTED DATABASES*. PhD thesis, Mississipi State University, 2004.

[19] Liangyou Chen, Hasan M. Jamil, and Wang N. Automatic wrapper generation for semi structured biological data based on table structure identification. In *DEXA International Workshop on Biological Data Management*, pages 1–10. IEEE, September 2003.

[20] Liangyou Chen, Hasan M. Jamil, and Wang N. Automatic composite wrapper generation for semi structured biological data based on table structure identification. *ACM SIGMOD*, 33(2):58–64, June 2004.

[21] Philipp Cimiano, Gunter Ladwig, and Steffen Staab. Gimme the context: Context driven automatic semantic annotation with c-pankow. In *Proceedings of WWW*, pages 332–341. ACM, May 10-14 2005.

[22] Philipp Cimiano and Steffen Staab. Learning by googling. In *SIGKDD Explorations 6(2)*, pages 24–34. ACM, December 2004.

[23] Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. MIT Press, USA, 2001.

[24] Valter Crescenzi and Giansalvatore Mecca. Automatic information extraction from large websites. *ACM*, 51(5):731–779, September 2004.

[25] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *27th VLDB Conference*, pages 109–118. ACM, September 2001.

[26] Altigran S. da Silva, Denilson Barbosa, Joao M. B. Cavalcanti, and Marco A. S. Sevalho. Labeling data extracted from the web. In *LNCS 4803*, pages 1099–1116. Springer-Verlag, December 2007.

[27] Yihong Ding, David W. Embley, and Stephen W. Liddle. Automatic creation and simplified querying of semantic web content: An approach based on information-extraction ontologies. In *LNCS*, pages 400–414. Springer, June 2006.

[28] Bilal El-Hajj-Diab and Hasan M. Jamil. Bioflow: A web-based declarative workflow language for life sciences. In *2nd International Workshop on Scientific Workflow*, pages 453–460. IEEE, Hawaii, USA, July 8 2008.

[29] David W. Embley, D. M. Campbell, Y. Jiang, Stephen W. Liddle, Y. K. Ng, D. Quass, and R. Smith. Conceptual model based data extraction from multiple record web pages. *Data and Knowledge Engineering*, 31(3):227–251, November 1999.

[30] David W. Embley, Y. Jiang, and Y. K. Ng. Record boundary discovery in web documents. In *SIGMOD*, pages 467–478. ACM, June 1999.

[31] David W. Embley, Cui Tao, and Stephen W. Liddle. Automating the extraction of data from html tables with unknown structure. *Data and Knowledge Engineering*, 54(1):3–28, November 2005.

[32] Avigdor Gal, Giovanni Modica, and Hasan M. Jamil. Ontobulider: Fully automatic extraction and consolidation of ontologies from web sources. In *20th International Conference on Data Engineering (ICDE)*, pages 1–10. IEEE, November 2004.

[33] Fatih Gelgi, Srinivas Vadrevu, and Hasan Davulcu. Fixing weakly annotated web data using relational models. In *LNCS 4607*, pages 1–15. ICWE, July 2007.

[34] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545. ACM, August 23-28 1992.

[35] Le Quang Hieu. Integration of web data sources: A survey of existing problems. In *Institute of Computer Science Heinrich-Heine-University Dusseldorf*, pages 1–5, 2005.

[36] Mingcai Hong, Jie Tang, and Juanzi Li. Semantic annotation using horizonal and vertical contexts. In *ASWC 2006, LNCS 4185*, pages 58–64. Springer-Verlag Berlin Heidelberg, September 2006.

[37] Thomas Hornung, Kai Simon, and Georg Lausen. Mashups over the deep web. In *WEBIST*, pages 228–241. Springer-Verlag Berlin Heidelberg, May 2008.

[38] Md. Shazzad Hosain. *An Algebraic Language for Semantic Data Integration on the Hidden Web*. PhD thesis, Wayne State University, 2010.

[39] Shazzad Hosain and Hasan M. Jamil. An algebraic language for semantic data integration on the hidden web. In *3rd International International Conference on Semantic Computing*, pages 1–8. IEEE, Berkley, California, September 2009.

[40] Shazzad Hosain and Hasan M. Jamil. Algebraic operator support for semantic data fusion in extended sql. In *8th IEEE International Conference on Cybernetic Intelligent Systems (UK and Ireland Chapter)*, pages 1–8. IEEE, University of Birmingham, Birmingham, UK, September 2009.

[41] http://base.google.com. Post it on base. find it on google.

[42] http://metaquerier.cs.uiuc.edu/repository. Bamm datasets.

[43] http://www.dia.uniroma3.it/db/roadRunner/experiments.html. Roadrunner datasets.

[44] Aminul Islam and Hasan M. Jamil. The power of declarative languages: A comparative exposition of scientific workflow design using bioflow and taverna. In *3rd International Workshop on Scientific Workflows*, pages 1–8. IEEE, Los Angeles, CA, July 2009.

[45] CUI Xiao Jun, PENG Zhi Yong, and Wang Hui. Multi-source automatic annotation for deep web. In *International Conference on Computer Science and Software Engineering*, pages 659–662. IEEE, December 2008.

[46] Thomas Kabisch, Ronald Padur, and Dirk Rother. Using web knowledge to improve the wrapping of web sources. In *22nd International Conference on Data Engineering Workshops (ICDEW 06)*, pages 1–10. IEEE Computer Society, September 2006.

[47] Zoe Lacroix. Web data retrieval and extraction. *Elsevier Journal of Data and Knowledge Engineering*, 44(1):347–367, November 2003.

[48] A. H. F. Laender, B. R. Neto, A. S. da Silva, and J. S Teixeria. A brief survey of web data extraction tools. In *SIGMOD Record*, pages 84–93. ACM, June 2002.

[49] Kristina Lerman, Cenk Gazen, Steven Minton, and Craig Knoblock. Populating the semantic web. In *Workshop on Adaptive Text Extraction and Mining (ATEM)*, pages 1–6. AAAI, July 24 2004.

[50] Wei Liu, Derong Shen, and Tiezheng Nie. An effective method supporting data extraction and schema recognition on deep web. In *LNCS 4976*, pages 419–431. Springer-Verlag Berlin Heidelberg, April 2008.

[51] Jiann Jyh Lu and Chun Nan Hsu. Query answering using ontologies in agent-based resource sharing environment for biological web information integration. In *IIWeb*, pages 197–202. IIWeb, March 2003.

[52] Yiyao Lu, Hai He, Hongkun Zhao, Weiyi Meng, and Clement Yu. Annotating structured data of the deep web. In *23rd International Conference on Data Engineering (ICDE)*, pages 376–385. IEEE, April 15-20 2007.

[53] Hoa Nguyen, Eun Yong Kang, and Juliana Freire. Labelex: A scalable approach for extracting form labels. In *24th International Conference on Data Engineering*, pages 1–10. IEEE, April 7-12 2008.

[54] Aleksander Pivk, Philipp Cimiano, and York Sure. From tables to frames. *Web Semantics*, 3(1):132–146, June 2005.

[55] Aleksander Pivk, Philipp Cimiano, York Sure, Matjaz Gams, Vladislav Rajkovic, and Rudi Studer. Transforming arbitrary tables into logical form with tartar. *Data and Knowledge Engineering*, 60(1):567–595, June 2007.

[56] Sriram Raghavan and Hector Garcia Molina. Crawling the hidden web. In *27th VLDB Conference*, pages 1–10. ACM, September 2001.

[57] Nikos Sarkas, Gautam Das, and Nick Koudas. Improved search for socially annotated data. In *PVLDB 2(1)*, pages 778–789. VLDB Endowment, August 2009.

[58] Pierre Senellart, Avin Mittal, Daniel Muschick, Remi Gilleron, and Marc Tommasi. Automatic wrapper induction froom hidden web sources with domain knowledge. In *WIDM*, pages 1–8. ACM, October 30 2008.

[59] Liangcai shu, Weiyi Meng, Hai He, and Clement Yu. Querying capability modeling and construction of deep web sources. In *WISE 2007 LNCS 4831*, pages 13–25. Springer-Verlag, December 3-7 2007.

[60] Avi Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw Hill, USA, 2005.

[61] Kai Simon and Georg Lausen. Viper: Augmenting automatic information extraction with visual perceptions. In *Conference on Information and Knowledge Management (CIKM)*, pages 1–8. ACM, October 31 - November 5 2005.

[62] Youngju Son, Hasan M. Jamil, and Farshad Fotouhi. *Advanced Type-Based Schema Mapping: A New Approach in Data Integration*. PhD thesis, Wayne State University, 2006.

[63] Weifeng Su, Jiying Wang, and Frederick H. Lochovsky. Ode: Ontology-assisted data extraction. *ACM Transaction on Database Systems*, 34(2):12:1–12:35, June 2009.

[64] Shoubiao Tan, Jin Fan, and Yuan Jiang. Web data extraction based on label library. In *World Congress on Computer Science and Information Engineering*, pages 134–138. IEEE Computer Society, August 2009.

[65] Peter D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*, pages 491–502. ACM, September 3-7 2001.

[66] Jiying Wang and Fred H. Lochovsky. Data extraction and label assignment for web databases. In *WWW*, pages 1–13. ACM, 2003.

[67] Tak-Lam Wong and Wai Lam. Adapting web information extraction knowledge via mining site-invariant and site-dependent features. *ACM Transaction on Internet Technology*, 7(1):1–40, February 2007.

[68] Hui Xu, Juanzi Li, and Peng Xu. List data extraction in semi structured document. In *WISE 2005, LNCS 3806*, pages 584–585. Springer-Verlag Berlin Heidelberg, November 2005.

[69] Guan Z and Hasan M. Jamil. Streamling biological data analysis using bioflow. In *International Symposium on Bio-Informatics and Biomedical Engineering (BIBE)*, pages 258–262. IEEE, March 2003.

[70] Patrick Ziegler. Data integration projects world-wide. In *http://www.ifi.unizh.ch/ pziegler/IntegrationProjects.html*, pages 1–8. Department of Informatics, University of Zurich, April 2007.

ABSTRACT

POST PROCESSING WRAPPER GENERATED TABLES FOR LABELING ANONYMOUS DATASETS

by

EMDAD AHMED

May 2011

Advisor: Dr. Hasan M. Jamil

Major: Computer Science

Degree: Doctor of Philosophy

A large number of wrappers generate tables without column names for human consumption because the meaning of the columns are apparent from the context and easy for humans to understand, but in emerging applications, labels are needed for autonomous assignment and schema mapping where machine tries to understand the tables. Autonomous label assignment is critical in volume data processing where ad hoc mediation, extraction and querying is involved.

We propose an algorithm LADS for Labeling Anonymous Datasets, which can holistically label/annotate tabular Web document. The algorithm has been tested on anonymous datasets from a number of sites, yielding very promising results. We report here our experimental results on anonymous datasets from a number of sites e.g., *music, movie, watch, political, automobile, synthetic* 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 98% 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.* We have introduced a new paradigm, *Web search engine based annotator* which can holistically label tabular Web document. We categorize column into three types: disjoint set column (DSC), repeated prefix/suffix column (RPS) and numeric column (NUM). For labeling DSC column, our method rely on *hit counts*

from Web search engine (e.g., Google, Yahoo and MSN). We formulate *speculative queries* to Web search engine and use the *principle of disambiguation by maximal evidence* to come up with our solution. Our algorithm LADS is guaranteed to work for the disjoint set column.

Experimental results from large number of sites in different domains and subjective evaluation of our approach show that the proposed algorithm LADS works fairly well. In this line we claim that our algorithm LADS is robust. In order to assign label for the Disjoint Set Column, we need a candidate set of labels (e.g., *label library*) which can be collected on-the-fly from user SQL query variable as well as from Web Form label tag. We classify a set of homogeneous anonymous datasets into meaningful label and at the same time cluster those labels into a *label library* by learning user expectation and materialization of her expectation from a site. Previous work in this field rely on *extraction ontologies*, we eliminate the need for domain specific ontologies as we could extract label from the Web form. Our system is novel in the sense that we accommodate label from the user query variable. We hypothesize that our proposed algorithm LADS will do a good job for autonomous label assignment. We bridge the gap between two orthogonal research directions: wrapper generation and ontology generation from Web site (i.e., label extraction). We are NOT aware of any such prior work that address to connect these two orthogonal research for value added services such as online comparison shopping.

AUTOBIOGRAPHICAL STATEMENT

EMDAD AHMED

Emdad Ahmed received his Ph.D. degree in Computer Science from Wayne State University (USA) in May 2011. He obtained Australian Govt scholarship (AusAID) to pursue Master of Engineering Science (MEngSc) from University of New South Wales, Australia in 2001. He has received Master of Business Administration (MBA) from Institute of Business Administration, Dhaka University in 1998 and BSc.Engg (Computer Science and Engineering) from Bangladesh University of Engineering and Technology (BUET) in 1994 respectively.

While his general research interest is in the field of databases, Bioinformatics, his current focus is in the areas of Web data Integration. His research has resulted in several referred international conference papers, including articles that appeared in ACM WIDM, SEDE and IEEE ICSC international conferences.

As a Ph.D. student in Wayne State University, Emdad Ahmed was a recipient of several competitive awards that include Graduate Teaching Assistant, Part Time Faculty during Spring/Summer, Graduate Teaching Assistant Professional Travel award etc.

He has been lecturer, Department of Computer Science and Engineering, Khulna University during 1994. He has worked as Computer Programmer (Local Consultant) in a World Bank Project, Female Secondary School Assistant Project under Directorate of Secondary and Higher Education of Ministry of Education, Bangladesh from 1994 to 1997. He has served as lecturer, Department of Electrical Engineering and Computer Science, North South University from 2001 to 2005. He has started faculty job at Montana State University Billings from Fall 2010.