Achieving Classification and Clustering in One Shot
Lesson Learned from Labeling Anonymous Datasets

First Author
Department, University
City, State, Zip
author1@university.edu

Second Author
Department, University
City, State, Zip
author2@university.edu

Abstract—We propose an algorithm LADSCOMPLETE which can holistically assign label for tabular web data. We categorize columns into three types: Disjoint Set Column (DSC), Repeated Prefix / Suffix Column (RPS) and Numerical Column (NUM). For labeling DSC column, our method rely on hits count 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. Experimental results from large number of sites in different domains and subjective evaluation of our approach show that the proposed algorithm LADSCOMPLETE works fairly well. In this line we claim that our algorithm LADSCOMPLETE is sound and complete. 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. We hypothesize that our proposed algorithm LADSCOMPLETE 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.

Index Terms—Wrapper, Web Form, Hidden Web, HTML Table, Web Data Integration.

I. INTRODUCTION

Our research is motivated by the fact that search engine like Google is researching on how to extract data from the HTML tables and later on how to use this information for searching using keywords. We are working hard toward developing a framework for automated information extraction from Internet documents. We are developing an intelligent Column Name Assignment algorithm LADSCOMPLETE for Labeling Anonymous Datasets. Current technology does not support assigning meaningful column names to the extracted tables when they are missing. Our algorithm LADSCOMPLETE has achieved as high as 98% probability of assigning good label to Anonymous Datasets. Labels are required for making unstructured text, web data to be structured one to facilitate, say, comparison shopping and services. That is where our contribution is. Previous works in this field like in DELA, Labeller component of ROADRUNNER [1], MULTIANNOTATOR [2], ADEL [3], VIPER [4] etc. all have limitations. In general the limitations come from two well known facts: (i) Not all web pages contain labels (e.g in search engine result pages, amazon.com book search results etc.) and (ii) We may NOT like the labels chosen by the web page authors (i.e user view is different from web page content view). Using our approach, we have overcome the above mentioned limitations.

II. PROBLEM STATEMENT: THE BIG PICTURE

There are four steps to the Web Data Extraction and Labeling workflow: Retrieval corresponds to $\mathcal{IR}$ to using any programming language API like Java, Python library URL, URLconnection etc; Extraction corresponds to $\mathcal{IE}$ to web wrapper like DELA [5] which can extract tabular web data as anonymous datasets. Note that $\mathcal{IR}$ and $\mathcal{IE}$ are two separate processes. The third step of the workflow corresponds to Web Data Integration (e.g link and combine), but before the third step come into play, Extraction process need to be augmented with Labeling / Annotation process, i.e we need to post process the wrapper generated tables for labeling anonymous datasets, i.e $\mathcal{IE}$ and Labeling / Annotation are two separate processes. The problem of wrapper generation function $\omega$ can be stated as follows [6]:

Definition 1. Wrapper Generation Given a Web page $S$ containing a set of implicit objects, determine a mapping $\mathcal{W}$ that populates a data repository $\mathcal{R}$ with the objects in $S$, i.e $\mathcal{W} : S \rightarrow \mathcal{R}$. The mapping $\mathcal{W}$ must also be capable of recognizing and extracting data from any other page $S'$ similar to $S$.

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 $A_r = \{attr_{r1}, attr_{r2}, \ldots, attr_{rm}\}$. 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 2. QC A Query Condition (QC) is a 3-tuple: $\{L, \Xi, \mathcal{V}\}$, where $L$, $\Xi$ and $\mathcal{V}$ are sets of labels, relational operators and instance data values respectively. $\Xi$ is any relational operator such as $=, \leq, \geq, \neq$ etc. Each Query Condition QC can be modelled using SQL syntax

\footnote{http://www.scotops.com/google-to-use-tables-on-web_280/}
as:

SELECT \{attr_{r_1}, attr_{r_2}, \ldots, attr_{r_m}\}
FROM DB
WHERE attr_{r_1} = val_{q_1}, attr_{r_2} = val_{q_2}, \ldots, attr_{r_n} = val_{q_n}
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 \( R \subseteq L \times V \). The following Figure 1 depicts the overall scenario of the problem we are going to model and address. Our model assumes that in response to the user query sub-

![Figure 1](image_url)

Definition 7. 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 I is an example of Unannotated Table.

Definition 8. 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 \( \forall i,j, D_i \cap D_j = \emptyset \). Example in Table I, \( A_1 \) contains title, \( A_2 \) contains author, they are Disjoint Set Column (DSC).

We categorize columns into three types: Disjoint Set Column (DSC), Repeated Prefix/ Suffix Column (RPS) and Numerical Column (NUM). Below we formally define RPS and NUM.

Definition 9. 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). Example in Table I, \( A_4 \) contains stars, it is a Repeated Prefix/ Suffix Column (RPS)

Definition 10. Numerical 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 Numerical Column (NUM). Example in Table I, \( A_3 \) contains $18.95, it is a Numerical Column (NUM)

In HiWE [9] task-specific information is organized in terms of a finite set of concepts or categories. Each concept has one or more labels and an associated set of values. We define LVS as follows:

Definition 11. LVS Label Value Set table (LVS) is of the form \((L,V)\), where \(L\) is a set and \(V = \{v_1, \ldots, v_n\}\) is a fuzzy/ graded set of values. For example, the label 'Company Name' could be associated with the set of values \(\{IBM, Microsoft, HP, \ldots\}\)

Definition 12. Label Extraction Let \( L = \{l_1, l_2, \ldots, l_m\} \) be a set of textual labels located in an HTML page \( P \), and let \( W \) be a Web form located in \( P \) that contains set of elements \( E = \{e_1, e_2, \ldots, e_n\} \). The label extraction problem consists of determining the set of mappings \( M \) between each form element \( e_i \) in \( E \) to exactly one label \( l_i \) in \( L \), such that: \( M = (e_i, l_i)|l_i \subseteq L \) and \( l_i \) best describes the meaning of \( e_i \).
Our label selection function \( \sigma \) 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 SQL query variable \( Q \). Thus \( \mathcal{L} \subseteq (\mathcal{F} \cup Q_s \cup Q_w) \).

**Research hypothesis:** Given a candidate set of labels (using \( \sigma \) function) and an anonymous datasets (using \( \omega \) function), can web search engines such as Google, Yahoo, MSN be used as \( \mu \) function to assign label for the anonymous datasets? We answer positively in this paper. Using our method, we can efficiently assign label for the columns in Table 1 \( A_1 \) and \( A_2 \) as title and author respectively.

### A. Motivation

Our research is motivated by the work of schema discovery [10]. Cafarella et al mentioned that some wrapper induction system like IEPAD, ROADRUNNER and EXALG use pattern discovery to extract likely database values, however they do NOT attempt to find a label for each extracted value (in ROADRUNNER, extracted data fields are specified with labels \{A, B, C, . . .\}), forcing a human to later mark the different attributes.

As mentioned by [11] in their system Ontology-Assisted Data Extraction (ODE), one of the limitation is that to label attributes, it is necessary that the labels appear in the query interfaces or query result pages within a domain. However, there are some attributes whose labels never appear in any query interface or query result page. Consequently, such attributes can NOT be labeled. To overcome this limitation, [11] suggested that it might be possible to use the Web. In this line, we have used hits count of Web Search Engine such as Google, Yahoo and Bing to label anonymous datasets and we will show that our approach works fairly well. To this end, we define and study the problem of autonomous label assignment for anonymous datasets of wrapper generated tables.

### B. Use of Web Search Engine

In general the approach that we have taken to assign column label is to use the hits count from web search engine to disambiguate column labeling. In the literature the method has been termed as ‘self annotating web’ [12], [13], that is web page is used to annotate web page. 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 telephone like network for Quality of Service (QoS). The similar cycle has been observed while labeling anonymous datasets. We had begun our journey from raw, unstructured text, then we developed structured text as DB (during 1970’s), again we are in need of raw, unstructured text (web page text) for Labeling.

Marti Hearst, 1992 [14] applied regular expression patterns, called lexico-syntactic patterns, for the extraction of semantic relations from the text. We have considered methods based on term co-occurrence on lexico-syntactic patterns. Useful lexico-syntactic patterns reused from Hearst in our context are:

### TABLE II

| \( P_1 \) \(<L> <V> \) | \( H_1 <L>s \) such as \( <V|L'> \) | \( H_2 <L>s \) as \( <V|L'> \) | \( H_3 <L>s, \) (especially including) \( <V|L'> \) |
|---|---|---|---|

\( P_1 \) refer to our phrase pattern i.e label followed by value (“L V”), \( H_1 \), \( H_2 \) and \( H_3 \) refer to Hearst patterns. Our approach to using web search engine also falls in the category of seminal work PMI-IR by [15].

### III. RELATED WORK

Cafarella et al [10] worked on navigating extracted data with schema discovery. We present Table 1 as high level understanding of the problem that we are ultimately going to address. In their Open Information Extraction (OIE) model, an unstructured tuple \( U \) can be described as \( U = (\text{Key}, \{V_1, V_2, \ldots\}) \) where Key is a unique key and each \( V_i \) is a value description, consisting of a value \( v \) and a set of (label, preference) pairs: \( V_i = (v, \{(P_1, L_1), (P_2, L_2), \ldots\}) \) where \( v \) is the actual value, \( L_i \) is a candidate label for that value, and \( P_i \) is its preference rank. This notion of label-value is just the opposite to our notion of LVS, defined earlier. The input to the schema discovery system is the set of all input tuples, \( U \), and the set of all labels occurring in all tuples, denoted \( L \). The output is a schema \( S \), and an assignment \( A \). The schema is a set of table schemas \( S = \{T_1, T_2, \ldots\} \), where each table schema is defined as a set of attributes from \( L \). Thus, each \( T_i \) can be identified with a set of attributes, \( Attr(T_i), \text{s.t.} Attr(T_i) \subseteq L \). The assignment, \( A \), is a many-to-many relationship between the unstructured tuples in \( U \) and the tables in \( S \), i.e, it consists of a set of pairs \( (U, T) \), where \( U \in U \) and \( T \in S \). Given \( L \) labels, there are \( 2^{|L|} \) possible unique tables. Our proposed system can reduce the complexity to a great extend. The details can be understood from the following discussion. Suppose we are working on car domain. We can decompose the functionality of car domain into 4 orthogonal category: car trading, car rental, car insurance and car service. Note that the labels for the 4 categories are disjoint and there is no point keeping all the labels together. Suppose for the above example \( |L| = |L| + |r| + |t| + |s| \), where \( |L| \) represents all the labels for the car domain, \( |r| \), \( |t| \) and \( |s| \) represent labels for car trading, rental, insurance and service respectively. Note that \( 2^{|l|} + 2^{|r|} + 2^{|t|} + 2^{|s|} \ll 2^{|L|} \). This is exactly what we want to achieve. By grouping related labels in different bin, we remove unnecessary “L V” validation query to be submitted in web search engine to disambiguate column labeling. To reduce unnecessary query to web search engine, the similar idea has been used by Cimiano et al in their system C-PANKOW [13].
IV. LABELING ALGORITHM

In this section we present the details of our proposed algorithm LADSComplete. Whereas [2] pointed out that no single annotator can annotate all the columns in web table, ours is a first attempt to holistically annotate all the columns.

Our algorithm LADSComplete is based on the assumption that many HTML tables’ column contains instances of a single concept (i.e. we assume that in a column the instance data values are homogeneous). Before presenting the algorithm, the following definition of Speculative Query will be helpful to make understand how we pose queries to web search engine.

**Definition 13.** A speculative query is a 3-tuple: \( (L, \mathcal{P}, \mathcal{V}) \), where \( L, \mathcal{P} \) and \( \mathcal{V} \) are set of labels, patterns and instance data values respectively. In order to overcome the limitation of keyword based search paradigm, we formulate speculative query to web search engine as exact phrase matching. Example: “title Discrete Mathematics” is a speculative query.

The step by step procedure to solve the labeling problem in our context are as follows:

1. **Step1:** Extract label from Web Form (refer to work in HiWE [9] and labelEx [16]) as well as from user SQL query variable (select, where clause)
2. **Step2:** Formulate and execute speculative queries to web search engines (e.g. Google, Yahoo and MSN)
3. **Step3:** Prune and Rank Results (e.g compute Statistical Fingerprints)

To reduce the computational complexity, we employ a random sampling function \( \psi \), which given an Unannotated Table \( T \), returns a subset of it \( T' \), where \( T' \subset T \). In our experiment, the function \( \psi \) 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 assign label for an Unannotated Table \( R(A_1, A_2, \ldots, A_n) \), we are in need of a candidate set of labels \( L \). Then we apply our label assignment function \( \mu \), which takes as parameter \( T' \) and a candidate set of labels \( L \) and the function \( \mu \) will assign label for \( A_1, A_2, \ldots, A_n \), i.e \( \mu: (T', L) \rightarrow A_1, A_2, \ldots, A_n \), where \( 1 \leq i \leq n \).

The labeling problem which we deal in this paper is that many HTML tables’ column contains instances of a single concept (i.e. we assume that in a column the instance data values are homogeneous). Before presenting the algorithm, the following two definitions will be helpful to make understand how we pose queries to web search engine.

**Definition 14.** 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 15.** Given an Unannotated Table \( R(A_1, A_2, \ldots, A_n) \) and a candidate set of labels \( \mathcal{L} = \{ l_1, l_2, \ldots, l_m \} \), the Label Attribute Affinity between \( A_j \) and \( l_i \), denoted \( LAA(A_j, l_i) \), is defined as:

\[
LAA(A_j, l_i) = \frac{1}{|A_j|} \sum_{i=1}^{|A_j|} \frac{DC(l_i \land v_j)}{DC(l_i \land v)}
\]

Both the sets, the label set \( \mathcal{L} \) (for example BAMM datasets [18]) as well as the value set \( \mathcal{V} \) (for example RoadRunner datasets [19]) are known - the task of the Labeling component is to identify correspondence between \( l_i \in \mathcal{L} \) and \( v_j \in \mathcal{V} \).

A web search engine can be used in this scenario as ranker.

**Algorithm 1** LADSComplete: Labeling Anonymous Datasets Complete

```plaintext
1: INPUT
2: \( L(1 \ldots m) \): A set of labels
3: \( A(1 \ldots n) \): A set of anonymous attributes
4: Precondition \( m > n \)
5: \( P(1 \ldots k) \): A set of patterns
6: \( NP(1 \ldots l) \): A set of generic numeric patterns
7: \( (e.g \$, year, dd = mm - yy) \)
8: \( V(1 \ldots t, 1 \ldots n) \): Tabular Data Value
9: \( s \): Odd Number of Search Engine i.e 1, 3, 5 etc
10: Search Engine Name : e.g Google, Yahoo, MSN
11: VARIABLE
12: \( N(1 \ldots s) \): Search Engine hits count
13: \( H(1 \ldots n, 1 \ldots m, 1 \ldots s) \)
14: \( 3 \times m \times n \) array to store hits count
15: \( R \rightarrow 1 \ldots t \) Random indexes between 1 \ldots \( t \)
generate one time
16: Formulate Speculative query : \( L \times P \times V \rightarrow N \)
17: for \( i = 1 \) to \( n \) do
18: Data Preprocessing
19: determine column type to be DSC, RPS, NUM
20: if column \( A_i \) is of type NUM then
21: Label \( A_i \) to be one from \( NP \)
22: else if column \( A_i \) is of type RPS then
23: Remove RPS from \( A_i \) and Label it with RPS
24: else
25: \( H[i][1..m][1..s] = 0 \)
26: for each \( r \in R \) do
27: \( iValue = V_r \)
28: for \( j = 1 \) to \( m \) do
29: \( lValue = L[j] \)
30: for \( p = 1 \) to \( k \) do
31: \( qVar = lValue + P[p] + iValue \)
32: for \( q = 1 \) to \( s \) do
33: \( H[i][j][q] + = N[q] \)
34: end for
35: end for
36: end for
37: end for
38: end for
39: Let \( z \) be the majority index \( (1 \ldots m)(1 \ldots s) \) of highest count in \( H[i][1..m][1..s] \)
40: \( A[i] = L[z] \)
41: end if
42: end for
43: end for
```
Table IV
Subjective Evaluation of Labeling

<table>
<thead>
<tr>
<th>Domain</th>
<th>No. of anonymous attr</th>
<th>matches</th>
<th>mismatches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Music</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Political</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Synthetic</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Watch</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Automobile</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table V
Greedy Labeling on Anonymous Datasets

<table>
<thead>
<tr>
<th>Domain</th>
<th>Candidate set of labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>Actor, Director, Genre, Rating, Title, Film</td>
</tr>
<tr>
<td>Music</td>
<td>Artist, Title, Album</td>
</tr>
<tr>
<td>Political</td>
<td>President, Governor, Senator, Vice President</td>
</tr>
<tr>
<td>Synthetic</td>
<td>Toy, Furniture, Electronics, Clothing</td>
</tr>
<tr>
<td>Watch</td>
<td>Ward, Brand, Gender, Display, Model, Category</td>
</tr>
<tr>
<td>Automobile</td>
<td>Model, Color, Make, Static, Type, Class, Style, Category</td>
</tr>
</tbody>
</table>

We have manually computed candidate set of labels for 4 domain: Book, Automobile, Movie and Music (BAMM) from UIUC web data integration repositories [18], the set of labels are shown in the Table III. The datasets has been crafted from around 50 deep web sources. We have run our algorithm LADSCOMPLETE on Automobile domain with the candidate set of labels from BAMM datasets [18] and anonymous datasets from RoadRunner Project [19]. We have tested our approach on small scale datasets and found that the results are favorable. We compare the results of LADSCOMPLETE with that of human subject matter expert. We define the goodness of our algorithm LADSCOMPLETE as the number of matches between LADSCOMPLETE assigned labels to that of human assigned labels. We define subjective evaluation function as: \( f(\text{dom}) = g(\text{dom}) + h(\text{dom}) \) where \( \text{dom} \) represent some domain and \( g(\text{dom}) \) represents the number of anonymous attributes (DSC columns only), \( g(\text{dom}) \) represents the number of matched anonymous attributes and \( h(\text{dom}) \) represents the number of mismatched anonymous attributes.

9 CS experts were chosen to label the anonymous datasets from a wide variety of domains. Some of the results are shown in Table IV. Clearly we will require that \( g(\text{dom}) \) should be much higher than \( h(\text{dom}) \), as is evidenced from our subjective evaluation in Table IV, shows that our algorithm LADSCOMPLETE is able to correctly assign good label for the DSC columns as per the user expectation most of the time. We have proved somewhere else that phrase pattern PL i.e “L V” alone is enough to disambiguate column labeling. Now we give alternative proof for the same by solid statistical method, Pointwise Mutual Information (PMI). PMI-IR was defined by Turney (2001) [15]. We have found that Web-based PMI statistics can be effective to disambiguate column labeling. We re-define PMI(L, V) into our labeling context as follows:

Definition 16. PMI(L, V) Pointwise Mutual Information between a label L and instance data value V is the number of hits for a query that combines the label L and instance data value V divided by the hits for the instance data value V alone.

\[ PMI(L, V) = \frac{\text{Hits}((L \text{ and } V))}{\text{Hits}(V)} \]

We have applied PMI(L, V) in our context to estimate which label L has the highest affinity with instance data value V.

VI. PROOF OF LEMMA

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

Lemma 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 LADSCOMPLETE has the Greedy Choice Property [20].

Lemma 2. Algorithm LADSCOMPLETE is not only greedy, but also optimal.

We define optimality as follows: the order of choosing anonymous attributes to be labeled in each iteration does NOT matter.

Now we present Table V, showing a number of domain by which we have done our experiment to prove the above two lemma. The probabilistic labeling result for these domains have been shown in section X. 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 permutation of the anonymous attributes. Therefore the worst case complexity to prove lemma 2 is \( O(n!) \). However it is sufficient to prove the second lemma by considering two orders only: forward and backward. Therefore we choose to run our algorithm LADSCOMPLETE using forward and backward order only. Forward order corresponds to choosing the anonymous attributes starting from...
TABLE III
MANUALLY COMPUTED SET OF LABELS FOR DIFFERENT DOMAIN

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

1, 2, . . . , n, whereas backward order correspond to do the reverse, n, n − 1, . . . , 2, 1. For both the cases, we have found that our algorithm LADSCOMPLETE have produced the same set of labels for the anonymous attributes. Thus we prove lemma 2. Now we present the following Theorem 1.

**Theorem 1.** Algorithm LADSCOMPLETE is guaranteed to work for DSC column.

**Proof** First we show that our algorithm LADSCOMPLETE is able to assign label for all the anonymous datasets. This step of the proof is trivial. Given a candidate set of m labels 1, 2, . . . , m and an anonymous datasets n, A1, A2, . . . , An, where m > n the algorithm LADSCOMPLETE terminates in time complexity of 900ms with assigning label for all the anonymous attributes A1, A2, . . . , An. Now we make use of our disjoint set column assumption and using Lemma 1 and 2, we prove Theorem 1. (*proved*)

VII. HOW TO POPULATE LABEL IN A DOMAIN

Suppose a user want to query a site in political domain. The domain has a set of labels like President, Vice President, Senator, Governor, Mayor, Party. Consider the following user query:

```
Select President, Party
From site S1
Where year = 1930;
```

The screen shot of the site S1 is given in Figure 2. In the screen shot, there are 4 columns, whereas user want result from 2 columns only. The user may issue query later to ask for the Party information as follows:

```
Select President, Party
From site S1
Where year = 1930;
```

We may consider a session of queries from a user to the same site. We define a session as “a series of interaction by the user toward addressing a single information need (I/N)”’. In this context, we have identified 2 different approaches to column labeling:

- Annotate all the available columns of a web table first time in one shot
- Annotate column every time incrementally (See in the 2nd query, Party information was not asked in the first query)

Now we present the high level discussion on how to populate labels in a domain:

**Step 1:**
Compute all the affinity of the users query variable to web table and report result as ranked order list of labels for the anonymous datasets.

**Step 2:**
If user was happy with the result:
It is an indication that the query was successful
Check the DOMAINs list of labels
For each label in User Query Variable DO
    If the user query label is already there in DOMAIN
        Do nothing
    Else
        Add the user supplied label to the DOMAIN
Else
    Do nothing

By doing so, we are achieving two objectives in one shot:

- Classifying a set of homogeneous objects into a label (fully automatic)
- Clustering a set of terms in a domain (semi automatic)

VIII. SYSTEM ARCHITECTURE AND IMPLEMENTATION

A. Four Level of Data Abstraction

We define four levels of data abstraction as follows: Concept Layer; Sub Concept Layer; Terminology Layer and Data Layer. Concept Layer is the highest one (e.g car domain). Sub Concept Layer is immediate below the Concept Layer (i.e car trading, car rental, car insurance, car service). Terminology Layer is immediate below the Sub Concept Layer (e.g terms like make, model, type). Data Layer is the lowest one, it will hold the actual data i.e car made of, car type (i.e Acura, Sedan etc). Table VI shows the different layers and the responsibility for who will take care of the layers and hints how to address it. Our proposed algorithm LADSCOMPLETE will connect the dots i.e bridge the gap between Terminology and Data Layer.

B. Layered Architecture

The high level architecture of the system is shown in Figure 3. Here we implement the system as a layered architecture consist of four layers as depicted in the following Figure 3. Due to space limitation, we omit the details of each layers.
IX. CONCLUSIONS AND FUTURE WORKS

Our contribution has been to develop a fully automatic, generic method for labeling anonymous datasets based on Multiple Search Engines’ Recommendation. We have used the principle of disambiguation by maximal evidence to assign column label. We have shown that the proposed algorithm LAD-SCOMPLETE works fairly well in different domain. Search engines’ hits count can be inaccurate, but have found to be useful in practice. It shows a new paradigm to overcome the annotation problem, unsupervised instance categorization. But its a difficult task, there are open domain, many categories. Labeling/ Semantic Annotation still has a long way to go, but it will go a long way as the demand is immense. Possible future works include:

- Label Column Name based on Search Engine snippets only
- 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.

X. SAMPLE RESULT OF LABELING

Here we provide some results of our experiment from a number of domains. The correct results are high lighted in the tables.

REFERENCES

<table>
<thead>
<tr>
<th>Table XII</th>
<th>Probabilistic Labeling on Watch Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability</td>
<td>Value</td>
</tr>
<tr>
<td>P(A₁</td>
<td>band)</td>
</tr>
<tr>
<td>P(A₁</td>
<td>brand)</td>
</tr>
<tr>
<td>P(A₁</td>
<td>gender)</td>
</tr>
<tr>
<td>P(A₁</td>
<td>display)</td>
</tr>
<tr>
<td>P(A₁</td>
<td>model)</td>
</tr>
<tr>
<td>P(A₁</td>
<td>category)</td>
</tr>
<tr>
<td>P(A₂</td>
<td>band)</td>
</tr>
<tr>
<td>P(A₂</td>
<td>brand)</td>
</tr>
<tr>
<td>P(A₂</td>
<td>gender)</td>
</tr>
<tr>
<td>P(A₂</td>
<td>display)</td>
</tr>
<tr>
<td>P(A₂</td>
<td>model)</td>
</tr>
<tr>
<td>P(A₂</td>
<td>category)</td>
</tr>
<tr>
<td>P(A₃</td>
<td>band)</td>
</tr>
<tr>
<td>P(A₃</td>
<td>brand)</td>
</tr>
<tr>
<td>P(A₃</td>
<td>gender)</td>
</tr>
<tr>
<td>P(A₃</td>
<td>display)</td>
</tr>
<tr>
<td>P(A₃</td>
<td>model)</td>
</tr>
<tr>
<td>P(A₃</td>
<td>category)</td>
</tr>
<tr>
<td>P(A₄</td>
<td>band)</td>
</tr>
<tr>
<td>P(A₄</td>
<td>brand)</td>
</tr>
<tr>
<td>P(A₄</td>
<td>gender)</td>
</tr>
<tr>
<td>P(A₄</td>
<td>display)</td>
</tr>
<tr>
<td>P(A₄</td>
<td>model)</td>
</tr>
<tr>
<td>P(A₄</td>
<td>category)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table XIII</th>
<th>Anonymous Datasets from Automobile Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate Set of Labels: Model, Color, Make, State, Class, Style, Category</td>
<td></td>
</tr>
<tr>
<td>A₁</td>
<td>A₂</td>
</tr>
<tr>
<td>Acura</td>
<td>Sedan</td>
</tr>
<tr>
<td>Acura</td>
<td>Coupe</td>
</tr>
<tr>
<td>Audi</td>
<td>Sedan</td>
</tr>
<tr>
<td>Audi</td>
<td>Coupe</td>
</tr>
<tr>
<td>Audi Convertible</td>
<td>2001 TT</td>
</tr>
<tr>
<td>Audi Wagon</td>
<td>2001 A4</td>
</tr>
<tr>
<td>Ford</td>
<td>Sedan</td>
</tr>
<tr>
<td>Cadillac</td>
<td>Sedan</td>
</tr>
<tr>
<td>Audi</td>
<td>Coupe</td>
</tr>
<tr>
<td>Chrysler</td>
<td>Sedan</td>
</tr>
<tr>
<td>Volvo</td>
<td>Sedan</td>
</tr>
<tr>
<td>Volvo</td>
<td>Coupe</td>
</tr>
<tr>
<td>Volvo Convertible</td>
<td>2001 C70</td>
</tr>
<tr>
<td>Volvo Wagon</td>
<td>2001 V70</td>
</tr>
<tr>
<td>Mercedes</td>
<td>Sedan</td>
</tr>
<tr>
<td>Mercedes</td>
<td>Coupe</td>
</tr>
<tr>
<td>Mercedes Convertible</td>
<td>2001 CLK-Class</td>
</tr>
<tr>
<td>Mercedes</td>
<td>Wagon</td>
</tr>
<tr>
<td>Saab</td>
<td>Sedan</td>
</tr>
<tr>
<td>Saab</td>
<td>Convertible</td>
</tr>
<tr>
<td>Jeep</td>
<td>Sport Utility Vehicle</td>
</tr>
</tbody>
</table>


