Towards a Global Schema for Web Entities

Conglei Yao, Yongjian Yu, Sicong Shou, Xiaoming Li
Department of Computer Science and Technology
Peking University
Beijing, 100871, P. R. China
{ycl, yyj, ssc}@net.pku.edu.cn, lxm@pku.edu.cn

ABSTRACT

Popular entities often have thousands of instances on the Web. In this paper, we focus on the case where they are presented in table-like format, namely appearing with their attribute names. It is observed that, on one hand, for the same entity, different web pages often incorporate different attributes; on the other, for the same attribute, different web pages often use different attribute names (labels). Therefore, it is imaginably difficult to produce a global attribute schema for all the web entities of a given entity type based on their web instances, although the global attribute schema is usually highly desired in web entity instances integration and web object extraction. To this end, we propose a novel framework of automatically learning a global attribute schema for all web entities of one specific entity type. Under this framework, an iterative instances extraction procedure is first employed to extract sufficient web entity instances to discover enough attribute labels. Next, based on the labels, entity instances, and related web pages, a maximum entropy-based schema discovery approach is adopted to learn the global attribute schema for the target entity type. Experimental results on the Chinese Web achieve *weighted average Fscores* of 0.7122 and 0.7123 on two global attribute schemas for person-type and movie-type web entities, respectively. These results show that our framework is general, efficient and effective.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition*

General Terms

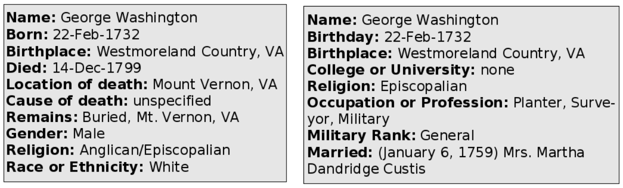
Algorithms, Experimentation

Keywords

Web Entities, Attribute Labels, Entity Instances, Global Attribute Schema

1. INTRODUCTION

With the rapid expansion of the Web, more and more entity instances, such as persons, books and movies, are appearing on the Web. Most of them exist in structured web pages where their contents are organized in table-like format. Each page has a set of data regions, and each region stands for an entity instance. It's a main research line to identify data regions and extract corresponding entity instances from structured pages [15, 22, 20, 24] in Web mining. The entity instances, defined as *data records* [15, 22] or *web objects* [20, 24], often include names, attribute values, and labels which describe the attribute types.



Name: George Washington
Born: 22-Feb-1732
Birthplace: Westmoreland Country, VA
Died: 14-Dec-1799
Location of death: Mount Vernon, VA
Cause of death: unspecified
Remains: Buried, Mt. Vernon, VA
Gender: Male
Religion: Anglican/Episcopalian
Race or Ethnicity: White

Name: George Washington
Birthday: 22-Feb-1732
Birthplace: Westmoreland Country, VA
College or University: none
Religion: Episcopalian
Occupation or Profession: Planter, Surveyor, Military
Military Rank: General
Married: (January 6, 1759) Mrs. Martha Dandridge Custis

Figure 1: Two person instances

For example, both instances in Figure 1 contain several attributes regarding the same person. For the attribute type *"Birthday"*, the first instance uses *Born* as its label, which is different from *Birthday* as used in the second. Therefore, we should make sure they are of the same attribute type if we want to integrate these two instances. In fact, it's common for entities of one entity type to have many types of attributes with each of them described by different labels. Consequently, to integrate the web entity instances of the same entity type, we have to establish a map between labels and their types. To establish this map, a global attribute schema, which reflects the main attribute types and their descriptive labels, must be constructed in advance.

We focus on automatically learning a global entity attribute schema for all the web entities of a specific entity type. For example, for person-type entities, a basic description that a person entity should contain attributes of *name, gender, birthday, birthplace* and *weight* is provided by the users. Then, based on this description, we can locate related web pages to extract person entity instances, mine other latent attributes, and then use the new attributes to locate more pages for more instances. This process runs iteratively until necessary attribute labels for the main attribute types have been discovered. Based on these attribute labels, entity instances and related pages, we discover the relations between different attribute labels, and, finally, construct the global entity attribute schema for person-type web entities.

The global attribute schema for web entities of a specific entity type is essential to integrate web entity instances. Moreover, from the global schema, the types of the impor-

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

tant attributes owned by the web entities of the target entity type can be automatically identified, and then used as the input to web object extraction systems to extract web objects with these important attributes. And this makes the results of these systems more valuable and reasonable.

However, for this aim, there are obvious difficulties. On the one hand, enough frequent attribute labels must be extracted from the Web, and they should also be able to represent the main attribute types which constitute the global schema. On the other hand, the number of frequent labels in learning a global attribute schema is usually large and their qualities are not uniformly high, due to the variety and complexity of web entity instances.

In this paper, we propose the problem of learning a global attribute schema for all the web entities of a specific entity type for the first time, and present a novel framework to address this problem. Under this framework, first, we propose an iterative web entity instances extraction approach to extract enough web entity instances as well as frequent attribute labels to learn a global attribute schema. Next, we present a maximum entropy-based method to automatically learn a global attribute schema based on the frequent labels, entity instances and related pages. Finally, we demonstrate our technique on person-type and movie-type entities on the Chinese Web, and experimental results show that our technique is general, efficient and effective.

This article is further structured as follows. Section 2 discusses related works. Section 3 formulates the problem of learning a global attribute schema. Section 4 presents the iterative method of entity instances extraction. Section 5 proposes the maximum entropy-based approach of schema discovery. Section 6 illustrates the experiments and analyzes the experimental results. Section 7 summarizes this paper.

2. RELATED WORK

To the best of our knowledge, no research has yet adequately addressed the problem of learning a global attribute schema from the Web for entities of a given entity type.

The previous study in [8] seeks to discover hidden schema model for query interfaces on deep Web. This study focuses on finding a hidden schema model to match schemas of query interfaces in the same domain. Its goal is the same as ours in finding a global schema. However, its method is effective and efficient only when the number of attribute labels in query interfaces is relatively small, because it creates the global schema based on all possible label partitions, and its time complexity is exponential to the number of labels. Moreover, it assumes that attribute labels in each query interface's schema are of different types, which is natural in deep Web. However, in our research, the attribute labels and the possible label partitions are both far more numerous than the aforementioned study. Further, the attribute labels in one entity instance are not always different types.

Besides, the other related researches can be summarized in the following aspects:

2.1 Schema Extraction

Schema extraction is the task of automatically discovering implicit structural information from semi-structured data. Web data belongs to semi-structured data class, i.e., data with self-contained schema [1]. Previous studies [19, 11, 3] discover schema for web data by utilizing the structure of web pages. In these studies, schema for web data can be constructed at different levels: giving a schema for a set of logically related sites by viewing them as a whole [2]; examining a single site [3]; or, structuring single page.

Similar to these studies, our research aims at learning a schema for web entities. However, our study is different from them in three aspects. First, we target learning a schema from large numbers of entity instances and related web pages, instead of from a single page, a single site or a set of logically related sites. Second, our study mines the attribute labels, entity instances and related pages, instead of analyzing the structure of HTML documents. And lastly, our study seeks to learn a global schema for all the web entities of a specific entity type, while previous related studies aimed to extract the schema of web data appearing in some sites or pages.

2.2 Schema Matching

Schema matching aims at discovering semantic correspondences of attributes of schemas across heterogeneous sources. Previous studies can be classified into schema-level matching [17, 5, 9, 16] and instance-level matching [14, 6, 12, 4]. A recent research [21] proposes a unified solution to address the two corresponding schema matching problems of intra-site and inter-site. [8] assumes that a hidden generative distribution exists in related schemas, and presents a statistical schema matching method to reconstruct the hidden generative distribution based on the input schemas. [10] addresses the problem of complex schema matching, which matches a set of m attributes to another set of n attributes from two individual schemas.

In our research, we first look for frequent attribute labels in entity instances from different pages, and then combine labels which belong to the same attribute type. Naturally, the attribute labels correspond to elements in schemas. However, our research is unlike schema matching. First, schema matching integrates different but related schemas, which are derived from different data resources. Meanwhile, our research combines attribute labels extracted from web entity instances to learn a global attribute schema. Second, the schemas used in schema matching are extracted from regular data sources, such as deep web databases, whose quality is high and their vocabularies are relatively small, while our attribute labels are collected from surface web pages, whose quality is not uniformly high and their quantity is relatively large.

3. PROBLEM FORMULATION

The general problem of learning a global attribute schema can be formulated as follows:

Formally, assume we are given a basic description for entities of a specific entity type in M_{init}, which is provided by users under the framework of a predefined entity model M. M_{init} presents users' knowledge about the entities of the target entity type. Then, our goal is to get a global attribute schema for all the web entities of the target entity type, which is denoted as S and is learned based on M_{init} and related web pages.

To describe the problem clearly, we first define a concept "entity" as follows:

Definition 1 An *entity* is something that has a distinct, separate existence, and owns an entity type and all the attributes which are owned by *all* the entites with the same entity type. A web entity is an entity whose instances are

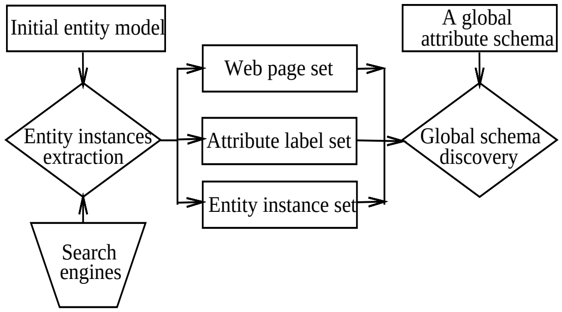


Figure 2: The framework of learning a global attribute schema

presented in web pages. Formally, an *entity* can be represented as $e = \{et, (t_1, v_1), \ldots, (t_i, v_i), \ldots, (t_n, v_n)\}$, where et is the entity type, t_i and v_i is the type and value of ith attribute of e, respectively.

Definition 2 An *entity instance* is a partial container of a specific entity. An *entity instance* contains some attribute labels and their values of the entity. Formally, an *entity instance* can be represented as $inst = \{(l_1, v_1), \ldots, (l_i, v_i), \ldots, (l_n, v_n)\}$, where l_i and v_i is the label and value of ith attribute of $inst$. For example, the two person instances in Figure 1 are both entity instances of the same entity.

Definition 3 An *entity model* is a model which characterizes the features of all the entities of a specific entity type, which include their entity type, the types of their attributes, and the labels describing their attributes. Formally, an *entity model* is represented as $M = \{et, L_1, \ldots, L_i, \ldots, L_n\}$, where et is the entity type, and L_i is a set of labels describing one attribute type.

Definition 4 An *initial entity model* is a special *entity model* provided by users to present the basic description of all the entities of the target entity type. Due to the limitation of users' knowledge, its content only covers a small fraction of the overall knowledge of all the target entities, but it is enough to characterize them.

As a result, an *initial entity model* must contain the entity type and some sets of attribute labels. An *initial entity model* can be formally represented as $M_{init} = \{et, L_1, \ldots, L_j, \ldots, L_n\}$, where for $\forall M_{init}$, $\exists M$, $|M| \geq |M_{init}|$, and for $\forall L_j \in M_{init}$, $\exists L_i \in M$, $L_j \subseteq L_i$.

To learn a global attribute schema, an *attribute schema* for all the entities of a target entity type is defined as follows:

Definition 5 An *attribute schema* for all the entities of a target entity type is a schema about attribute types owned by these entities. It reflects the types of attributes owned by these entities and the labels describing each attribute type. Similarly, a *global attribute schema* for all the entities of a specific entity type is the schema which defines all the main attribute types and labels describing each attribute type. An *attribute schema* can be represented as $S = \{(t_1, L_1), \ldots, (t_i, L_i), \ldots, (t_n, L_n)\}$, where t_i is the ith attribute type and can be initialized by one attribute label in L_i, and L_i is the label set of the ith attribute. Based on the above definitions, our problem can be formulated as follows: Given an initial entity model M_{init} provided by users under an entity model framework M, automatically extract enough entity instances $\{inst_i\}_{i=1}^n$ from the Web, mine attributes labels from $\{inst_i\}_{i=1}^n$ and learn an global attribute schema S using attributes labels through the mining of $\{inst_i\}_{i=1}^n$ and related web pages.

Algorithm 1 The iterative entity instances extraction algorithm

```
Input: an initial entity model $M_{init}$.
Output: an attribute label set $L$, an entity instance set $I$,
    and a related page set $P$.
 1: initialize three empty sets $L$, $I$ and $P$;
 2: create a query set $Q = \{q_i\}_{i=1}^n$ based on $M_{init}$;
 3: for each query $q_i$ in $Q$ do
 4:   issue $q_i$ to a search engine $SE$, get the retrieved page
      set $R_i$;
 5:   for each page $r_{ij}$ in $R_i$ do
 6:     if $r_{ij}$ is a structured page then
 7:       extract entity instances from $r_{ij}$, put them into
          $I$, and put $r_{ij}$ into $P$;
 8:     end if
 9:   end for
10: end for
11: if iteration termination criteria is fulfilled then
12:   put all attribute labels in $I$ into $L$, and terminate the
      algorithm;
13: else
14:   create one-element label set for each attribute label
      in $I$, put all the label sets into $M_{init}$, and go to 2 to
      continue;
15: end if
```

4. ENTITY INSTANCES EXTRACTION

The framework of learning a global attribute schema is illustrated in Figure 2, in which the first step is to iteratively extract enough entity instances that contain sufficient frequent labels to discover the global schema. To achieve this goal, first, we automatically locate related structured pages which contain entity instances. Then, for each page, we extract entity instances. In this step, the emphasis is to determine whether the extracted entity instances are enough to obtain sufficient frequent labels to terminate the extraction process. However, we could not know how many frequent labels are enough in advance. To determine that number, we need to observe whether the number of frequent labels converges after the creation of a certain number of new instances or new iterations.

We propose an iterative algorithm as in Algorithm 1 to automatically locate related structured pages and extract enough entity instances inside. There are three important issues in this algorithm as follows:

1. Query construction

Each query consists of several attribute labels in M_{init}, and the query set created in each iteration does not include the queries created in previous iterations. Moreover, in order to locate structured pages containing entity instances of the target entity type, each query should contain enough labels in M_{init}. If the labels in a query is not enough, unrelated pages will be retrieved. For example, the query {'Name' AND 'Age' AND 'Weight'} will locate many target structured pages which contain person instances, while the query {'Name' AND 'Weight'} will locate many other unrelated pages. Furthermore, to retrieve enough pages to extract enough instances, the labels in one query should not be excessive because the query with excessive labels will match only a few related pages. For instance, the query {'Name' AND 'Gender' AND 'Age' AND 'Birthday' AND 'Birthplace' AND 'Height' AND 'Weight' AND 'Race'} will match fewer pages than the query {'Name' AND 'Gender' AND 'Age' AND 'Birthday'}. Our experimental result indicates that the proper number of labels in one query is 4.

2. Entity instances extraction

After retrieving related pages, we attempt to extract their containing entity instances. For each page, we first identify its data regions with the help of related mining techniques such as [15]. If one or more regions are discovered, which implies that this page might be a structured page with entity instances–then we scan each region to count the label number corresponding to the query in that region. If the ratio between this number and the number of labels in the query reaches a certain threshold (0.7 in our experiments), this region is regarded as a candidate data region containing one entity instance. We then extract this region, mine the pattern of the labels in the query appearing in this region, and utilize the pattern to locate each label with its value.

3. Iteration termination criteria

To get enough frequent labels, the termination criteria of the iterative extraction process is a key point. We assume that the frequency of a frequent label is above a certain threshold, and such labels are sufficient if no new frequent label is discovered after a predefined number (τ) of new instances have been extracted, or a predefined number (σ) of continuous iterations have been performed.

Then, after each iteration, if at least τ (300 in our experiments) new instances are extracted without new labels, we can conclude that the labels are sufficient and iterations can be terminated. And if σ (7 in our experiments) continuous iterations have been performed while the number of new instances are below τ, we consider that the labels are sufficient and iterations should also be terminated. Experimental results in section 6 affirm the effectiveness of our iteration termination criteria.

5. GLOBAL SCHEMA DISCOVERY

Assume the set of frequent labels is $L = \{l_i\}_{i=1}^{n}$, the set of web entity instances is $I = \{inst_j\}_{j=1}^{m}$, and the set of related pages is $P = \{p_k\}_{k=1}^{t}$, our goal is to mine the content of L, I and P to combine labels of the same attribute type together, and, finally, to create the attribute schema S.

To achieve this goal, we first preprocess L using the constraints embedded in I to generate candidate attribute types, and get two sets of label sets C and L'. Each element of C represents a unique candidate attribute type and consists of attribute labels belonging to this attribute type. Each element of L' also consists of attribute labels belonging to the same attribute type, but it does not represent a unique attribute type. In other words, the attribute type represented by each L''s element is independent of the attribute types represented by another elements. Then, based on C and L', we propose a maximum entropy-based method to discover the attribute types hidden in L, and learn the global attribute schema.

5.1 Candidate attribute types generation

The aim of *candidate attribute types generation* is to create two sets C and L' based on L by utilizing the constraints on labels embedded in I. The constraints can be represented as two factors: *labels concurrence factor* and *same entity-value factor*.

5.1.1 Two factors

Labels concurrence factor: This factor is to measure the probability that two labels are of different attribute types. Its main idea is that, it is possible for two labels, which often appear together in entity instances, to be of different attribute types. This is based on the observation that two attribute labels appearing in one entity instances are often used to describe different attribute types. Strictly speaking, when two labels appear in one instance, they should not describe the same attribute type. However, due to the diversity of web entity instances, there are some instances in which two labels are actually of the same attribute type. To measure the probability that two labels have different attribute types, we define the *labels concurrence factor* as follows :

$$C(l_i, l_j) = \frac{|I_i \cap I_j|}{min(|I_i|, |I_j|)} \qquad (1)$$

where l_i and l_j are two labels in L, I_i and I_j are two sets of instances which contain l_i and l_j, respectively.

Same entity-value factor: This factor measures the probability of two labels being of the same attribute type. The rationale behind this factor is that, when two labels often appear with the same value in different instances of the same entity, it is possible for them to be of the same attribute type. This inference is derived from the observation that different web pages often describe the same attribute type in the same entity's instances by using different labels. Then, if two instances are found to belong to the same entity (which will be discussed later), each pair of their labels appearing in them with the same value could be considered to be of the same attribute type. However, this inference may not stand for all situations, because some labels from different instances of the same entity may often contain some meaningless values such as “unknown”, “none”, and “secret”, when these labels are actually not of the same attribute type. Consequently, we define the *same entity-value factor* as follows to measure the probability of two labels l_i and l_j being of the same attribute type:

$$S(l_i, l_j) = \frac{|IP'_{ij}|}{|IP_{ij}|} \qquad (2)$$

where
$IP'_{ij} = \{(inst_{m_i}, inst_{m_j}) | SameEntity(inst_{m_i}, inst_{m_j}) = true, \exists v_{ij}, (l_i, v_{ij}) \in inst_{m_i}, (l_j, v_{ij}) \in inst_{m_j}\}$
and
$IP_{ij} = \{(inst_{m_i}, inst_{m_j}) | SameEntity(inst_{m_i}, inst_{m_j}) = true, \exists v_i, (l_i, v_i) \in inst_{m_i}, \exists v_j, (l_j, v_j) \in inst_{m_j}\}$

$SameEntity(inst_{m_i}, inst_{m_j}) = true$ means $inst_{m_i}$ and $inst_{m_j}$ are actually of the same entity, which is determined by the names and the overlap of attribute values of two instances. The overlap of attribute values can be calculated as

$$O(inst_{m_i}, inst_{m_j}) = \frac{|V_{m_i} \cap V_{m_j}|}{min(|V_{m_i}|, |V_{m_j}|)}$$

where V_{m_i} and V_{m_j} are the attribute value sets of $inst_{m_i}$ and $inst_{m_j}$, respectively. Two instances $inst_{m_i}$ and $inst_{m_j}$ are considered as the same entity's instances if their names are

Algorithm 2 The algorithm of candidate attribute types generation

```
Input: one label set $L$.
Output: two set of label sets $C$ and $L'$.
 1: initialize two empty sets $C$ and $L'$;
 2: group all labels in $L$ which are of the same attribute
    type into one group, and put each group into $L'$;
 3: repeat
 4:   for each element $L_i$ in $L'$ do
 5:     if $L_i$ represents an attribute type different from
        types in $C$ then
 6:       put $L_i$ into $C$, and remove it from $L'$ ;
 7:       terminate the $for$ loop;
 8:     end if
 9:   end for
10: until no new element is put into $C$
```

the same and the overlap of attribute values $O(inst_{m_i}, inst_{m_j})$ is above a certain threshold. This assessment is simple and effective, as illustrated in section 6.

5.1.2 Generating candidate attribute types

Based on the two factors defined above, we can generate candidate attribute types from the label set L using Algorithm 2. In this algorithm, two issues need to be explained:

- Two labels l_i and l_j have the same attribute type if their *same entity-value factor* $S(l_i, l_j)$ is above a certain threshold δ. And this relation can be transferred to more labels. As a result, we can group labels which are of the same attribute type into one set, as illustrated in $2nd$ line in Algorithm 2.

- Two labels l_i and l_j have different attribute types if their *labels concurrence factor* $C(l_i, l_j)$ is above a certain threshold ϵ. Furthermore, for two attribute label sets L_1 and L_2 where labels in each of them have the same attribute type, if one label in L_1 and another in L_2 have different attribute types, all the labels in L_1 have a different attribute type from those of L_2. By this means, we can determine whether L_i represents a new attribute type different from types in C, as in $5th$ line in Algorithm 2.

5.2 Discovering global schema

After generating candidate attribute labels, we get C and L' based on the original label set L. To learn a global attribute schema, first, we initialize the schema S using C, since each element of C represents a unique attribute type. Next, by utilizing our proposed maximum entropy-based method, we find new attribute types represented by some elements of L' and group them with their labels into S, or find the attribute types in S to which some elements of L' belong and group labels in these elements into the corresponding elements of S. The algorithm of global schema discovery is illustrated in Algorithm 3.

The frequency of a label in the above algorithm is the percentage of the extracted entity instances owning the label in instance set I in all the extracted instances. Moreover, the key point of this algorithm is to use the maximum entropy-based method to determine whether a set of labels represents a new attribute type, which will be explained in detail.

Algorithm 3 The algorithm of global schema discovery

```
Input: two sets of label set $C$ and $L'$.
Output: an attribute schema $S$.
initialize an empty schema $S$;
for each element $C_i$ in $C$ do
  select label $l_{ij}$ with maximum frequency from $C_i$ ;
  $t_i \leftarrow l_{ij}$, and put $(t_i, C_i)$ into $S$;
end for
for each element $L_k$ in $L'$ do
  if $L_k$ represents a new attribute type then
    select label $l_{km}$ with maximum frequency from $L_k$;
    $t_k \leftarrow l_{km}$, and put $(t_k, L_k)$ into $S$;
  else
    discover the attribute type $t_l$ in $S$ which $L_k$ belongs
    to, and put all labels in $L_k$ into $t_l$'s label set $L_l$ ;
  end if
end for
```

5.2.1 Maximum entropy-based method

Maximum entropy model is a general-purpose statistical model that can freely incorporate various problem-specific knowledge in terms of features which are not required to be strongly independent. As a result, one can choose arbitrary features to reflect the characteristics of the problem domain as faithfully as possible.

As in our problem, given an attribute schema S, and a label set L_k whose elements are of the same attribute type, our goal is to determine whether labels in L_k are of a new attribute type, or of an existing attribute type in S. And this determination is related to various dependent features, and is proper to be solved using maximum entropy model.

Suppose $L_k = \{l_h\}_{h=1}^{|L_k|}$, for each label $l_h \in L_k$, there are some entity instances which contain l_h in the instances set I. For each instance $inst_j$ which appears in web page w_j and contains l_h, we can calculate a probability of l_h being of an attribute type t_i in S as $p(t_i|c_{hj})$, where c_{hj} is a context of l_h and is composed of some features created based on l_h, $inst_j$ and w_j. Then, for L_k, we can calculate its average probability being of the attribute type t_i as

$$AverP(L_k, t_i) = \frac{\sum_h \frac{\sum_{j=1}^{|I_l|} p(t_i|c_{hj})}{|I_l|}}{|L_k|} \tag{3}$$

where I_l is a subset of I and all instances in I_l contain the label l_h.

If $AverP(L_k, t_i) \geq \mathcal{P}$, where $\mathcal{P}$ is a certain threshold, and L_k can not be determined by *labels concurrence factor* to be of different attribute type from t_i, then labels in L_k are of the attribute type of t_i; Else, labels in L_k are not of the type t_i. Moreover, if labels in L_k are determined to be not of all the types in S, they should be of a new attribute type.

Given the instance $inst_j$ and the web page w_j, the probability of l_h being of the attribute type t_i is

$$p(t_i|c_{hj}) = \frac{p(t_i, c_{hj})}{\sum_i p(t_i, c_{hj})} \tag{4}$$

Under the maximum entropy framework, let t_i be t, c_{hj} be c, $p(t_i, c_{hj})$ can be replaced as $p(t, c)$, which is the joint probability of t and c, and maximizes the entropy $H(p)$.

$$H(p) = -\sum p(t, c) \log p(t, c)$$

Table 1: Feature "templates" on the current "scene" sc_h

Features "templates"	
$l_{h+1} = X$	$\&t_h = T$
$l_{h-1} = X$	$\&t_h = T$
$v_h = X$	$\&t_h = T$
$t_{h+1} = X$	$\&t_h = T$
$t_{h+1} = X$	$\&t_h = T$
$w_{-1} = X$	$\&t_h = T$
$w_1 = X$	$\&t_h = T$

under the following constraints

$$\sum p(t,c) f_j(p,c) = \sum \tilde{p}(t,c) f_j(t,c), 1 \leq j \leq k$$

where $\tilde{p}(t,c)$ is the observed distribution of attribute types and the contexts of attribute labels in training data, and $f_j(t,c)$ is a feature created on the basis of attribute type t and attribute label context c.

The distribution $p(t,c)$ in above constaints is given by

$$p(t,c) = \pi \prod_{j=1}^{k} \alpha_j^{f_j(t,c)}$$

where π is a normalization factor, the α_js are the unknown parameters of the model, and each α_j corresponds to a $f_j(t,c)$ and can be seen as the weight of $f_j(t,c)$. There are several algorithms designed to estimate these unknown parameters, and we select *Limited-Memory Variable Metric* because it has been proved to be especially effective[18].

In our problem, the context of an attribute label is based on the set of labels, values, types, and words surrounding the label in the web page, which can be seen as the "scene" of the label. The "scene" of label l_h in an instance $inst_j$ which appears in a web page w_j is

$$sc_h = \{l_h, l_{h+1}, l_{h-1}, v_h, t_{h+1}, t_{h-1}, w_{-1}, w_1\}$$

where l_{h-1} and l_{h+1} are the labels before and behind l_h in $inst_j$, v_h is the value of l_h, t_{h-1} and t_{h+1} are the type of l_{h-1} and l_{h+1}, and w_{-1} and w_1 are the words before and behind l_h in w_j. If l_h is the first label in $inst_j$, l_{h-1} and t_{h-1} are defined to be special characters "$\mathcal{H}$" and "$\mathcal{T_H}$" ; if l_h is the last label in $inst_j$, l_{h+1} and t_{h+1} are defined to be characters "$\mathcal{E}$" and "$\mathcal{T_E}$". Similary, if l_h is the first word in w_j, w_{-1} is set to "$\mathcal{F_W}$", and if it's the last word, w_1 is set to "$\mathcal{L_W}$" .

Based on the "scene" of l_h, the features can be created by scanning each pair (sc_h, t_h) in the training data with the feature "templates" given in Table 1, where t_h is the attribute type of the label with the "scene" sc_h, and the training data is created using the attribute types in the current schema S. Given sc_h as the current "scene", a feature always asks some yes/no question about sc_h, and constrains t_h to a certain attribute type. The instantiations for the variables X and T in Table 1 are obtained by automatically scanning training data.

For example, for a label "Birth time" named l_h, one feature with scene sc_h might be

$$f_j(sc_h,\, l_h) = \begin{cases} 1 & \text{if } l_{h+1}\text{=``Gender''\& } t_h\text{= ``Birthday'' ;} \\ 0 & \text{otherwise.} \end{cases}$$

Table 2: Four initial entity models for person-type and movie-type entities

Model	Details of model
M_{init0}	$\{person, \{Name\}, \{Age\}, \{Gender, Sex\}, \{Birthplace\}, \{Birthday, Birthtime\}, \{Occupation\}\}$
M_{init1}	$\{person, \{Name\}, \{Age\}, \{Gender\}, \{Weight\}\}$
M_{init2}	$\{movie, \{Title\}, \{Director\}, \{Language\}, \{Region\}\}$
M_{init3}	$\{movie, \{Title\}, \{Director\}, \{Actor\}, \{Genre, Type\}, \{Running\ time\}\}$

For this feature, each entity instance in training set is automatically scanned to initialize it. If l_h is found in one instance with attribute type "Birthday", and "Gender" is the label behind it, $f_j(sc_h,\, l_h)$ is to 1 in this instance; else, it's set to 0.

6. EXPERIMENTS

The proposed iterative entity instances extraction method and global attribute schema learning approach are fully implemented and evaluated on person-type and movie-type entities. The goal of the experimental study are:(i) to check the performance of the iterative entity instances extraction algorithm; (ii) to discover the effectiveness of candidate attribute types generation method, (iii) to evaluate the performance of our global attribute schema learning method.

6.1 Experiment Design

We select *Google* as the search engine to extract entity instances on the Chinese Web. Due to the limitation of Google's Web interface, we expand each query for several times using Recursive Query Expansion (RQE) [7], based on a set of high-frequency words in a standard Chinese Web collection named CWT100G (http://www.cwirf.org).

We select person-type and movie-type entities to evaluate the performance of our technique. For entities of each entity type, we specify two different initial models and guarantee each model is able to characterize them. We list these four initial models in Table 2, where M_{init0} and M_{init1} are models for person-type entities, and the others are for movie-type entities.

To the best of our knowledge, our research is the first study aimed at learning a global attribute schema for all the web entities of a specific entity type. The only relevant study is [8], which also aims at mining a global schema from an attribute label set. While, this study requires high-quality schemas as input and is sensitive to errors in schemas, and its efficiency decreases sharply with the increase of the unique labels' number in schemas. We select the algorithm MGS_{sd} in [8] as the baseline, and implement it to compare the result's quality and the algorithm's efficiency with our method.

6.2 Evaluation Metrics

To evaluate the performance of the iterative entity instances extraction algorithm, we utilize three metrics: *precision*, *entity-type error ratio*, and *attribute-label error ratio*. *Precision* is defined as the percentage of the *correct instances*, which are truly of the specified entity type and all their elements (pairs of label and value) belong to the corresponding entities, in all the extracted instances. *Entity-type error ratio* is the ratio of instances which are not of the

Table 3: Extraction results of the four initial entity models

Model	# iterations	# time (hour)	# instances	# pages	# frequent labels	# unfrequent labels
M_{init0}	6	90	111879	556211	839	9012
M_{init1}	8	110	129954	666742	863	9264
M_{init2}	10	86	107798	257665	406	1849
M_{init3}	9	79	93853	232348	383	1713

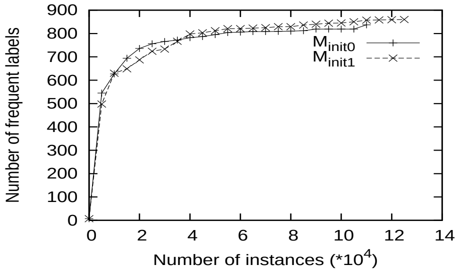


Figure 3: Number of frequent labels for different person-type entity instances

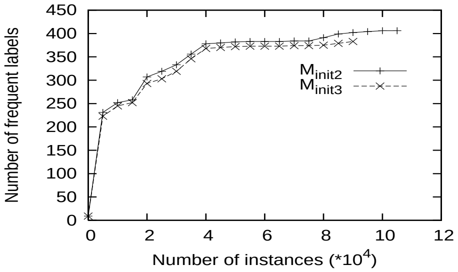


Figure 4: Number of frequent labels for different movie-type entity instances

specified entity type, to all the extracted instances. And *attribute-label error ratio* is the ratio of the extracted instances in which some elements actually do not belong to the corresponding entities, to all the extracted instances. The reason for the *entity-type error* is that the iterative instances extraction process might locate some pages containing other information which is wrongly extracted as the desired entity instances. And the reason for the *attribute-label error* is that the structures of some pages are too flexible to accurately mine the data regions from them.

Actually, the learning of an attribute schema can be seen as the clustering of labels with the aim of grouping labels of the same attribute type into the same cluster. Assume the gold schema is $S_{gold} = \{(t_1, L_1), \ldots, (t_i, L_i), \ldots, (t_n, L_n)\}$, and the learned schema is $S_{learned} = \{(t_1, L_1), \ldots, (t_j, L_j), \ldots, (t_m, L_m)\}$. Then, the labels clustering result corresponding to S_{gold} is $C_{gold} = \{L_1, \ldots, L_i, \ldots, L_n\}$, and labels clustering result corresponding to $S_{learned}$ is $C_{learned} = \{L_1, \ldots, L_i, \ldots, L_m\}$. We utilize *weighted average Fscore* (*waFscore*), which is often used in clustering evaluation[13, 23], as the evaluation metric to measure the quality of the learned schema $S_{learned}$, using the *waFscore* of $C_{learned}$ compared with C_{gold}.

6.3 Experiment Results

6.3.1 Iterative entity instances extraction

The results using different initial models are illustrated in Table 3, in which frequent labels are those whose frequencies are above 0.005. For person-type entities, both models (M_{init0} and M_{init1}) are capable of extracting enough instances. Figure 3 shows the change of the frequent labels' number with the increase of the number of extracted person instances. It shows that for both models, the number of frequent labels increases sharply at the beginning of extraction process, and begin to converge with the progress of extraction process. Moreover, in the 839 frequent labels created by M_{init0}, 826 labels also appear in the 863 labels created by M_{init1}, which indicates that most of the frequent labels have been discovered. In the following experimental results, we select M_{init0}'s result as the working basis for person-type entities.

For movie-type instances, the two models are also capable of extracting enough instances, and the change of the frequent labels' number with the increase of the extracted movie instances' number is illustrated in Figure 4. The same conclusion can be drawn from this figure. Furthermore, it is observed that 376 frequent labels created by M_{init3} also appear in the 406 labels created by M_{init2}, and we select M_{init3}'s result as the working basis for movie-type entities.

Table 4: Evaluation of the extracted entity instances

Entity type	Precision	Entity-type error ratio	Attribute-label error ratio
person-type	93%	1%	6%
movie-type	87.5%	0.5%	12%

To evaluate the quality of the extracted instances, we randomly select 200 instances from the results of M_{init0} and M_{init3}, respectively, and do the evaluation by manually checking them. The result is illustrated in Table 4. As we can see our algorithm could extract instances of specified entity type precisely, at the same time some errors exist for the errors in the mining of data regions.

Moreover, in order to find out why the *attribute-label error ratio* of movie-type entities (12%) is higher than that of person-type entities (6%), we manually check the pages including these instances, and find that movie entity instances often appear in some arbitrary created pages with other movie instances, and the borders between them are not clearly identified by HTML tags, which makes the data region mining technique invalid when processing these pages. While person entity instances often appear in well-structured pages which include the resumes of persons, and data regions can be accurately mined from these pages.

6.3.2 Candidate attribute types generation

In this section, we first evaluate the performance of *labels concurrence factor* and *same entity-value factor*, and then report the generated candidate attribute types.

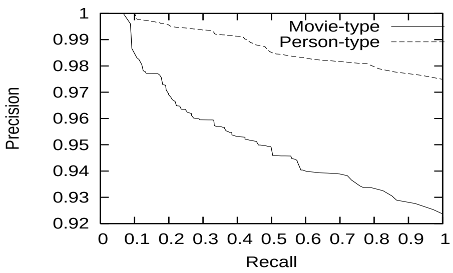


Figure 5: Precision and recall of *labels concurrence factor* for different thresholds

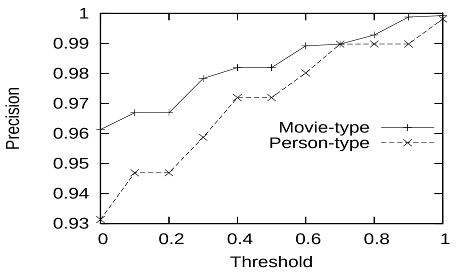


Figure 6: Precision of judgment of same entity for different thresholds

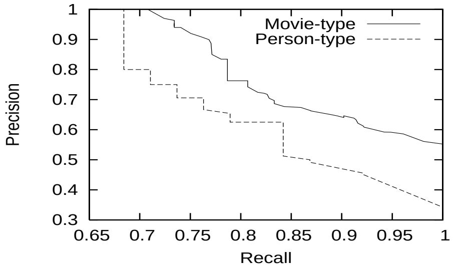


Figure 7: Precision and recall of *same entity-value factor* for different thresholds

Labels concurrence factor: For person-type entities, we randomly select 110 frequent attribute labels, compute *labels concurrence factor* for each pair of them, and get 2672 pairs with nonzero *labels concurrence factors*. Then, we manually evaluate whether labels in these pairs are actually of different types and find 2606 pairs of labels which are of different types. Finally, we utilize our algorithm to automatically judge whether two labels are of different types, by determining whether their *labels concurrence factors* is above a certain threshold. Similarly, we randomly select 55 frequent labels for movie-type entities, and do the same experiment. We get 1286 pairs with nonzero *labels concurrence factors* and find 1201 pairs of labels which are of different types.

Figure 5 shows the relation between the precision and recall of the judgment when using different thresholds. We can see that the precision of judgment for person-type entities is always higher than that of movie-type entities. It is because the *attribute-label error ratios* of person-type entities are lower than those of movie-type entities as illustrated in section 6.3.1, and more pairs of movie-type entities' labels which are not of different types appear together in some instances due to higher *attribute-label error ratios*.

To get precise judgment, we select 0.15 as the best threshold for person-type entities, and 0.18 as the best threshold for movie-type entities.

Same entity-value factor: The basis of this factor is the judgment of whether two instances are of the same entity. For person-type instances, we randomly select 50 person names from extracted instances and make sure each person name has at least 3 different instances, and totally get 176 instances. Then, we calculate *overlaps of attribute values* for all pairs of instances which own the same name. After that, for each pair of instances which own the same name, we manually make the judgment on whether they are of the same entity. We also select 50 movies from 215 movie instances and run the same experiment. Figure 6 shows the precisions of the proposed method in section 5.1.1 for different thresholds. As we can see that this simple method is effective to make right judgments, and we select 0.75 as the best threshold to be used in experiments.

Based on the judgment about two instances being of the same entity, we can calculate *same entity-value factor* for each label pair. If the factor is above a certain threshold, our algorithm can judge that they are of the same type. Figure 7 shows the relation between the precision and recall of the judgment with different thresholds for 110 frequent labels of person instances and 55 frequent labels of movie instances. As we can see that the precision of judgment for movie-type entities is always higher than that of person-type entities. It is because movie-type entity instances include less meaningless attribute values than person-type entity instances.

In order to get precise result, we select 0.84 as the best threshold for person-type entities, and 0.82 as the best threshold for movie-type entities.

Candidate attribute types: For person-type entities, we select the top 400 frequent attribute labels to generate candidate attribute types, because these labels account for 96.9 percent of all the labels' occurrences in the extracted instances. We get 3941 pairs of labels which are of different attribute types and 55 pairs of labels which are of the same attribute types, and generate 19 candidate attribute types. For movie-type entities, we select the top 150 frequent labels since they account for 97.1 percent of all the labels' occurrences, and get 1947 pairs of labels which are of different types and 21 pairs of labels which are of the same types, and generate 7 candidate attribute types.

6.3.3 Global Attribute schema learning method

We select top 400 frequent labels of person-type entities and top 150 frequent labels of movie-type entities as the basic label sets (L_{person} and L_{movie}) to evaluate the performance of our global attribute schema learning method. First, for each label set, three annotators independently create two schemas by discovering the meaning of attribute labels with the help of related entity instances and pages. Then, a final schema for each label set is created by another annotator who is familiar with the meaning of attributes labels based on three schemas. Finally, we get two gold schemas, $S_{glodPerson}$ and $S_{goldMovie}$, and create the gold attribute labels clustering results, $C_{goldPerson}$ and $C_{goldMovie}$.

Comparison with the baseline As mentioned in section 6.1, we select MGS_{sd} in [8] as the baseline, in which

Table 5: Details of four label sets created on L_{person}

Label set	Occurrence ratio	# Created different schemas
Top10	0.5679	809
Top20	0.6824	3388
Top30	0.7235	4594
Top40	0.7527	5202

Table 6: Comparison between our method and baseline for person-type entities

Label set	Method	WaFscore	Running time(s)
Top10	Our method	**1.0**	60.67
	Baseline	1.0	**0.068**
Top20	Our method	**1.0**	96.65
	Baseline	0.7917	**0.248**
Top30	Our method	**1.0**	395.03
	Baseline	0.7527	**3.875**
Top40	Our method	**0.9837**	**631.52**
	Baseline	—	about 1553586

Table 7: Details of four label sets created on L_{movie}

Label set	Occurrence ratio	# Created different schemas
Top10	0.7361	356
Top20	0.8213	1631
Top30	0.8598	2286
Top40	0.8821	2887

Table 8: Comparison between our method and baseline for movie-type entities

Label set	Method	WaFscore	Running time(s)
Top10	Our method	**1.0**	46.24
	Baseline	1.0	**0.038**
Top20	Our method	**1.0**	74.06
	Baseline	0.8333	**0.413**
Top30	Our method	**0.9877**	286.65
	Baseline	0.7444	**0.609**
Top40	Our method	**0.9694**	**547.36**
	Baseline	—	about 51372504

the input is a set of schemas consisting of attribute labels. Moreover, due to the time complexity of the baseline, for L_{person} and L_{movie}, we only select the top 10, 20, 30 and 40 labels from them, and get four label sets. Then, for each created label set L_c, we create schemas by automatically scanning each extracted instance $inst_i$ to create one set of labels which are both in L_c and $inst_i$, and regarding this label set as the schema. Finally, we run MGS_{sd} using these schemas to create the desired attribute schemas, and compare the result's quality and the algorithm's efficiency with our method.

For the baseline, since the hypothesis space is too big (360600 for person-type entities, and 2826021 for movie-type entities) when the number of labels is 40, we only estimate its running time. From the analysis of the baseline's running log, we find that running time is mainly spent on χ^2 estimation. From the theoretical analysis of the algorithm, we discover that the time spent on χ^2 estimation of each hypothesis is the same. As a result, by recording the running time spent on χ^2 estimation for a small number of hypothesizes, we estimate the running time of all the hypothesizes for the baseline.

For person-type entities, the details of the four label sets created on L_{person} are illustrated in Table 5, in which the second column, *occurrence ratio*, is the ratio of the occurrences of labels of the corresponding label set to the occurrences of all the labels in extracted instances, and the third column is the number of different schemas created by this label set. Table 6 shows the comparison of result quality and efficiency between our method and the baseline. As we can see the result's quality of our method is better than the baseline, which is because the baseline is too sensitive to the errors of schemas, while our method utilizes probabilistic method to eliminate errors. Furthermore, when the label set is small, the efficiency of the baseline is higher than our method, while with the increase of the label set's size, the efficiency begins to decline sharply for its exponential time complexity. However, the decline of our method's efficiency is far more smoother than the baseline.

Moreover, the baseline is only effective when the number of labels is less than 40, and become impractical even when processing the Top40 label set, which only accounts for 75.72% of the labels' occurrences and is inadequate to create the global attribute schema. Therefore, the baseline can not create the global attribute schema effectively and efficiently. On the contrary, as the evaluation result illustrated, our method can learn the global attribute schema effectively and efficiently.

We run the same experiment on movie-type entities, and the details of the label sets created on L_{movie} are illustrated in Table 7. The comparison result is illustrated in Table 8, and we can see that our method also outperforms the baseline.

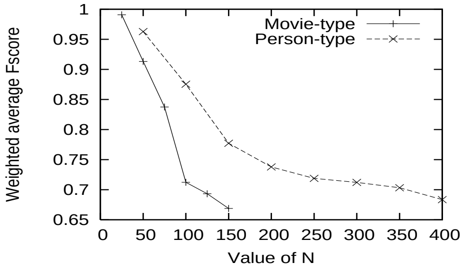


Figure 8: WaFscores on Top$N$ label sets

Performances on different label sets We create top 50, 100, 150, 200, 250, 300, 350 and 400 label sets from L_{person}, and top 25, 50, 75, 100, 125 and 150 label sets from L_{movie}, and compare the results' quality on these different label sets.

The comparison is illustrated in Figure 8. As we can see that our method perform well when the size of label set is relatively small and the performance declines with the growth of the label set. This is because that the labels with lower frequency own less features to characterize their attribute types, while the labels with higher frequency possess more features. However, even for the Top300 label set of person-type entities, which accounts for 95.35% of the labels' occurrences, the *wavFscore* of the created global attribute schema is still acceptable (0.7122), and the time spent on schema creation is 2427.692 seconds (about 40 minutes); for the Top100 label set for movie-type entities, which account for 94.90% of the labels' occurrences, the *waFscore* of the created global attribute schema is also acceptable (0.7123), and the time used to create the schema is 996.862 seconds (about 17 minutes).

7. CONCLUSION

This paper explores the problem of learning a global attribute schema for web entities of a given entity type. This problem is essential to facilitate the integration of entity instances and perform valuable and reasonable web object extraction, and is difficult due to the complexity and variety of web entity instances. We propose a general framework to automatically learn a global attribute schema, and further specialize it to develop an iterative instances extraction algorithm to extract sufficient instances and attribute labels, as well as a maximum entropy-based approach to construct the global attribute schema. We demonstrate our technique on person-type and movie-type entities on the Chinese Web, and create two global attribute schemas for the entities of these two types, and the *weighted average Fscores* for the two created schemas are 0.7122 and 0.7123, respectively. These results validate the efficiency and effectiveness of our technique in learning the global attribute schema for web entities of the specific type.

8. ACKNOWLEDGMENTS

We thank Dr. Tao Meng, Lijiang Chen, Jing He, Shengliang Gao, Mengcheng Duan, Nan Di, Yuan Liu and Bo Peng for their comments. The work in this paper was supported by NSFC Grant 60773162 and 863 Project Grant 2006AA01Z196.

9. REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web.* Morgan Kaufmann San Francisco, 2000.

[2] V. Carchiolo, A. Longheu, and M. Malgeri. Extracting Logical Schema from the Web. *Applied Intelligence*, 18(3):341–355, 2003.

[3] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. *Proc. of IPSJ Conference*, pages 7–18, 1994.

[4] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: discovering complex semantic matches between database schemas. *Proc. of SIGMOD*, pages 383–394, 2004.

[5] H. Do and E. Rahm. COMA-A System for Flexible Combination of Schema Matching Approaches. *Proc. of VLDB*, 2002.

[6] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. *ACM SIGMOD Record*, 30(2):509–520, 2001.

[7] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). *Proc. of WWW*, pages 100–110, 2004.

[8] B. He and K. Chang. Statistical schema matching across web query interfaces. *Proc. of SIGMOD*, pages 217–228, 2003.

[9] B. He and K. Chang. A holistic paradigm for large scale schema matching. *ACM SIGMOD Record*, 33(4):20–25, 2004.

[10] B. He, K. Chang, and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. *Proc. of SIGKDD*, pages 148–157, 2004.

[11] G. Huck, P. Fankhauser, K. Aberer, and E. Neuhold. Jedi: Extracting and Synthesizing Information from the Web. *Proc. of the 3rd IFCIS International Conference on Cooperative Information Systems*, pages 32–43, 1998.

[12] J. Kang and J. Naughton. On schema matching with opaque column names and data values. *Proc. of SIGMOD*, 2003.

[13] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. *Proc. of SIGKDD*, pages 16–22, 1999.

[14] W. Li, C. Clifton, and S. Liu. Database Integration Using Neural Networks: Implementation and Experiences. *Knowledge and Information Systems*, 2(1):73–96, 2000.

[15] B. Liu, R. Grossman, and Y. Zhai. Mining data records in Web pages. *Proc. of SIGMOD*, pages 601–606, 2003.

[16] J. Madhavan, P. Bernstein, and A. Doan. Corpus-Based Schema Matching. *Proc. of ICDE'05*, pages 57–68.

[17] J. Madhavan, P. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. *The VLDB Journal*, pages 49–58, 2001.

[18] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. *Proc. of CoNLL*, pages 1–7, 2002.

[19] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. *Proc. of SIGMOD*, pages 295–306, 1998.

[20] Z. Nie, F. Wu, J. Wen, and W. Ma. Extracting Objects from the Web. *Proc. of ICDE'06*, 2006.

[21] J. Wang, J. Wen, F. Lochovsky, and W. Ma. Instance-based schema matching for web databases by domain-specific query probing. *Very Large Data Bases (VLDB)*, 2004.

[22] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. *Proc. of WWW*, pages 76–85, 2005.

[23] Y. Zhao and G. Karypis. *Evaluation of hierarchical clustering algorithms for document datasets.* ACM Press New York, NY, USA, 2002.

[24] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. Simultaneous record detection and attribute labeling in web data extraction. *Proc. of SIGKDD*, pages 494–503, 2006.