JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 38, 697-711 (2022)
DOI: 10.6688/JISE.202205_38(3).0012

# Discovering Entity Columns of Web Tables Effectively and Efficiently[*]

SI-YU CHEN AND NING WANG[+]
*School of Computer and Information Technology*
*Beijing Jiaotong University*
*Beijing, 100044 P.R. China*
*E-mail: ChenSY725@126.com; nwang@bjtu.edu.cn*[+]

Compared with traditional relational tables, web tables have no designated key attributes or entity columns, which make them difficult for machines to understand. The effectiveness of existing methods for entity column detection usually depends on the coverage of knowledge base, and efficiency of traversing knowledge base is low. In this paper, we propose a novel framework for discovering entity columns in web tables based on approximate primary functional dependency. We build the table schema dependency graph to reflect semantic dependency relationships between columns of a web table. By calculating the importance of each attribute node in the table schema dependency graph iteratively based on LeaderRank, our method can detect entity columns accurately and efficiently for both single-entity tables and multi-entity tables. The experimental results on real web datasets show that our method significantly outperforms previous work in both effectiveness and efficiency, especially for large tables.

***Keywords:*** web table, entity column, functional dependency, table schema dependency graph, LeaderRank

## 1. INTRODUCTION

Nowadays, more and more tabular data has emerged on the Internet [1]. These structured web tables have attracted much attention because they contain a large amount of information. Compared with traditional relational tables, web tables are irregular, uncertain and heterogeneous, which make them difficult for machines to understand their semantics automatically. In order to utilize the rich wealth of web tables, semantic recovery is necessary. Containing the names of real-world entities, the entity column is the most semantically representative attribute column and often serves as key attributes of a web table [2]. Discovering entity columns effectively and efficiently will be greatly helpful for the annotation and understanding of web tables.

P. Venetis proposed a learning-based approach to detect entity columns, which built feature functions and judged entity columns by SVM or binary classifier [2]. K. Braunschweig extended feature set with other features such as Relative Column Positions and Key Indicators to identify entity columns by a classifier [3]. Due to the irregular and heterogeneous characteristics of web tables, it's hard to find the distribution law of entity columns, so the accuracy of above methods cannot be guaranteed. The mainstream approach of detecting entity column is based on knowledge base. J. J. Wang [4] and D. Deng [5] used knowledge base to annotate table name and entity column with assumption that

Received May 18, 2020; revised October 7, 2020 & January 11, 2021; accepted January 26, 2021.
Communicated by Jiun-Long Huang.
[*] This work is supported by the National Key R&D Program of China (2018YFC0809800).
[+] Corresponding author.

there is only one entity column in a table, while N. Wang [6] proposed a framework for identifying multiple entity columns in a web table based on Probase [7]. However, the effectiveness of these methods entirely depends on the coverage of knowledge base, and efficiency of traversing knowledge base is low. Furthermore, these methods are difficult to extend to large-scale web tables because traversing the large knowledge base is time-consuming process especially for large tables.

We propose to identify entity columns based on primary functional dependency [8], which only focuses on the functional dependency with only primary attributes in its determining set. This kind of functional dependency can express the determination relationship between primary attributes and non-primary attributes and are more helpful for entity column detection and topic discovery on web tables. Based on approximate primary functional dependency set, we first build the table schema dependency graph which reflects semantic dependency relationships between columns of a web table. Then we calculate and sort the importance of each column in table schema dependency graph and select semantic intensive columns as entity column candidates. Independent of knowledge base, our method can detect entity columns accurately and efficiently for both single-entity tables and multi-entity tables, and is scalable for large web tables. Our major contributions are summarized as follows:

(1) We are the first to propose an effective and efficient entity column discovery framework based on approximate primary functional dependency.
(2) We propose to build table schema dependency graph and entity column scoring model for a web table, which can be used to find semantically important attribute columns as the final entity columns.
(3) The experimental results show that our method is more effective and efficient than previous work in either single-entity or multi-entity tables, and is more scalable for large web tables.

## 2. RELATED WORK

In big data era, it is very important for machine to understand tables on the web. Early work in table understanding focused on exacting tables from documents and web pages [9-14], but relatively little work is about understanding the semantics and meaning associated with tables. In this section, we briefly explain recent research work on detecting functional dependency and identifying entity columns in tables.

Approximate functional dependency (AFD) is derived from functional dependency (FD) to describe approximate determination relationship between attributes on tables with heterogeneous and noisy data. The discovery of AFDs from web tables are helpful for applications like entity column detection and topic discovery. Early researches only considered AFDs with single attribute on the left-hand side, in which CORDS could automatically discover statistical correlations and soft functional dependencies between columns [15], D. Z. Wang gave counting-based algorithms for deriving AFDs and their probabilities [16], P. Mandros proposed to search AFDs by adopting an information theoretic approach [17]. Furthermore, AFDMiner was proposed for mining approximate primary functional dependency (aPFD) for both single-entity web tables (with only one entity column) and multi-entity web tables (with more than one entity column) [8]. aPFDs focus on dependencies with primary attributes as the determining attributes, which can highlight semantic information and are more helpful for entity column detection on web tables.

Entity columns contain the names of real-world entities and often serve as key attributes of a web table. Without designated key attributes, it is difficult for computers to understand the main topic of a web table and associate a concept in the knowledge taxonomy with it. P. Venetis took the left-most column containing neither numbers nor dates as subject column. They also proposed a learning-based approach to detect entity columns automatically by using five features and binary classifier [2]. Some of features, especially the uniqueness of values and the column index, are specifically tailored to the characteristics of single-concept tables. K. Braunschweig extended the feature set with six additional features and trained a classifier using state-of-the-art techniques to identify potential entity columns [3]. In contrast, J. J. Wang relied on Probase to detect entity columns and make their judgment based on two kinds of evidence, requiring the entity column to contain entities of the same concept and the header to contain attributes that describe entities in the entity column [4]. Above entity column detection techniques assume that there is only one entity column in a table which describes single concept, and neglect tables that describe properties of multiple concepts as well as the relationships between them.

In fact, web tables are usually not normalized and can contain columns describing multiple concepts. K. Braunschweig proposed a normalization approach to decompose multi-concept tables into smaller single-concept tables [18]. The basis of normalization is entity keys. Multiple entity columns were identified in [6] based on knowledge base and concept-attribute dependency. However, the effectiveness of this method depends on the coverage of knowledge base, and it is not scalable to large web tables. In contrast, we propose to discover entity columns based on approximate primary functional dependency. In dependent of knowledge base, we build the table schema dependency graph to reflect semantic dependency relationships between columns of a web table, and detect entity columns effectively and efficiently for web tables.

## 3. PROBLEM MODELING AND SOLUTION OVERVIEW

Our entity column detection method is based on approximate primary functional dependency and table schema dependency graph. We will give problem modeling and solution overview in this section.

### 3.1 Problem Modeling

Approximate functional dependency (AFD) is derived from functional dependency (FD) to describe approximate determination relationship between attributes on tables with heterogeneous and noisy data. For any relation table $r$, $X \overset{\sim}{\rightarrow} Y$ is an AFD if and only if $X \rightarrow Y$ holds for most of the tuples in $r$, where $X$ ($X \subseteq R$) is called the determining set and $Y$ ($Y \in R$) is called the dependent attribute [8]. We focus on approximate primary functional dependency for entity column detection.

**Definition 1 (aPFD):** Let $R$ be a relation, $X$ is a set of primary attributes of $R$ and $Y$ is a non-primary attribute of $R$, if $X \overset{\sim}{\rightarrow} Y$ is an AFD, then it is called an approximate primary functional dependency (aPFD for short), denoted as $\hat{X} \overset{\sim}{\rightarrow} Y$.

An aPFD $\hat{X} \overset{\sim}{\rightarrow} Y$ represents the determination relationship between a set of primary attributes $X$ and a set of non-primary attributes $Y$ in a web table. aPFDMiner is proposed in [8] as a framework to mine approximate primary functional dependency for web tables. By metrics and pruning strategies, aPFDMiner quantifies PFDs' strength denoted as *strength* ($X \overset{\sim}{\rightarrow} Y$) and obtains aPFD set effectively and efficiently. We use $\{(X, Y, strength\ (X \overset{\sim}{\rightarrow} Y))\}$to denote an aPFD set.

Based on an aPFD set for a web table, we are able to build a table schema dependency graph to reflect dependency relationship between columns in a web table. For a table with $N$ attributes, its schema dependency graph will have $N + 1$ nodes, including $N$ attribute nodes and one background node. By the strength of aPFDs in a web table, we can calculate the importance of each attribute node in the table schema dependency graph iteratively and identify entity columns in the table.

**Definition 2 (DV):** For a table with $N$ attributes, a dependency vector (DV for short) is a $N$-dimensional vector for each node in its table schema dependency graph. We can build two types of dependency vector for a web table:

(1) The dependency vector of each attribute node *attr*, denoted as $\overrightarrow{dep}_{attr}$. The $i$th component of $\overrightarrow{dep}_{attr}$ represents the dependency degree that *attr* depends on the $i$th attribute node $v_i$, denoted as $Dep(attr, v_i)$.

(2) The dependency vector of background node BNode, denoted as $\overrightarrow{dep}_{BNode}$. The $i$th component of $\overrightarrow{dep}_{BNode}$ represents the similarity between the $i$th column name and the table name, denoted as $Dep(BNode, v_i)$.

**Definition 3 (TSDG):** For a web table $T$ with $N$ attributes, its table schema dependency graph (TSDG for short) is defined as a rooted weighted directed graph $G = (r, V, E, W)$:

– $r$ is the background node without out-degree, but has directed edges pointing to each attribute node.

– $V$ is a node set $V(G) = \{v_1, v_2, \ldots, v_{N+1}\}$, containing $N$ attribute nodes and one background node $r$.

– $E$ is a set of dependency relationships between nodes in $V$, and $\langle v_i, v_j\rangle \in E$ iff $\hat{v}_i \overset{\sim}{\rightarrow} v_j$ holds on T or $v_i$ is the background node.

– $W$ is a set of weights. For each $\langle v_i, v_j\rangle \in E$, there exists $w\langle v_i, v_j\rangle \in W$, $w\langle v_i, v_j\rangle = Dep(v_i, v_j)$.

A background node is introduced to reflect semantic relationship between table name and its columns. If the similarity between a table name and one of its column name is high, this column might contain more information about table's topic. Intuitively, it is more possible that this column is an entity column.

We will give examples and the explanation of above definitions in Section 4.2 after the details of DV calculation is introduced.

## 3.2 Solution Overview

There is no doubt that entity columns should bring the strongest semantics in web tables. So the problem of detecting entity columns in a web table could be converted into another problem of finding semantic intensive columns within column set of the table. We can settle the problem by assigning a numerical weight to each column in a web table and measuring its relative importance within the column set, just like calculation of leaders in social networks [19].

We propose a novel entity column mining framework ECMiner. After building table schema dependency graph based on dependency vector, ECMiner calculates and sorts the importance of each attribute column by LeaderRank [19, 20], and selects semantic intensive columns as entity column candidates. Fig. 1 gives the framework of ECMiner, which implements entity column detection in three steps.

**DV Calculation.** We use aPFD set from aPFDMiner, semantic similarity between column name and table name to calculate dependency vectors for a web table.

**TSDG Building.** For a web table with $N$ attributes, the corresponding TSDG has $N+1$ nodes including $N$ attribute nodes and one background node. The directed edges in TSDG are introduced based on dependency vectors for attribute nodes and the background node.

**Scoring and Selection.** We calculate the importance of each attribute node in TSDG and select important attribute columns as the final results.

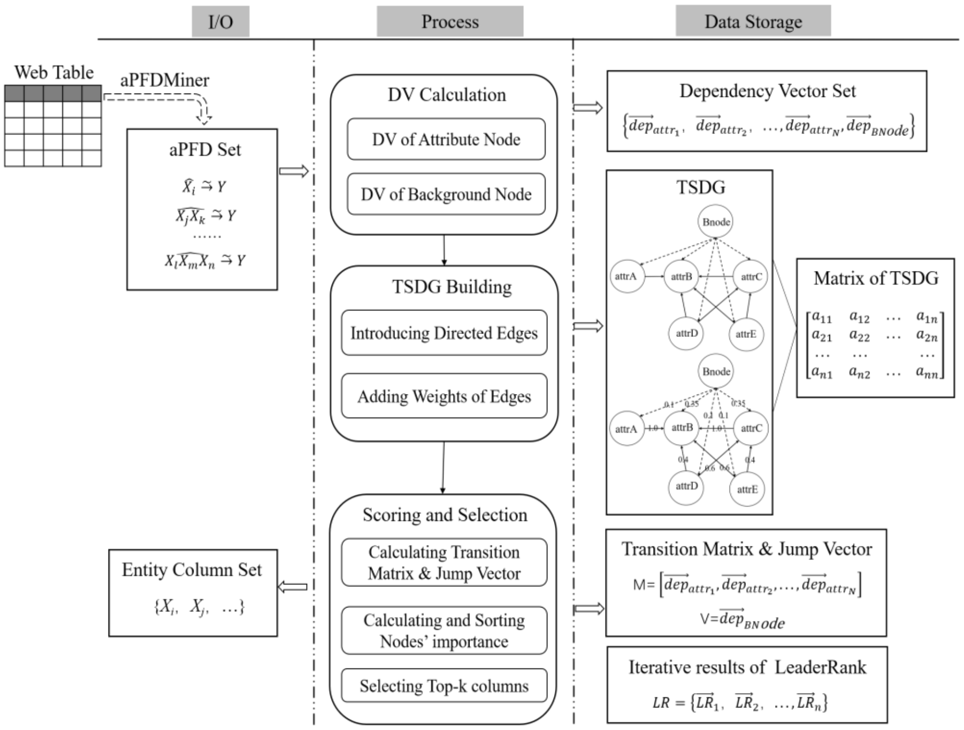

I/O
Process
Data Storage
Web Table
aPFDMiner
aPFD Set
$\widehat{X_i} \rightrightarrows Y$
$\widehat{X_j X_k} \rightrightarrows Y$
……
$X_l \widehat{X_m} X_n \rightrightarrows Y$
DV Calculation
DV of Attribute Node
DV of Background Node
TSDG Building
Introducing Directed Edges
Adding Weights of Edges
Scoring and Selection
Calculating Transition Matrix & Jump Vector
Calculating and Sorting Nodes' importance
Selecting Top-k columns
Entity Column Set
$\{X_i, X_j, \dots\}$
Dependency Vector Set
$\{\overrightarrow{dep}_{attr_1}, \overrightarrow{dep}_{attr_2}, \dots, \overrightarrow{dep}_{attr_N}, \overrightarrow{dep}_{BNode}\}$
TSDG
Bnode
attrA
attrB
attrC
attrD
attrE
Matrix of TSDG
$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$
Transition Matrix & Jump Vector
$M=[\overrightarrow{dep}_{attr_1}, \overrightarrow{dep}_{attr_2}, \dots, \overrightarrow{dep}_{attr_N}]$
$V=\overrightarrow{dep}_{BNode}$
Iterative results of LeaderRank
$LR=\{\overrightarrow{LR}_1, \overrightarrow{LR}_2, \dots, \overrightarrow{LR}_n\}$

Fig. 1. An entity column mining framework for web tables.

## 4. ENTITY COLUMN MINING

We give the details about our entity column mining framework for web tables. Specific problems regarding DV calculation, TSDG building, scoring and selection approaches are examined, and the proposed scheme is refined.

### 4.1 DV Calculation

Before constructing the TSDG for a web table, we should calculate the DVs for each attribute node and the background node first.

**DV of Attribute Node.** For attribute node *Y*, its DV represents the dependency degree that *Y* depends on other attributes. The degree of dependency can be obtained by $strength(\hat{X} \overset{\sim}{\rightarrow} Y)$ from aPFD set. DV $\overrightarrow{dep}_Y$ is calculated by Eq. (1), the *i*th component of which represents the dependency degree that *Y* depends on the *i*th attribute node $X_i$.

$$\overrightarrow{dep}_Y = \left( \frac{Strength(\hat{X}_1 \overset{\sim}{\rightarrow} Y)}{\sum_{i\in(1,N)} Strength(\hat{X}_i \overset{\sim}{\rightarrow} Y)}, \ldots, \frac{Strength(\hat{X}_N \overset{\sim}{\rightarrow} Y)}{\sum_{i\in(1,N)} Strength(\hat{X}_i \overset{\sim}{\rightarrow} Y)} \right) \tag{1}$$

In relational table *r*, the *Strength* of $X \overset{\sim}{\rightarrow} Y$ reflects dependency degree between two attributes. It can be quantified by $Conf(X \overset{\sim}{\rightarrow} Y)$, $InfoGain(X \overset{\sim}{\rightarrow} Y)$ and the proportion of tuples in *r* on which $X \overset{\sim}{\rightarrow} Y$ holds. $Conf(X \overset{\sim}{\rightarrow} Y)$ is introduced to measure the probability that *Y* becomes the dependent attribute of *X*, while $InfoGain(X \overset{\sim}{\rightarrow} Y)$ is defined as the reduction rate of information uncertainty for *Y* under *X*. The details of strength quantification is described in [8].

**DV of Background Node.** The DV of background node $\overrightarrow{dep}_{BNode}$ can be calculated by Eq. (2). The *i*th component of $\overrightarrow{dep}_{BNode}$ is the similarity between the *i*th column name $attr_i$ and the table name *title*. In order to satisfy the condition of LeaderRank [19], we add the average value 1/*N* on each component to ensure that it is not equal to zero. We then normalize the vector and get the final dependency degrees.

$$\overrightarrow{dep}_{BNode} = \left( \frac{Sim(attr_1, title) + \frac{1}{N}}{1 + \sum_{i\in(1,N)} Sim(attr_i, title)}, \ldots, \frac{Sim(attr_N, title) + \frac{1}{N}}{1 + \sum_{i\in(1,N)} Sim(attr_i, title)} \right) \tag{2}$$

We use the fuzzy match to calculate the similarity between table name and column name [21]. If the table name or any column name of the web table does not exist, the each component of $\overrightarrow{dep}_{BNode}$ is equal to the average value 1/*N*.

### 4.2 TSDG Building

Based on DVs for a web table, we can build its TSDG. For a web table with *N* attributes, there are *N* attribute nodes and one background node in its TSDG. After calculating DV of each node, we can introduce the directed edges between attribute nodes with dependency degree not equal to zero. The background node have edges pointing to each attribute node. The weight of each edge can be obtained from the components in DVs of attribute nodes and the background node.

For the web table shown in Table 1, its initial TSDG has 5 attribute nodes and one background node. Assuming that we get a set of approximate primary functional dependency (aPFD) and its strength by aPFDMiner as: $strength(Film \overset{\sim}{\rightarrow} Year) = 1.4$; $strength(Film \overset{\sim}{\rightarrow} Director) = 1.2$; $strength(Film \overset{\sim}{\rightarrow} Nationality) = 0.8$; $strength(Film \overset{\sim}{\rightarrow} Award)$ = 0.9; $strength(Director \overset{\sim}{\rightarrow} Nationality) = 1.2$; $strength(Director \overset{\sim}{\rightarrow} Award) = 0.6$; then aPFD = {(Film, Year, 1.4), (Film, Director, 1.2), (Film, Nationality, 0.8), (Film, Award, 0.9), (Director, Nationality, 1.2), (Director, Award, 0.6)}. Furthermore, if we use *title* to represent the name of Table 1 and assume that *Sim*(*Film*, *title*) = *Sim*(*Director*, *title*) = 0.5, *Sim*(*Year*, *title*) = *Sim*(*Nationality*, *title*) = *Sim*(*Award*, *title*) = 0, we can figure out DVs according to formula (1) and Eq. (2) as: $\overrightarrow{dep}_{Bnode}$ = (0.1, 0.35, 0.35, 0.1, 0.1), $\overrightarrow{dep}_{Year}$ = (0.1, 0, 0, 0), $\overrightarrow{dep}_{Film}$ = (0, 0, 0, 0, 0), $\overrightarrow{dep}_{Director}$ = (0, 1, 0, 0, 0), $\overrightarrow{dep}_{Nationality}$ = (0, 0.4, 0.6, 0, 0), $\overrightarrow{dep}_{Award}$ = (0, 0.6, 0,4, 0, 0). During calculation of DV of attribute node, we regard $strength(X \overset{\sim}{\rightarrow} Y)$ as 0 if $X \overset{\sim}{\rightarrow} Y$ is not in the aPFD set. After that, we introduce into the TSDG 6 directed edges between attribute nodes and 5 directed edges from the background node to each attribute node. And finally, we add weight for each edge based on dependency degree between nodes and create a final TSDG as in Fig. 2 (a).

**Table 1. An example for web table.**

| | Year | Film | Director | Nationality | Award |
|---|---|---|---|---|---|
| 1 | 1971 | Love | Karoly Makk | Hungary | Jury Prize |
| 2 | 1996 | Crash | David Cronenberg | Canada | Special Prize |
| 3 | 1997 | Western | Manuel Poirier | France | Jury Prize |
| 4 | 2007 | Silent Light | Carlos Reygadas | Mexico | Jury Prize |
| 5 | 2006 | Red Road | Andrea Arnold | UK | Jury Prize |

## 4.3 Scoring and Selection

Inspired by LeaderRank [19], an improved algorithm for sorting web pages based on PageRank, our ECMiner settles the problem of detecting entity columns of a web table by calculating and sorting the weight of each node in a TSDG, then selecting the semantic intensive attribute columns as the final result. ECMiner iteratively calculates the weight $\overrightarrow{LR}_i$ of each node in the TSDG as Eq. (3).

$$\overrightarrow{LR}_i = (1-d)*\mathrm{V} + d*\mathrm{M}*\mathrm{V}$$
$$= \begin{cases} = (1-d)\overrightarrow{dep}_{BNode} + d\left[\overrightarrow{dep}_{attr1}, \overrightarrow{dep}_{attr2}, \ldots, \overrightarrow{dep}_{attrN}\right]\overrightarrow{dep}_{BNode}, & i = 1 \\ = (1-d)\dfrac{\overrightarrow{LR}_{i-1}}{\left|\overrightarrow{LR}_{i-1}\right|} + d\left[\overrightarrow{dep}_{attr1}, \overrightarrow{dep}_{attr2}, \ldots, \overrightarrow{dep}_{attrN}\right]\dfrac{\overrightarrow{LR}_{i-1}}{\left|\overrightarrow{LR}_{i-1}\right|}, & i > 1 \end{cases} \tag{3}$$

*d* represents damping coefficient (usually 0.8), indicating that a part of weight of the node is not used for transition. *M* is the transition matrix, calculated by the DVs of attribute nodes as $M = [\overrightarrow{dep}_{attr_1}, \overrightarrow{dep}_{attr_2}, \ldots, \overrightarrow{dep}_{attr_N}]$, ensuring the weight of each node can be emphasized reasonably. *V* is a jump vector with the initial value $V = \overrightarrow{dep}_{BNode}$ to effectively avoid the hanging nodes and shorten the process of iteration.

After several iterations, ECMiner will reach convergence and the weights of the nodes are stable. We sort the components in the final *LR*, and select nodes with weight greater than some threshold as the final entity columns.

Algorithm 1 describes the process of entity column detection. The input is a transition matrix *M* and a jump vector *V*. At the beginning, the initial weight vector *LR* is calculated (line 1). Then, *LR* is calculated iteratively according to Eq. (3) until convergence is reached (line 2-12). At last, if *LR* value of column *k* reaches some threshold *c*, the column could be regarded as an entity column in current table (13-20).

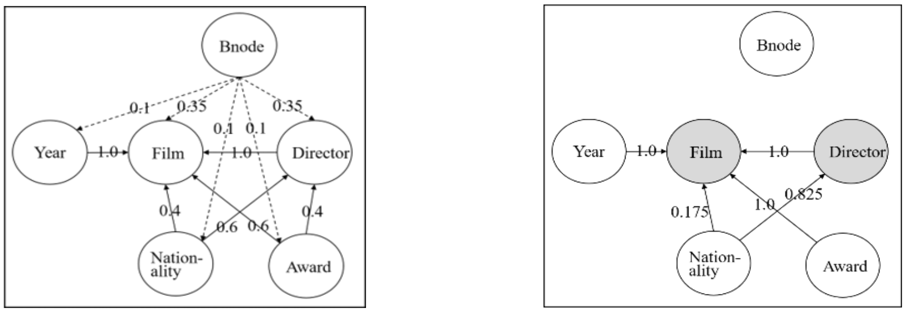
**Algorithm 1:** Entity Column Detection

```
Input: the transition matrix M    //the DVs of attribute nodes as M = [$\overrightarrow{dep}_{attr_1}, \overrightarrow{dep}_{attr_2}, \dots, \overrightarrow{dep}_{attr_N}$]
       the jump vector V          //the DV of background node $V = \overrightarrow{dep}_{BNode}$
Output: a list of entity columns Entityarr
1.  $LR_1 = (1 - d)*V + d*M*V$;
2.  $LR_{curr} = LR_1$;
3.  done = false;
4.  $i = 2$;
5.  while (!done)
6.          done = true;
7.          calculate $LR_i$ using Eq. (3);
8.          for $k = 1$ to $N$ do
9.                  if ($|LR_i[k] - LR_{curr}[k]| / LR_{curr}[k] > \Delta$) then done = false;
10.         $LR_{curr} = LR_i$;
11.         $i = i+1$;
12.  end while
13.  $j = 1$;
14.  for $k = 1$ to $N$ do
15.        if ($LR_{curr}[k] > c$) then
16.             Entityarr[$j$] = $k$;
17.             $j = j + 1$;
18.        endif;
19.  Entityarr = SortbyLR (Entityarr)
20.  return Entityarr;
```

For the TSDG shown in Fig. 2 (a), we can calculate the transition matrix *M* and jump vector *V* using Eqs. (1) and (2), and get results as follows.

$$M = [\overrightarrow{dep}_{attr_1}, \overrightarrow{dep}_{attr_2}, \dots, \overrightarrow{dep}_{attr_N}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0.4 & 0.6 \\ 0 & 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad V = \overrightarrow{dep}_{BNode} = \begin{bmatrix} 0.1 \\ 0.35 \\ 0.35 \\ 0.1 \\ 0.1 \end{bmatrix}.$$

After several iterations in Algorithm 1, the weights of attribute nodes are stable, and the final $\overrightarrow{LR}$ = {0, 0.79, 0.21, 0, 0}, so we can mark “Film” and “Director” as the entity columns of the web table in Fig. 2 (b).



(a) The table schema dependency graph. (b) The TSDG after iteration.
Fig. 2. The TSDG and its convergence result of Table 1.

## 5. PERFORMANCE EVALUATION

This section verifies the performance of the proposed entity column detection method by experiments on two true web table datasets and comparison of the performance with existing methods.

### 5.1 Experimental Environment

In order to evaluate the effectiveness and efficiency of our entity column mining framework for web tables, we implemented our algorithms in java and used two true web table datasets (the WikiTables Dataset 2017 (WIKI) [22] and the WebDataCommons Web Table Corpus 2015 (WDC) [23]) to compare our ECMiner with exsiting entity column discovery algorithms EDModel [4] and PGModel [6]. We extracted and selected 200 tables with regular table structures and meaningful contents as experimental data. The characteristics of 3 comparison method are described as follows.

**EDModel:** relies on Probase to detect entity columns and make the judgement based on two kinds of evidence, which requires the entity column to contain entities of the same concept and the header to contain attributes that describe entities in the entity column. EDModel assumes that there is only one entity column in a table which describes single concept, and neglect tables that describe properties of multiple concepts as well as the relationships between them.

**PGModel:** is the first work for identifying multiple entity columns in a web table based on concept-attribute dependency in knowledge base. The effectiveness of PGModel depends on the coverage of knowledge base, and it is not scalable to large web tables.

**ECMiner:** discover entity columns based on aPFDs instead of knowledge base. Highlighting semantic information by aPFDs, ECMiner can detect entity columns accurately and efficiently for both single-entity tables and multi-entity tables, especially for large tables.

According to the size of the data set and the number of entity columns, we divide the original data set into a LS data set (Large-size tables with Single entity column), a LM data set (Large-size tables with Multiple entity columns), an MS data set (Medium-size tables with Single entity column), an MM data set (Medium-size tables with Multiple entity columns), an SS data set (Small-size tables with Single entity column), an SM data set (Small-size tables with Multiple entity columns). We compare ECMiner with EDModel and PGModel in precision, coverage, F-measure, and runtime on above 6 datasets. The characteristic statistics of single-entity and multi-entity table datasets are given as Table 2 and Table 3 respectively.

**Table 2. Characteristic statistics of single-entity table dataset.**

| Dataset | | #Col | #Row |
|---|---|---|---|
| SS | Min | 2 | 24 |
| | Max | 4 | 52 |
| | Avg | 3 | 38 |
| MS | Min | 4 | 56 |
| | Max | 7 | 120 |
| | Avg | 5 | 76 |
| LS | Min | 6 | 120 |
| | Max | 11 | 282 |
| | Avg | 8 | 156 |

**Table 3. Characteristic statistics of multi-entity table dataset.**

| Dataset | | #Col | #Row |
|---|---|---|---|
| SM | Min | 4 | 12 |
| | Max | 6 | 54 |
| | Avg | 5 | 36 |
| MM | Min | 6 | 56 |
| | Max | 9 | 144 |
| | Avg | 7 | 89 |
| LM | Min | 8 | 152 |
| | Max | 12 | 470 |
| | Avg | 10 | 183 |

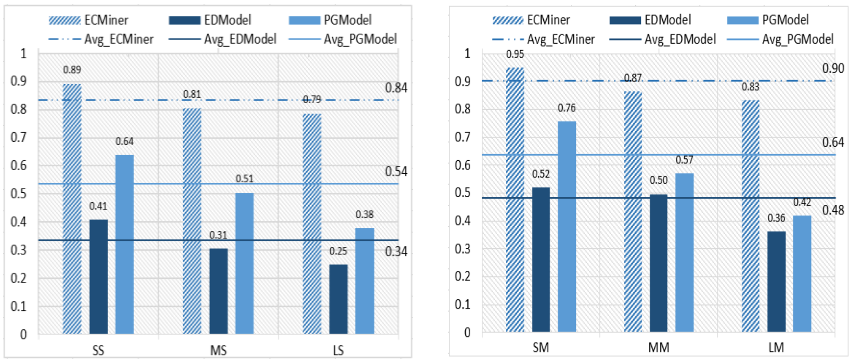
## 5.2 Evaluation of Effectiveness

We use precision, recall and F-measure to evaluate the effectiveness of entity column detection methods. Precision measures the percentage of the output entity columns that are desired. Compared with accuracy which measures the percentage of true positive and true negative in all samples, precision metric is more targeted to the effectiveness of entity column detection. In addition, we use recall to measure the percentage of the desired entity columns that are output, and F-measure is the weighted harmonic mean of precision and recall.

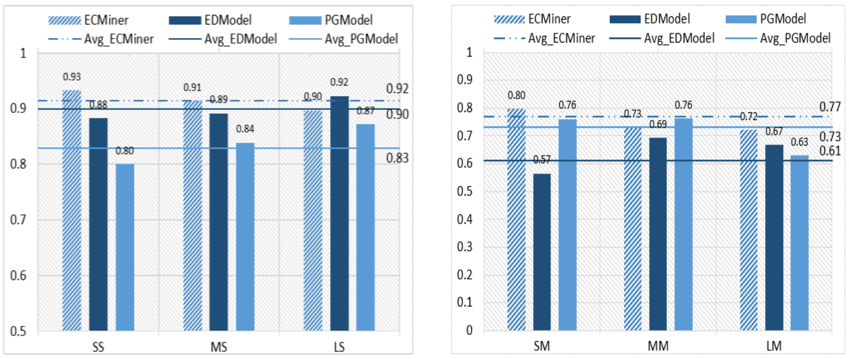
$$Precision = \frac{|values\_truth \cup values\_found|}{|values\_found|} \tag{4}$$

$$Coverage = \frac{|values\_truth \cap values\_found|}{|values\_truth|} \tag{5}$$

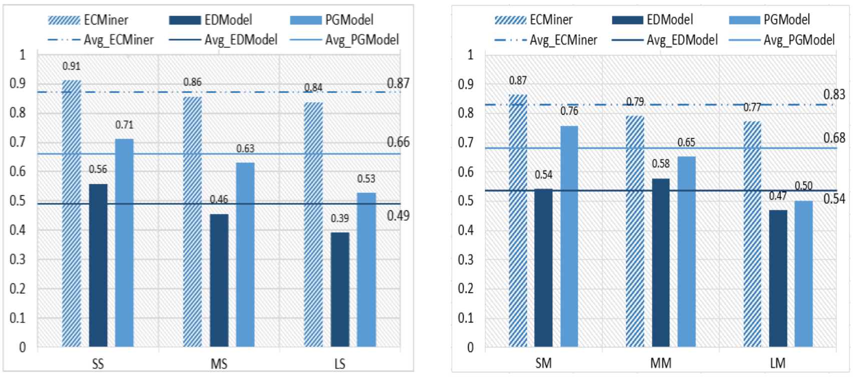
$$F-measure = \frac{2 * Coverage * Precision}{Coverage + Precision} \tag{6}$$



(a) Single-entity tables. (b) Multi-entity tables.
Fig. 3. Comparison of different methods in precision.



(a) Single-entity tables. (b) Multi-entity tables.
Fig. 4. Comparison of different methods in coverage.



(a) Single-entity tables. (b) Multi-entity tables.
Fig. 5. Comparison of different methods in F-measure.

Figs. 3-5 show the performance of different algorithms on single-entity tables and multi-entity tables respectively in precision, coverage, F-measure. We have the following observations:

1. For precision, ECMiner significantly outperforms EDModel and PGModel for both single-entity and multi-entity tables, especially on large dataset. As shown in Fig.3, the average precision of our ECMiner is 0.84 and 0.90 for single-entity and multi-entity tables, respectively, greatly exceeding EDModel (0.34 and 0.48) and PGModel (0.54 and 0.64). Table 4 specially give growth rates between ECMiner and other baseline methods on datasets with different size (Small, Medium or Large). The precision of EDModel and PGModel entirely depend on the coverage of the knowledge base for attribute values in web tables. Not being restricted by the knowledge base and the number of entity columns, ECMiner gets higher precision because it is based on aPFDs which focus on dependencies with primary attributes as determining attributes, and it considers the semantic information of the table name and column name. In some cases, especially when the quality of web tables is not high, ECMiner may focus on the wrong node during iteration. It is our future work to improve our method to make it robust to low-quality web tables.

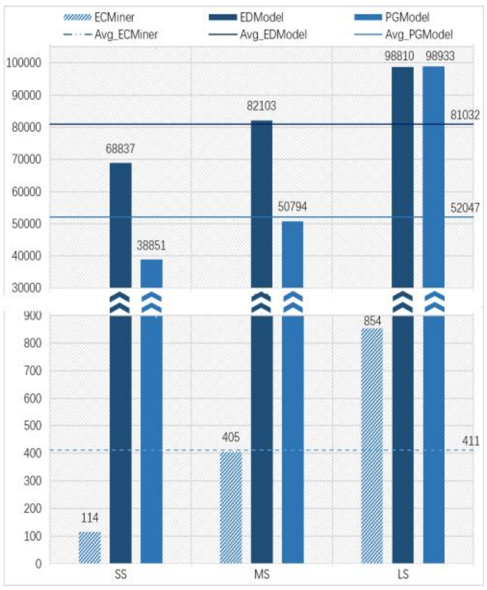
**Table 4. Precision growth between ECMiner and other methods.**

| | | Small | Medium | Large | Average |
|---|---|---|---|---|---|
| EDModel | Single-entity | 117% | 161% | 216% | 147% |
| | Multi-entity | 83% | 74% | 131% | 87% |
| PGModel | Single-entity | 39% | 59% | 108% | 56% |
| | Multi-entity | 25% | 53% | 98% | 41% |

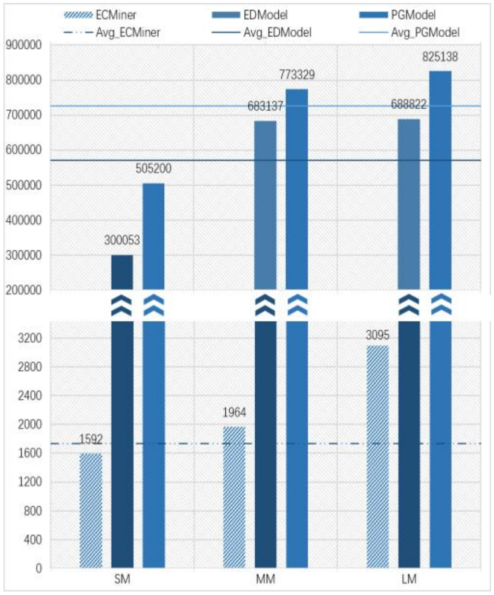
2. As the sizes of web tables increase from Small to Large, the precisions of the three methods gradually decrease. Among them, the precision of ECMiner reduces 11% on single-entity tables and 16% on multi-entity tables, while EDModel reduces 39% and 16%, and PGModel reduces 41% and 45%. Obviously, ECMiner has a smaller decline in precision than other methods for large tables. The precision of ECMiner is more stable on larger tables because each column is always abstracted as one attribute node in ECMiner no matter how many attribute values the column have. But EDModel and PGModel match more noisy labels when the number of attribute value increases.
3. As shown in Fig. 4, ECMiner also performs better in coverage with average 0.92 on single-entity tables and 0.77 on multi-entity tables than the baseline methods. Taking advantage of aPFDs between columns, ECMiner uses the LeaderRank to highlight the importance of entity columns. Furthermore, ECMiner is not sensitive to the number of entity columns and is less likely to miss entity columns, so its average coverage is higher. For large datasets with more attributes in tables, the opportunity for iteration algorithm to focus on the wrong node will increase. From the experimental results, we can see that the coverage rate does not grow while increasing data size.
4. As the precision and coverage are higher, the F-measure of our ECMiner is also better than baseline methods. In Fig. 5, the average F-measure of ECMiner reaches 0.87 and 0.83 for single-entity and multi-entity tables, respectively, significantly outperforming EDModel (0.49 and 0.54) and PGModel (0.66 and 0.68).

## 5.3 Evaluation of Efficiency

Fig. 6 gives the comparison of different methods in runtime. The efficiency of our ECMiner significantly outperforms other baseline methods, on either single-entity tables or multi-entity tables. The average time cost of ECMiner is 0.5% of EDModel and 0.8% of PGModel for single-entity tables, while it is only 0.3% of EDModel and 0.2% of PGModel for multi-entity tables. As we know, EDModel and PGModel need to traverse the large knowledge base to detect entity columns, which is time-consuming process especially for large tables. However, our ECMiner is based on LeaderRank with good convergence and efficiency.



(a) Single-entity tables.



(b) Multi-entity tables.

Fig. 6. Comparison of different methods in runtime.

## 6. CONCLUSIONS

Discovery of entity columns will greatly promote the understanding of web tables. We propose an entity column discovery framework ECMiner based on approximate primary functional dependency. After building table schema dependency graph based on dependency vector to reflect semantic dependency relationships between columns of a web table, ECMiner calculates and sorts the importance of each attribute column by LeaderRank, and selects semantic intensive columns as entity column candidates. Our method can detect entity columns accurately and efficiently for both single-entity tables and multi-entity tables. The experimental results on real web datasets show that our method significantly outperforms previous work in both effectiveness and efficiency, especially for large tables.

## REFERENCES

1. M. J. Cafarella, A. Halevy, H. Lee, J. Madhavan, C. Yu, D. Wang, and E. Wu, "Ten years of WebTables," in *Proceedings of the VLDB Endowment*, Vol. 11, 2018, pp. 2140-2149.
2. P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. X. Miao, and C. Wu, "Recovering semantics of tables on the web," in *Proceedings of the VLDB Endowment*, Vol. 4, 2011, pp. 528-538.
3. K. Braunschweig, "Recovering the semantics of tabular web data," Ph.D. Thesis, Dresden University of Technology, 2015.
4. J. J. Wang, H. X. Wang, Z. Y. Wang, and K. Q. Zhu, "Understanding tables on the web," in *Proceedings of the 31st International Conference Conceptual Modeling*, 2012, pp. 141-155.
5. D. Deng, Y. Jiang, G. L. Li, J. Li, and C. Yu, "Scalable column concept determination for web tables using large knowledge bases," in *Proceedings of the VLDB Endowment*, Vol. 6, 2013, pp. 1606-1617.
6. N. Wang and X. R. Ren, "Identifying multiple entity columns in web tables," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 28, 2018, pp. 287-309.
7. W. T. Wu, H. S. Li, H. X. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," in *Proceedings of SIGMOD Conference*, 2012, pp. 481-492.
8. S. Y. Chen, N. Wang, and M. M. Zhang, "Mining approximate primary functional dependency on web tables," *IEICE Transactions on Information and Systems*, Vol. E102-D, Vol. 3, 2019, pp. 650-654.
9. M. D. Adelfio and H. Samet, "Schema extraction for tabular data on the web," in *Proceedings of the VLDB Endowment*, Vol. 6, 2013, pp. 421-432.
10. H. H. Chen, S. C. Tsai, and J. H. Tsai, "Mining tables from large scale HTML texts," in *Proceedings of Conference on Computational Linguistics*, 2000, pp. 166-172.
11. W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from web tables," in *Proceedings of WWW Conference*, 2007, pp. 71-80.
12. D. Jannach, K. Shchekotykhin, and G. Friedrich, "Automated ontology instantiation from tabular web sources – the AllRight system," *Web Semantics*, Vol. 7, 2009, pp. 136-153.
13. D. Pinto, A. McCallum, X. Wei, and W. B. Croft, "Table extraction using conditional random fields," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003, pp. 235-242.
14. Y. Wang and J. Hu, "A machine learning based approach for table detection on the web," in *Proceedings of WWW Conference*, 2002, pp. 242-250.
15. I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga, "CORDS: automatic discovery of correlations and soft functional dependencies," in *Proceedings of SIGMOD Conference*, 2004, pp. 647-658.
16. D. Z. Wang, L. Dong, and A. D. Sarma, "Functional dependency generation and applications in pay-as-you-go data integration systems," in *Proceedings of the 12th International Workshop on Web and Databases*, 2009.

17. P. Mandros, M. Boley, and J. Vreeken, “Discovering reliable approximate functional dependencies,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 355-363.
18. K. Braunschweig, M. Thiele, and W. Lehner, “From web tables to concepts: a semantic normalization approach,” in *Proceedings of the 34th International Conference on Conceptual Modeling*, 2015, pp. 247-260.
19. L. Y. Lü, Y. C. Zhang, C. H. Yeung, and T. Zhou, “Leaders in social networks, the delicious case,” *PLoS ONE*, Vol. 6, 2011.
20. S. Xu and P. Wang, “Identifying important nodes by adaptive LeaderRank,” *Physic A: Statistical Mechanics & its Applications*, Vol. 469, 2017, pp. 654-664.
21. W. W. Cohen, P. Ravikumar, and S. E. Fienberg, “A comparison of string distance metrics for name-matching tasks,” in *Proceedings of International Conference on Information Integration on the Web*, 2003, pp. 73-78.
22. http://downey-n1.cs.northwestern.edu/public/.
23. http://www.webdatacommons.org/webtables/index.html.

**Si-Yu Chen (陈思宇)** received her M.S. degree in School of Computer and Information Technology, Beijing Jiaotong University, China. She currently works in China CITIC Bank. Her research interests include web data integration, data quality and data mining.

**Ning Wang (王宁)** received her Ph.D. degree in Computer Science in 1998 from Southeast University in Nanjing, China. She is currently serving as a Professor in School of Computer and Information Technology, Beijing Jiaotong University, China. Her research interests include web data integration, big data management, data quality and crowdsourcing.