Computing and Informatics, Vol. 38, 2019, 1039–1066, doi: 10.31577/cai_2019_5_1039

GATHERING INFORMATION ON THE WEB BY CONSISTENT ENTITY AUGMENTATION

Weijuan Sun, Ning Wang*

School of Computer and Information Technology
Beijing Jiaotong University
Beijing, 100044, P.R. China
e-mail: {15120437, nwang@}bjtu.edu.cn

Abstract. Users usually want to gather information about what they are interested in, which could be achieved by entity augmentation using a vast amount of web tables. Existing techniques assume that web tables are entity-attribute binary tables. As for tables having multiple columns to be augmented, they will be split into several entity-attribute binary relations, which would cause semantic fragmentation. Furthermore, the result table consolidated by binary relations will suffer from entity inconsistency and low precision. The objective of our research is to return a consistent result table for entity augmentation when given a set of entities and attribute names. In this paper we propose a web information gathering framework based on consistent entity augmentation. To ensure high consistency and precision of the result table we propose that answer tables for building result table should have consistent matching relationships with each other. Instead of splitting tables into pieces we regard web tables as nodes and consistent matching relationships as edges to make a consistent clique and expand it until its coverage for augmentation query reaches certain threshold γ. It is proved in this paper that a consistent result table could be built by considering tables in consistent clique to be answer tables. We tested our method on four real-life datasets, compared it with different answer table selection methods and state-of-the-art entity augmentation technique based on table fragmentation as well. The results of a comprehensive set of experiments indicate that our entity augmentation framework is more effective than the existing method in getting consistent entity augmentation results with high accuracy and reliability.

Keywords: Web table, data integration, entity augmentation, consistency

* Corresponding author

1 INTRODUCTION

In recent years, a vast amount of structured data in web pages attracts more and more attention. Google presented WEBTABLES system in 2008 [1, 2], which crawled 14.1 billion HTML tables containing 154 million relational data. Now, we can also get web tables from some commercial table search engines, such as Google Tables, Google Fusion Tables and Microsoft's Excel PowerQuery [3, 4, 5, 6]. Web tables containing relational information could be used to integrate data, one application of which is entity augmentation. Our paper focuses on entity augmentation using web tables. In other words, given a set of entities, a set of names of attributes, we regard web tables as data source to return the value of attributes for each entity.

For entity augmentation, Yakout et al. implemented InfoGather to augment entities by holistic matching [7]. They only considered binary query table with single attribute to be extended by making assumption that web tables are entity-attribute binary (EAB) relations. As for n-ary query tables, they *split* the table into several EAB relations, i.e., the subject column with each of the other columns comprise a set of EAB relations. Under this strategy, attributes in a web table are thought to be independent. Other entity augmentation systems, such as SearchJoin [8, 9, 10], use the same methods, getting result table by consolidating multiple entity-attribute binary tables.

The most significant problem for current entity augmentation methods is entity inconsistency. For example, Figure 1 gives an example of entity augmentation by InfoGather, in which augmentation for only one attribute at a time could be performed. At the beginning, the second column of t_1 is selected to augment the value of column *author* in query table according to schema matching. Then, the third column of t_2 is selected to augment the value of column *year* in query table. The topic of t_1 is about book, while the topic of t_2 is about film. Consolidated by t_1 and t_2, the result table shown in Figure 2 suffers from entity inconsistency because values for *author* and *year* in the same row are not from the same entity.

The quality of the entity augmentation result could be evaluated by several aspects such as consistency, coverage and precision. By observation, we find some limitations in existing technology that leads to low coverage, low precision and low consistency.

First, existing methods usually perform entity augmentation for one attribute at a time by splitting an n-ary web table into several binary entity-attribute tables. As we discussed above, this kind of method will cause entity inconsistency and then lead to low precision.

Second, web tables usually miss header text. For example, in Figure 1, t_3 misses the label of the second attribute. When we search web tables to match the query table Q, t_3 could not be found because its attributes do not overlap Q's attributes. In this circumstance, low coverage is caused by losing some matching values.

Third, a web table is usually regarded to be matched with a query table when they both have the same entities and the same attributes. But if we only consider schema level features to find matching tables, there may be semantic conflicts be-

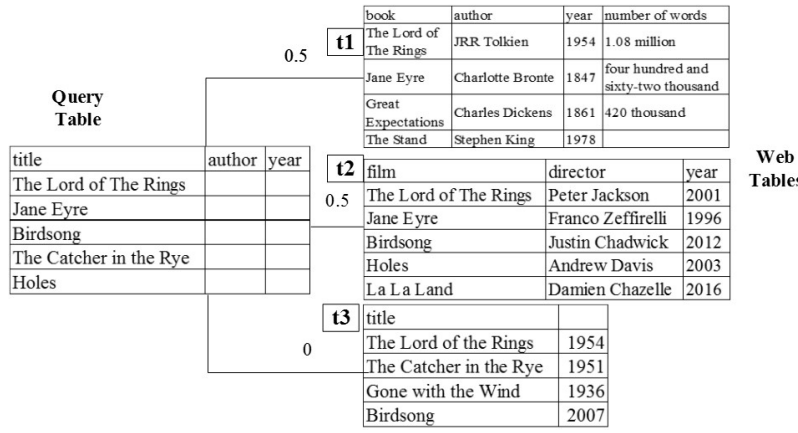

Figure 1. Example of entity augmentation using web tables

title	author	year
The Lord of The Rings	JRR Tolkien	2001
Jane Eyre	Charlotte Bronte	1996
Birdsong		2012
The Catcher in the Rye		
Holes		

Figure 2. The result table

tween two tables. For example, in Figure 1, t_2 is a matching table for the query table when only considering matches of entities and attribute names. However, entity inconsistency occurs because the query table is about books and t_2 is about films. This situation will also lead to low precision.

Entity inconsistency occurs when the semantics of web tables is ignored. For n-ary query table, we should regard all the attributes as a whole, instead of splitting the attributes into pieces. In the following paragraphs, we will make a distinction between an answer table and a result table. After entity augmentation, a query table Q corresponds to one result table, which contains value of attributes to be extended. In order to construct a result table, we should find a set of answer tables which provide attribute values for the final result table. Consistent entities could be obtained when answer tables match with each other either in semantics or in value, which will later be defined as consistent matching relationship in Section 2. Given a set of web tables T and a query table Q with entities, the process of entity augmentation could be thought as continuing to find answer tables from T and augmenting entities until coverage of the result table for Q reaches a specific threshold γ. If we take web tables as nodes and consistent matching relationships as edges, answer tables for a query table make up a clique because those answer tables should be matched with each other to make up consistent matching relationship.

Entity augmentation problem could be converted into another problem of building clique in a graph.

At the beginning, we find seed tables which are the most related with the query table to build initial cliques, each of which has only one node (seed table). To get a result table with a specific coverage γ, we have to expand the initial clique with more nodes (web tables) to form a clique satisfying the requirement of coverage, which is later defined as Consistent γ-Coverage Clique in Section 2. When a consistent γ-coverage clique is built successfully, the nodes in this clique are answer tables for the result table. At last, we will get a set of consistent γ-coverage cliques and corresponding answer tables, from which we choose the optimal clique and its corresponding result table.

The main contributions of this paper are:

1. We propose consistent matching relationships to solve entity inconsistency, which should be hold not only between any two answer tables but also between each answer table and the query table. We also propose answer table selection method based on consistent matching degree.
2. To our best knowledge, we are the first that propose to settle consistent entity augmentation problem by building consistent γ-coverage clique. We regard web tables having consistent matching relationships with query table as nodes and consistent matching relationships between web tables as edges, therefore we get a consistent clique. It is proved that a consistent result table could be built by considering tables in consistent clique to be answer tables.
3. We have performed extensive experiments on 4 real-life datasets of web tables. Experimental results demonstrate that our entity augmentation framework has high accuracy and reliability, meanwhile ensuring entity consistency.

This paper is structured as follows. We start in Section 2 by modeling the problem. The entity augmentation method by building consistent γ-coverage clique is described in Section 3. Experimental evaluating results are presented in Section 4. Related work is discussed in Section 5 and we conclude in Section 6.

2 PROBLEM MODELING

In recent years, entity augmentation has attracted more and more attention from researchers. Yakout et al. proposed indirect matching to augment entities [7]. Lehmberg et al. proposed Mannheim Search Join Engine to extend query table [8, 9, 10]. During entity augmentation, they only considered the entity column and the column to be augmented. The above mentioned entity augmentation techniques are all based on the assumption that columns are independent.

In case that there exist some n-ary query tables and web tables, this assumption will lead to low precision and entity inconsistency. When splitting a complete web table into several pieces, the semantics of this table is segmented, thus causing entity inconsistency in the result table and leading to low precision. In order to get

consistent entity and high precision for result table, we propose that answer tables should have consistent matching relationships with each other to ensure high consistency and have consistent matching relationship with query table to ensure high precision of the result table. In order to make consistent matching relationship more understandable, we individually define concepts of semantic relevance (Definition 1) and table matching degree (Definition 2). Consistent matching relationship between two tables is made up of semantics part and value part. Semantic relevance gives the degree of how two tables are semantic related. Also, table matching degree reflects the probability that two tables consistently match in value. Before discussing the problem model, we introduce notations in Table 1.

Notation	**Description**
$Q(E, A)$	The query table with entity set E and attribute set A
$T = \{t_1, t_2, t_3, \ldots\}$	A set of web tables
γ	A specific coverage
AT	An answer table set for the query table
RT	The result table for the query table
$SRD(t_i, t_j)$	The semantic related degree for table t_i and t_j
$TMD(t_i, t_j)$	The table matching degree between t_i and t_j
$U(V, S)$	The clique U with the node set V and the edge set S
$cov(RT, Q)$	The coverage of result table RT to query table Q
$cov(U, Q)$	The coverage of clique U to query table Q
$SMS(Q, t)$	The semantic matching score between query table Q and web table t

Table 1. Notations

2.1 Consistent Matching Relationship

Considering two tables, we think they are semantically related if their entity sets are semantically related, because concepts of entity columns are thought to represent the tables' semantic concepts [11]. In the following paragraphs, we will first introduce how to get semantic related degree between two tables by calculating entity sets' relatedness.

We use Probase [12] to determine if two entity sets are semantically related. For each entity in one table, we calculate its relatedness to each entity in another table by using Jaccard Similarity on two entities' concept sets returned by Probase. Then, we aggregate the pairwise entity relatedness to get two tables' (t_i and t_j) semantic related degree, denoted as $SRD(t_i, t_j)$, which could be calculated using the following formula:

$$SRD(t_i, t_j) = \frac{\sum_{\forall e_i \in E_i, \forall e_j \in E_j} Jaccard(C(e_i), C(e_j))}{|E_i|\,|E_j|} \tag{1}$$

where E_i, E_j are entity sets of t_i and t_j, respectively, $C(e)$ denotes concept set of entity e, and $Jaccard(C(e_i), C(e_j)) = \frac{|C(e_i) \bigcap C(e_j)|}{|C(e_i) \bigcup C(e_j)|}$.

Definition 1 (Semantic Relevance)**.** Given two tables t_i and t_j, we call the table t_i is semantically related with t_j, denoted as $t_i \overset{sem}{\approx} t_j$, if $SRD(t_i, t_j) \geq \theta$.

Generally, two tables are thought to match with each other in value if both tables have the same value in the same attribute for the same entities. For example, if two tables both have entity "United States", we expect them to have "Washington" in column *capital*. If there are a certain proportion of same entities with same value in column *capital* of two tables, two *capital* columns are regarded as matching columns. So, to determine if two tables are consistent matching with each other in value, we have to find their mapping columns with matching labels. Tables are matched in value if all the mapping columns are matching columns. Specially, because a query table misses attribute values, it is thought to match with a web table in value when they have same entities and same attributes. We propose the concept of *Table Matching Degree* to determine whether two tables are matched in value or not.

Definition 2 (Table Matching Degree)**.** Given two tables t_i and t_j, query table Q, C_i and C_j are respectively mapping columns for t_i and t_j. The table matching degree between t_i and t_j, denoted as $TMD(t_i, t_j)$, could be calculated using the following formula:

$$TMD(t_i, t_j) = \begin{cases} \frac{|t_i.E \bigcap t_j.E|}{\min\{|t_i.E|,|t_j.E|\}} \times \frac{|t_i.A \bigcap t_j.A|}{\min\{|t_i.A|,|t_j.A|\}}, & \text{if } t_i = Q \vee t_j = Q, \\ \frac{|\{<C_i,C_j>|C_i \approx C_j\}|}{|\{<C_i,C_j>\}|}, & \text{if } |\{< C_i, C_j >\}| \neq 0, \\ -1, & \text{otherwise,} \end{cases} \tag{2}$$

where $t.E$ denotes t's entities set, and $t.A$ denotes a set of attributes names of table t; $C_i \approx C_j$ denotes columns C_i and C_j are matching columns which meet column matching degree threshold (given in Section 3.3.2).

In Definition 2, table matching degree is -1 when there are no mapping columns between two web tables. Under this situation, whether two tables have consistent matching relationship or not is determined only by considering two tables' semantic relevance. When there are mapping columns in two tables, whether two tables have consistent matching relationship or not is determined by considering both semantic relevance and table matching degree between two tables.

Definition 3 (Consistent Matching Relationship)**.** Given two tables t_i and t_j, we call t_i has consistent matching relationship with t_j, denoted as $t_i \overset{cm}{\leftrightarrow} t_j$, iff ($t_i \overset{sem}{\approx} t_j \wedge ((TMD(t_i, t_j) \geqslant \tau \wedge t_i \neq Q \wedge t_j \neq Q) \vee TMD(t_i, t_j) = -1 \vee TMD(t_i, t_j) > 0)$).

Theorem 1. Consistent matching relationship is symmetric. Given tables t_i and t_j, if $t_i \overset{cm}{\leftrightarrow} t_j$ holds, then $t_j \overset{cm}{\leftrightarrow} t_i$ holds.

Proof. Under the condition that $t_i \overset{cm}{\leftrightarrow} t_j$, it is obvious that $t_i \overset{sem}{\approx} t_j$ and $((TMD(t_i, t_j) \geqslant \tau \wedge t_i \neq Q \wedge t_j \neq Q) \vee TMD(t_i, t_j) = -1 \vee TMD(t_i, t_j) > 0)$ hold. According to Equation (1) and Definition 1, $t_j \overset{sem}{\approx} t_i$ holds. And, according to Equation (2),

$TMD(t_i, t_j) = TMD(t_j, t_i)$ holds, which means that $((TMD(t_j, t_i) \geqslant \tau \wedge t_j \neq Q \wedge t_i \neq Q) \vee TMD(t_j, t_i) = -1 \vee TMD(t_j, t_i) > 0)$ holds. So, $t_j \overset{cm}{\leftrightarrow} t_i$ holds. That is to say, consistent matching relationship is symmetric. □

2.2 Problem Statement

For a consistent entity augmentation result, the consistent matching relationship should both exist between any two answer tables and exist between each answer table and the query table. A result table built by answer tables satisfying the above conditions is considered to be a consistent result table.

Definition 4 (Consistent Result Table)**.** Given a query table Q and a set of web tables T, RT is a result table for Q and AT is its corresponding answer tables set. RT is a consistent result table for Q if and only if:

1. Each table in AT has consistent matching relationship with the query table Q.
2. Tables in AT have consistent matching relationships with each other.

Problem Statement. Given query table $Q(E, A)$ and a set of web tables T, where $Q.E$ denotes query table's entities, and $Q.A$ denotes a set of names of attributes which are to be augmented. *Consistent Entity Augmentation* is to find a group of answer tables AT ($AT \subseteq T$) for building a consistent result table RT whose coverage to query table reaches the specific threshold γ.

If we take web tables having consistent matching relationships with query table as nodes and consistent matching relationships as edges, then a clique is the complete subgraph in which tables as nodes consistently match with each other. Tables in a clique come to be answer tables when their corresponding result table's coverage to query table is larger than the specific threshold. So, entity augmentation problem could be converted into building consistent γ-coverage clique (Definition 7) problem.

Definition 5 (Result Coverage)**.** Given a query table Q, a clique U composed of answer tables for Q and the corresponding result table RT, the coverage of result table RT or clique U to Q, denoted as $cov(RT, Q)$ or $cov(U, Q)$, could be calculated using the following formula:

$$cov(RT, Q) = cov(U, Q) = \frac{\#\text{augCells}(RT)}{\#Cells(Q)} \tag{3}$$

where $\#\text{augCells}(RT)$ and $\#Cells(Q)$ denote the number of cells augmented by the clique U, and the number of cells which should be augmented in the query table, respectively.

Definition 6 (Consistent Clique)**.** Given a query table Q and a set of candidate tables CT, a clique $U(V, S)$ is a consistent clique for Q when:

- V is a subset of CT, and each table in V has consistent matching relationship with Q;
- S is a set of consistent matching relationships between table pairs in V, $\forall(t_i, t_j) \in S$, $t_i \overset{cm}{\leftrightarrow} t_j$ holds;

where CT is selected from source web tables T, having at least one same entity with the query table.

Definition 7 (Consistent γ-Coverage Clique)**.** Given a query table Q, a consistent clique $U(V, S)$ for Q, and a coverage threshold γ, a clique U is a consistent γ-coverage clique for Q when $cov(U, Q) \geq \gamma$.

Theorem 2. Given a query table Q, a set of web tables T, a consistent result table RT for Q with γ coverage could be built using V as answer tables if and only if existing a consistent γ-coverage clique $U(V, S)$ for Q.

Proof. Firstly, we prove necessity. When a consistent result table RT for Q with γ coverage exists, we could regard its answer tables as nodes set V and consistent matching relationships between tables as edges set S, then get a complete graph $U(V, S)$. According to Definition 4, consistent matching relationships in RT exist both between any two answer tables and between each answer table and the query table, which makes the complete graph $U(V, S)$ to be a consistent clique for Q. Furthermore, under the condition that coverage of result table RT to Q reaches γ, it is obvious that $U(V, S)$ is a consistent γ-coverage clique for Q.

Secondly, we will prove sufficiency. If existing a consistent γ-coverage clique $U(V, S)$, according to Definition 7, every table in V has consistent matching relationship with Q. For clique U, $t_i \overset{cm}{\leftrightarrow} t_j$ holds for any two tables t_i, t_j in V. Using V as answer tables, a consistent result table RT could be built for Q. Furthermore, when $cov(U, Q) \geq \gamma$, $cov(RT, Q) \geq \gamma$ holds. □

Theorem 2 is given to validate that we could get a consistent result table for query table Q with γ coverage by building Consistent γ-Coverage Clique. Based on this theorem, the entity augmentation problem could be converted into the *Consistent γ-Coverage Clique Problem* as follows.

The Consistent γ-Coverage Clique Problem. Given a query table Q and a set of web tables T, the consistent γ-coverage clique problem is to build a set of cliques whose coverage to query table Q are larger than γ.

3 ENTITY AUGMENTATION BY BUILDING CONSISTENT γ-COVERAGE CLIQUE

3.1 Solution Overview

Given the query table is missing some attributes' values, as one application of data integration, the aim of entity augmentation is to return a result table containing

query table's missing data. Existing entity augmentation techniques assume web tables are EAB relations [7, 8, 9, 10]. When an n-ary query table has multiple columns to be extended, the result table consolidated by binary tables will suffer from entity inconsistency problem. The goal of our paper is to build consistent results for n-ary entity augmentation queries.

Figure 3 gives the framework for consistent entity augmentation. At first, we find candidates from source tables with aid of the index $EI(Q)$. Given a query table Q and a set of web tables T, index $EI(Q)$ will return tables having at least one same entity with the query table. In order to get consistent entity for result table, we propose that answer tables should have consistent matching relationships with each other to ensure high consistency and precision of the result table. Based on graph theory, consistent entity augmentation query problem could be converted into consistent γ-coverage clique building problem. We also prove that tables in consistent γ-coverage clique are answer tables for entity augmentation query in Theorem 2.

To build γ-coverage cliques, we first find seed cliques as initials using *semantic matching score*. For each seed clique, we try to expand it with tables and get a corresponding consistent γ-coverage clique. For a seed clique not satisfying the requirement of coverage, we have to find other web tables called clique tables to improve its coverage. In order to get clique tables, we calculate table potential for each candidate table, which is made up of its consistent matching degree with the query table and its consistent matching degree with each table in clique. Obviously, a web table is selected to be a clique table, if it has consistent matching relationship not only with query table but also with each table in existing clique. In other words, the higher is the table potential, the greater is the probability of being a clique table. Based on this intuition, we select one with the highest potential as a new clique table, and add edges between this new addition and each original clique table. After that, we continue to expand the clique by adding tables with high potential until its coverage reaches γ. For each seed table, we will get a consistent γ-coverage clique and corresponding answer tables. To get final consistent result table, we select the optimal clique by measuring several indicators such as consistency support degree, diversity of source and coverage. At last, tables in optimal clique are regarded as answer tables which provide attribute values for the final consistent result table.

3.2 Finding Seed Cliques Based on Semantic Relevance

In our solution, the first step is to find seed cliques, which is the base for building consistent γ-coverage cliques. In most cases, a query table contains less information, so seed cliques are introduced to provide as much information as possible for entity augmentation. As shown in Figure 1, "The Lord of The Rings" is not only a best-selling novel, but it is also a popular movie. The information provided by the query table is not enough to determine entity's concept. If we provide more information, such as *publishing time*, it will be helpful for identifying concept of the entity. So, it is very important to find seed cliques with tables "matching with" query table.

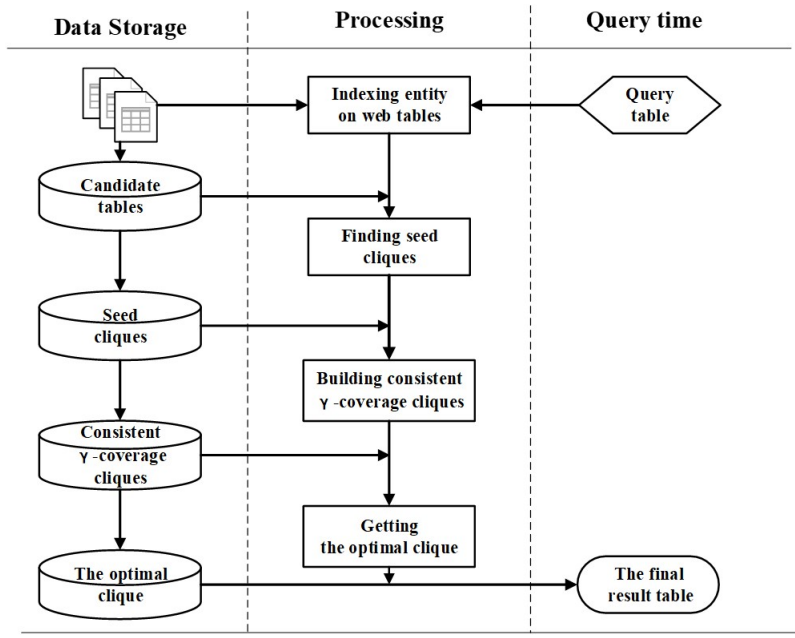

Figure 3. A framework for consistent entity augmentation

At the beginning, we can get a graph made up of isolated tables without consistent matching relationships among them. At that time, each node in the graph could be regarded as an initial clique, which would be expanded by other nodes (web tables) to form a consistent γ-coverage clique. In order to improve accuracy and reduce time cost, we should select seed cliques having more consistent matching degree with the query table as initial cliques. Because each initial clique only has one node, the problem of finding seed cliques could be converted to the problem of finding seed tables.

To find seed tables, schema matching techniques are usually used, which is generally based on schema level features (e.g. attribute names) and instance level features (e.g. attribute values). Previous works only consider schema information and select seed tables when they have the same entities and same attribute names with the query table [7]. Using the existing method, for example in Figure 1, we can get seed tables t_1 and t_2 because their matching scores are the same. Obviously, t_2 is a false seed table whose entities are films, while entities of query table are books. During entity augmentation, error will be amplified by using t_2 as the seed table.

This problem is due to only considering schema information. In fact, based on schema level features, we can also consider semantic relevance between a candidate table and the query table. And, according to *Problem Statement* described in Section 2.2, every answer table should have a consistent matching relationship with the

query table, so are seed tables. As for selection of seed tables, we hope that seed tables have a higher consistent matching degree than other tables, based on that they all have consistent matching relationships with a query table. To measure consistent matching degree between a web table and a query table, we need to calculate *semantic matching score*.

Definition 8 (Semantic Matching Score)**.** Given the query table $Q(E, A)$ and a web table $t(K, B)$, the semantic matching score between Q and t, denoted as $SMS(Q, t)$, could be calculated using the following formula:

$$SMS(Q,t) = \phi(SRD(Q,t),\theta) * \phi(TMD(Q,t),0) \tag{4}$$

where $\phi(p,\theta) = [\![p > \theta]\!]^{p}_{-\infty}$, which denotes $\phi(p,\theta) = p$ when $p > \theta$, otherwise $\phi(p,\theta) = -\infty$. $SRD(Q,t)$ denotes semantic related degree between query table and web table. When $SRD(Q,t) < \theta$, we think the table t and the query table are not semantically related.

Given the query table and candidate tables CT, for each table in CT, we can get its semantic matching score with query table Q. According to semantic matching scores, we can get top-k seed tables, which makes the initial cliques taking shape.

3.3 Building Consistent γ-Coverage Cliques

For any seed clique, we need to expand it by adding web tables (denoted as clique tables) if its coverage is smaller than γ. Based on seed cliques, we can build consistent γ-coverage clique by dynamically adding clique tables. According to Definition 6, a clique is considered as a consistent clique when its nodes have a consistent matching relationships with the query table and its edges are consistent matching relationships. Through Definition 8, every seed clique meets the requirement of being a consistent clique. The way to find a clique table is considering its consistent matching degree both with the query table and with each table in the current clique. A node potential reflects consistent matching degree between the clique table and the query table, while an edge potential reflects consistent matching degree between two tables in the clique. The potential of a candidate table is the sum of its node potential and all edge potentials between itself and each table in the clique.

3.3.1 Node Potential

Node potential is proposed to measure the probability of a candidate to become a clique table, which mainly considers the coverage of the clique after adding the web table into it. In order to represent the contribution of a web table for increasing clique's coverage, we propose the concept of *Supplementary Coverage*.

Definition 9 (Supplementary Coverage)**.** Given a query table $Q(E, A)$, candidate tables CT and a consistent clique $U(V, S)$ for Q whose coverage is smaller than γ.

The supplementary coverage of a web table t ($t \in CT - V$) about the clique U for the query table Q, denoted as $SC(t, U, Q)$, is the incremental coverage of clique U improved by table t.

$$SC(t, U, Q) = cov(U \cup \{t\}, Q) - cov(U, Q). \tag{5}$$

Node Potential. Given a query table $Q(E, A)$, candidate tables CT and a consistent clique $U(V, S)$ for Q whose coverage is smaller than γ. Node potential $\varphi_{node}(t, U, Q)$ for table t ($t \in CT - V$) is calculated as follows:

$$\varphi_{node}(t, U, Q) = \phi(SRD(Q, t), \theta)) * SC(t, U, Q). \tag{6}$$

3.3.2 Edge Potential

For web tables missing labels, Equation (6) could not give accurate node potential for those tables. Using Figure 1 as an example, the supplementary coverage of t_3 is 0 due to missing a column label in t_3. Actually, with label *year* in the second column, t_3's supplementary coverage should be 0.2 based on seed table t_1. We can settle this problem by transferring the label *year* from the third column of t_1 to the second column of t_3 by finding matching columns.

Two columns will be matching columns when their column matching degree is larger than a specific threshold. For two columns C_i and C_j in tables t_i and t_j, respectively, their column matching degree, denoted as $CM(C_i, C_j)$, reflects the similarity degree of two columns. Column matching degree is the ratio of the same entities with the same values in two columns. For column matching degree of two columns, we mainly consider three situations:

1. both columns' elements are character values;
2. both columns' elements are years;
3. both columns' elements are numeric values.

For character values, two values are regarded as the same when their EditDistance is larger than a specific threshold. For two years, two values are regarded as the same when they equal to each other. Specially, for numeric values, two values are regarded as same when they meet a specific unit conversion.

Definition 10 (Matching Columns)**.** Given two tables t_i and t_j, C_i and C_j are columns for t_i and t_j, respectively, they are regarded as matching columns, denoted as $C_i \approx C_j$, if $CM(C_i, C_j) > \sigma$ holds.

In order to calculate node potential for a web table with missing column labels, we try to transfer labels of columns which are query table's mapping columns to their matching columns.

After transferring labels, we can get edge potential over the table pair when two tables have a consistent matching relationship with each other. Edge potential denotes consistent matching degree between two tables, considering two tables' consistent matching degree both in semantics and in value.

Edge Potential. Given candidate tables CT and a consistent clique $U(V,S)$ for a query table Q whose coverage is smaller than γ. Edge potential $\varphi_{edge}(t_i,t_j)$ for edge between table t_i in $CT-V$ and table t_j in V could be calculated using following formula:

$$\varphi_{edge}(t_i,t_j)=\begin{cases}\phi(SRD(t_i,t_j),\theta), & \text{if } TMD(t_i,t_j)=-1,\\ \frac{\phi(SRD(t_i.t_j),\theta)+\phi(TMD(t_i.t_j),\tau)}{2}, & \text{otherwise.}\end{cases} \tag{7}$$

3.3.3 Getting Clique Tables

The goal of this stage is continuously to find clique tables from the set of candidate tables CT, to make the coverage of clique approach γ. So, to find clique table for clique $U(V,S)$, we regard the sum of node potential and edge potential as table potential, denoted as $\varphi_{table}(t_i)$.

$$\varphi_{table}(t_i)=\varphi_{node}(t_i,U,Q)+\sum_{\forall t_j\in V}\varphi_{edge}(t_i,t_j) \tag{8}$$

where $t_i\in CT-V$.

According to Equation (8), we can get the clique table t_U for the clique $U(V,S)$ using the following formula:

$$t_U=\arg\max_{\forall t_i\in CT-V}\varphi_{table}(t_i). \tag{9}$$

In order to build consistent γ-coverage clique, we select seed tables first according to semantic matching score. Then, for each seed table whose coverage is smaller than γ, we separately calculate table potentials for candidates and select one with the maximal value as clique table. Each time when new clique table is putting in, the clique is expanded and its coverage is improved. Above progress is repeated until clique's coverage reaches γ.

Algorithm 1 gives an algorithm for finding clique table. The input of this algorithm is the query table Q, candidate tables set CT and the clique $U(V,S)$ whose coverage is smaller than γ. At first, labels from matching columns are transferred to tables with missing labels (line 8). Then, table potential is calculated for each candidate table (line 9). At last, the clique table t with the maximal table potential is chosen (line 11–12).

Algorithm 2 introduces the procedure of iteratively building consistent γ-coverage cliques based on seed cliques. The input is the query table Q, candidate tables

Algorithm 1: findCTables($Q(E, A), CT, U(V, S)$)

Input: $Q(E, A)$: the query table;
CT: candidate table set;
$U(V, S)$: a clique whose coverage is smaller than γ;
Output: t_U: clique table for $U(V, S)$

```
1  map ← ∅;
2  for each table t_i in CT − V do
3      sum ← 0;
4      for each table t_j in V do
5          for each column C_j in t_j' column set Γ used to augment Q do
6              for each column C_i in t_i do
7                  if CM(t_i, t_j) > σ and A(C_i) ≠ A(C_j) then
8                      A(C_i) ← A(C_j);
9          sum ← sum + φ_edge(t_i, t_j);
10     add ⟨t_i, sum + φ_node(t_i, U, Q)⟩ to map;
11 map = HashMap(map);
12 return top − 1 table in map;
```

set CT and the number k of seed cliques. At the beginning, the set of initial cliques is empty (line 1). Then, we get seed cliques (line 2). For each seed clique, the function $expand\_Clique(Q, CT, U_i(V, S), \varepsilon)$ is executed iteratively until the clique's coverage reaches γ (line 4). In practice, sometimes when the result table returned could not satisfy the requirement of coverage γ, we would return the clique whose coverage is the closest to γ instead. Under this situation, a clique U is returned when its coverage converges (line 13–14).

3.4 Getting Consistent Result Table Based on Optimal Clique

After getting a set of consistent γ-coverage cliques, we can obtain the corresponding answer tables for the query. At this stage, we should select an optimal clique, whose nodes are answer tables for building the final consistent result table. We consider the following indicators for selecting the optimal clique:

Consistency Support Degree: We measure consistency support degree using the average value of table potentials in clique $U(V, S)$. A higher value means the result table could keep higher consistency with the query table.

Diversity of Source: This indicator reflects diversity of answer tables (i.e., the diversity of tables in clique $U(V, S)$). We measure diversity of source simply by the number of answer tables. Generally, the more diverse the data source is, the less consistent the result table will be. In fact, fewer answer tables are helpful for maintaining the entity consistency.

Algorithm 2: *build_CC Cliques*(Q, CT, k)

Input: $Q(E, A)$: the query table;
CT: candidate table set;
k: the number of seed cliques;
Output: $\mathbb{C}$: a set of consistent γ-coverage cliques

```
1  ℂ ← ∅;
2  get the seed cliques U_1, U_2, ··· , U_k;
3  for each seed clique U_i(V, S) do
4      U_i(V, S) ← expand_Clique(Q, CT, U_i(V, S), ε);
5      add U_i(V, S) in ℂ;
6  return ℂ;
7
8  Function expand_Clique(Q, CT, U_i(V, S), ε)
9  if cov(U_i, Q) ≥ γ then
10     return U_i(V, S);
11 else
12     t ← findCTables(Q(E, A), CT, U_i(V, S));
13     if SC(t, U_i, Q) ≤ ε then
14         return U_i(V, S);
15     else
16         add t in V; for each t_i in V do
17             add ⟨t, t_i⟩ in S;
18         expand_Clique(Q, CT, U_i(V, S), ε);
```

Coverage: Even though we set the coverage threshold, the coverage of result tables returned by different cliques will be different. Apparently, we prefer clique with a higher coverage.

We select optimal clique to build the final consistent result table by using the following formula.

$$U_{final} = \arg\max_{U \in \mathbb{C}} \frac{\varphi(U) * cov(U, Q)}{|V|^2} \tag{10}$$

where $\varphi(U) = SMS(Q, t_{seed}) + \sum_{t_i \in U.V - \{t_{seed}\}} \varphi_{table}(t_i)$ and $t_{seed} \in U.V$, $\mathbb{C}$ is a set of consistent γ-coverage cliques returned by Algorithm 2.

After getting the optimal clique, we could build a consistent result table using answer tables in this clique. Algorithm *get_RT* describes the procedure with answer tables AT, query table Q and the similarity threshold α as input. In Algorithm 3, array *flag* is a sign array used to record whether a cell in the result table has been filled or not (line 1). To avoid entity inconsistency, the fill-in progress is based on tables. By sorting answer tables according to their *semantic matching*

Algorithm 3: $get\_RT(AT, Q(E,A), \alpha)$

Input: $Q(E,A)$: the query table;
AT: answer tables set;
α: the similarity threshold;
Output: RT: the final result table

```
1 flag[] ← 0, RT ← Q;
2 while AT! = ∅ do
3     t ← arg max_{t∈AT} SMS(Q, t);
4     for each rt.cell ∈ RT do
5         if flag[rt.cell] = 0 && arg max_{t.cell∈t} sim(rt.cell, t.cell) ≥ α then
6             t.cell_max ← arg max_{t.cell∈t} sim(rt.cell, t.cell);
7             flag[rt.cell] ← 1;
8             rt.cell(v) ← t.cell_max(v);
9     AT ← AT − {t};
```

score, we select all available information from tables in turn to fill the result table (line 2–9).

4 EXPERIMENTS

We are the first who have proposed to settle consistent entity augmentation problem by building a consistent γ-coverage clique. We conduct a set of experiments to evaluate the effectiveness of our method (denoted as EACC) for getting high quality of entity augmentation results by selecting answer tables based on consistent matching degree. In the following paragraphs, we first evaluate the effectiveness of our answer table selection method based on consistent matching degree, then evaluate the performance of our consistent entity augmentation system based on building consistent γ-coverage clique.

Four real-life datasets (Books, Country, Company and Song) from WDC Web Table Corpora (`http://webdatacommons.org/webtables/`) and four query tables in Table 2 are used in our experiments. We compiled the complete ground truth for query tables by manually identifying and extracting the desired information from Wikipedia [13].

Dataset Name	Entity	Attributes to be Augmented	Number of Entities
Books	Book name	Author, Published date	101
Country	Country name	Capital, Area	26
Company	Company name	Headquarter, Industry	48
Song	Song name	Artist, Time	47

Table 2. Features of query tables

All methods were written in Java and all tests were conducted on a PC with a 2.93 GHz Intel CPU and 8 GB RAM, running Windows 7.

4.1 Effectiveness of Our Answer Table Selection Method Based on Consistent Matching Degree

The effectiveness of our answer table selection method based on consistent matching degrees is evaluated by comparing other three methods as follows:

1. EACC: It is our entity augmentation method based on consistent matching relationship and consistent matching degree, which finds answer tables by building consistent γ-coverage clique.
2. EATSP: It uses topic sensitive pagerank (TSP) [14] to select answer tables and the corresponding result tables. TSP score is computed for each table as its matching degree with the query table. EATSP implements entity augmentation queries according to TSP score of each table.
3. EAWOS: It selects answer tables depending on Table Matching Degree calculated only by value similarity between tables, which could be regarded as EACC method without semantics.
4. DMA: Different from above three methods, DMA is a kind of direct matching approach only considering web tables which match directly with the query table. It selects a web table as an answer table according to consistent matching degree between this web table and the query table.

As we know, InfoGather is a typical entity augmentation system which selects answer tables by TSP scores. For n-ary queries, however, InfoGather splits tables into several EAB relations, while our method EACC regards all attributes in a table as a whole. Instead of splitting attributes into pieces, EATSP regards all the attributes as a whole but selects answer tables by TSP scores. Compared with EACC, EAWOS considers only value matching but ignoring semantic matching between tables. EACC, EASTP and EAWOS are all approaches which consider indirectly matching tables in addition to the directly matching ones. The direct match approach and indirect match approach are described in [7]. DMA is a naive method that attempts to directly match the query table with the web tables. The comparison of four entity augmentation methods are listed in Table 3.

Augmentation Method	**Matching Mode**	**Matching Score**
EACC	Indirect & Direct	Consistent matching degree
EASTP	Indirect & Direct	TSP
EAWOS	Indirect & Direct	Table matching degree
DMA	Direct	Consistent matching degree

Table 3. Features of four methods

We extracted 16 GB web tables from WDC Web Table Corpora as data source and deleted some off-grade web tables – such as tables with confusing information, more empty values, and ambiguous subject columns. There are 3 000 000 web tables in the data source in which the maximum number of rows and columns are 454 and 42, and the minimum number of rows and columns are 3 and 2. The average number of rows and columns in web tables are 13 and 5. We run four methods respectively to augment four query tables in Table 2 and compare their performance. Under different coverage threshold (varying from 0.1 to 0.8), their coverage, precision, consistency and reliability are compared. The evaluation metrics are as follows.

$$cov = \frac{|values\_found|}{\#Cells(Q)}, \tag{11}$$

$$pre = \frac{|values\_found \bigcap values\_truth|}{|values\_found|}, \tag{12}$$

$$con = \begin{cases} 1, & \text{if } |AT| = 1, \\ 1 - |\log_{10}[\![avgSim(t_1,t_2) \geq 0.1]\!]_{0.1}^{avgSim(t_1,t_2)}|, & \text{otherwise}, \end{cases} \tag{13}$$

where $avgSim(t_1,t_2) = avg \sum_{t_1,t_2 \in AT} \frac{SRD(t_1,t_2)+TMD(t_1,t_2)}{2}$.

Coverage is the ratio of filled values to values to be filled. Precision is the fraction of correctly filled values that have been really filled. Values extracted from Wikipedia are regarded as truth values. Consistency is measured by the average value of similarity between any two answer tables. Obviously, answer tables having high consistency will receive a high similarity. *Semantic relevance* and *table matching degree* between two tables are considered for evaluating their similarity. Inspired by F-measure, reliability is defined as harmonic mean of coverage, consistency and precision.

$$relia = \frac{3 * cov * pre * con}{cov * pre + pre * con + cov * con}. \tag{14}$$

4.1.1 Evaluation for Quality of Result Tables

We have implemented four methods for entity augmentation: EACC, EATSP, EAWOS and DMA. Figures 4, 5, 6 show respectively coverage, consistency, and precision for four query tables. We varied results by considering different coverage threshold from 0.1 to 0.8.

From Figures 4, 5, 6, we have the following observations:

1. With the increase of γ, the coverage results of four methods are increasing. In most cases, the coverage of EATSP and DMA is higher than that of others under different coverage thresholds. Answer tables in EATSP and DMA should only match with the query table, while answer tables in EACC and EAWOS should

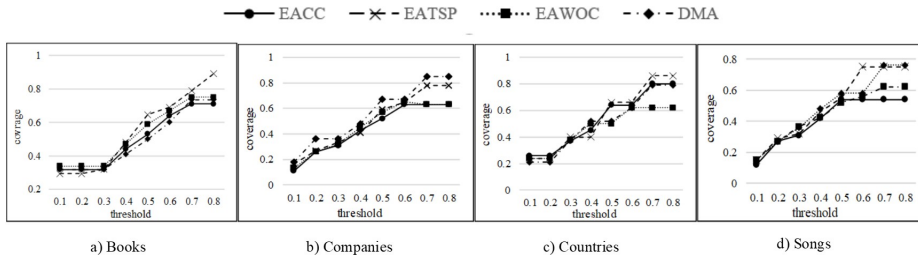

Figure 4. Coverage values on four datasets

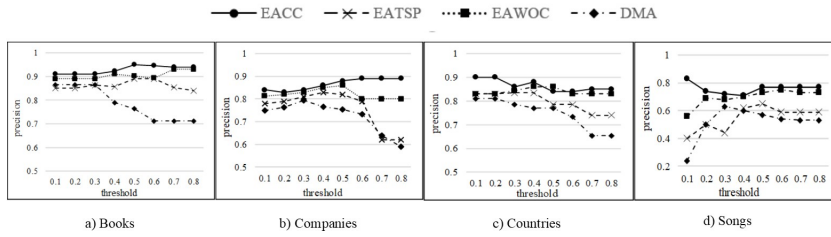

Figure 5. Precision values on four datasets

meet consistent matching relationships with each other, which causes less answer tables to be left in the clique. On most datasets, EATSP's coverage is higher than that of DMA, because EATSP considers indirectly matching tables in addition to the directly matching ones, and DMA only considers directly matching tables.

2. On four datasets, the precision of EACC and EAWOS is obviously higher than that of others, because two methods all get answer tables based on matching relationships between web tables. By comparison, EACC is better than EAWOS. EACC considers consistent matching relationships either in semantics or in value, but EAWOS only considers table matching relationships in value, which greatly reduces precision. EATSP gets answer tables based on topic sensitive

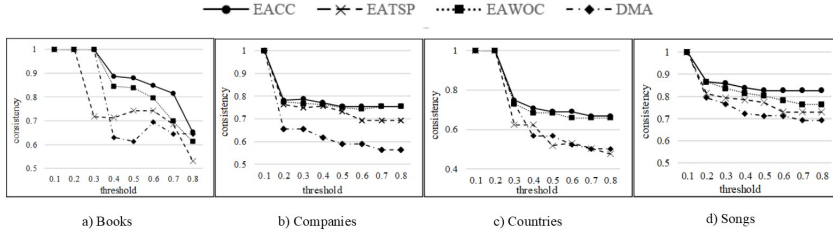

Figure 6. Consistency values on four datasets

pagerank algorithm (TSP), considering mainly schema level features among tables and text features on web page. Due to ambiguity of entity and redundancy of web page information, EATSP gets lower precision. In Figure 5, EACC based on consistent matching degree significantly outperforms EATSP, EAWOS and DMA, which confirms that precision results could be improved by considering consistent matching relationship.

3. For we evaluate the consistency mainly based on the average similarity value between answer tables, the consistency value reaches the highest when there is only one answer table. With the increase of coverage threshold, most consistency results of four algorithms on four datasets decrease due to the increasing number of answer tables. In most cases, the consistency results of our EACC are greater than EAWOS, EATSP and DMA, because EACC selects answer tables not only considering consistent matching degree between each candidate table and the query table but also consistent matching degree among answer tables.

Experimental results show that EACC algorithm based on consistent matching degree significantly outperforms EATSP, EAWOS and DMA in precision and consistency even though the coverage of EACC is a little lower than that of three other algorithms in some cases.

4.1.2 Evaluation for Reliability of Result Tables

Based on coverage, precision and consistency, we can get the reliability of result tables under different coverage thresholds using Equation (14). Figure 7 shows the experimental results.

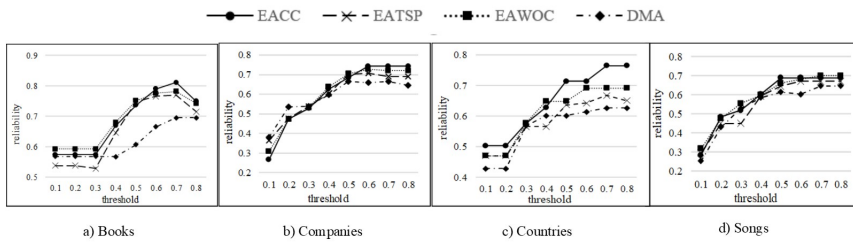

Figure 7. Reliability values on four datasets

With the increase of coverage thresholds, the general trends of reliability results for four algorithms are increasing. For reliability which is the harmonic mean of precision, coverage and consistency, EACC has the highest reliability. As the threshold grows, the reliabilities of four algorithms reach highest (0.81 for EACC, 0.76 for EATSP, 0.78 for EAWOS and 0.695 for DMA).

In summary, even though the coverage of EACC is a little lower than that of three other algorithms in some cases, EACC has highest precision, consistency

and reliability, which will be helpful for returning consistent entity augmentation results. According to above comparison experiments, we come to the conclusion that our answer table selection method based on consistent matching degree either in semantics or in value is helpful for returning effective and consistent result table for entity augmentation queries.

4.1.3 Evaluation for Runtime Performance

We evaluate the runtime performance of the given four methods. We give the runtime performance with the increase of coverage thresholds in Figure 8.

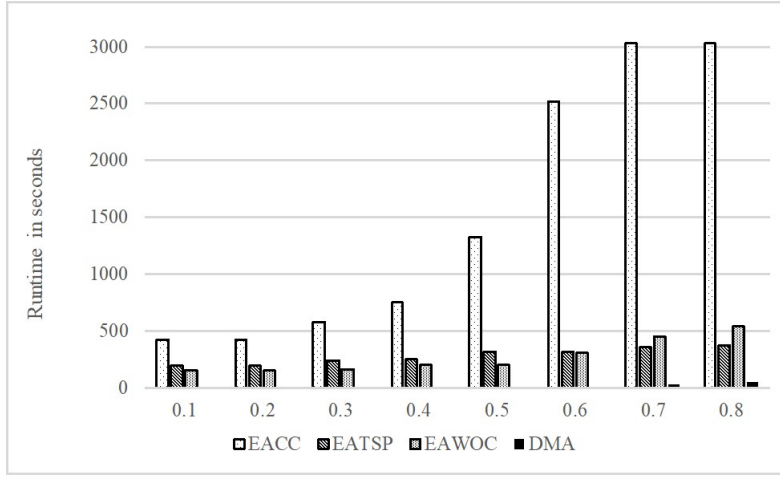

Figure 8. Runtime values on different coverage thresholds

In Figure 8, with the increase of coverage threshold, EACC's runtime has the most obvious growth trend and relatively high runtime. That is because EACC costs a lot of time in searching semantics and creating cliques meeting the requirement of coverage. EAWOS's runtime is obviously lower than EACC, because selection of answer tables in EAWOS does not consider semantics. In Figure 8, we can find that the runtime of EATSP varies little under different coverage thresholds, for it just calculates matching degree between each candidate and the query table, no matter what coverage threshold is. DMA is a naive method that attempts to directly match the user query table with the web tables, which takes the least time.

Although EACC takes much time for searching semantics and creating cliques meeting the requirement of coverage, it has prominent performance in precision, consistency and reliability for entity augmentation. The experiments demonstrate that consistent matching relationship proposed by this paper is much helpful for settling entity inconsistency problem caused by existing methods. Using parallel algorithms to improve the efficiency of consistent entity augmentation method is our future work.

4.2 Performance of Consistent Entity Augmentation

Web tables are assumed as EAB relations in existing entity augmentation techniques. InfoGather is such a typical entity augmentation system proposed by Mohamed Yakout, that is based on graphical models and topic sensitive pagerank algorithm. To compare our algorithm EACC with InfoGather, we use coverages of result tables returned from InfoGather as EACC's coverage thresholds.

Different from InfoGather which answers n-ary queries by splitting attributes into pieces, our EACC method augments entities by building consistent γ-coverage cliques based on consistent matching degree.

We do experiments to compare the performance of EEAC and InfoGather, including their coverage, precision, consistency and reliability. The experimental results are given in Figure 9.

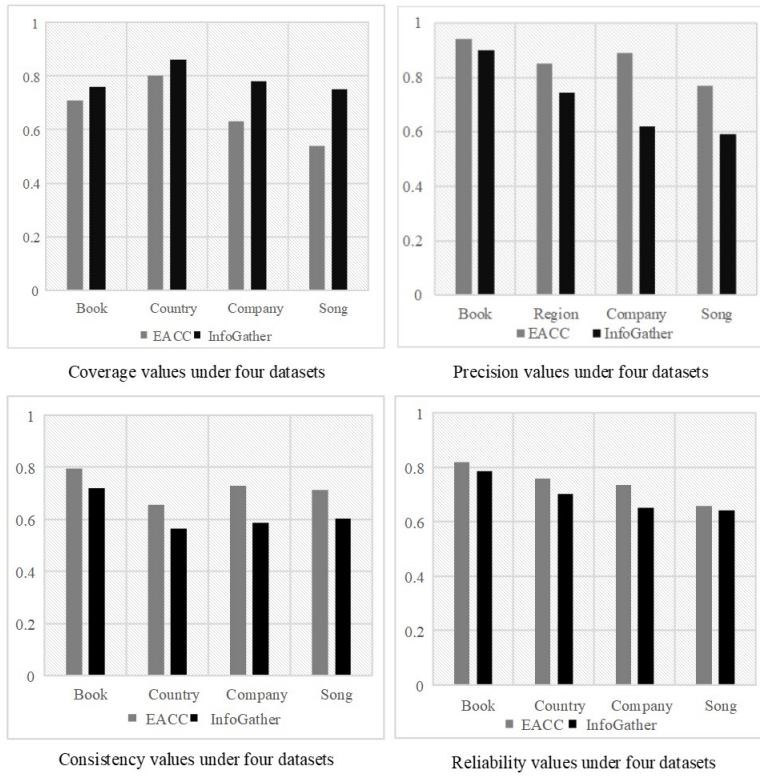

Figure 9. Entity augmentation comparison between EACC and InfoGather

From Figure 9, we have the following observations:

1. The coverages of InfoGather on four datasets are higher than that of EACC. EACC requires answer tables to have consistent matching relationships not only with each other but also with the query table, which certainly decreases the number of answer tables.
2. The average precisions (over all four datasets) of EACC and InfoGather are 0.86 and 0.71, respectively, demonstrating that EACC significantly outperforms InfoGather in precision. The consistency of EACC is also higher than that of InfoGather because InfoGather composes the result table from many data sources on a per-entity basis. And, InfoGather augments entities by splitting tables into several EAB relations, which can easily lead to entity inconsistency. Our EACC can get higher precision and higher consistency by considering semantic relevance and consistent matching relationships between tables.
3. Consequently, for reliability which is the harmonic mean of precision, coverage and consistency, our EACC method performs better than InfoGather. In Figure 9, the result set on Song dataset has the largest coverage difference between EACC and InfoGather. However, it has the minimum reliability difference between EACC and InfoGather. By observation, we find Song is the largest among four datasets. In fact, the larger the dataset is, the more answer tables there will be. Due to the restriction of consistent matching relationship, EACC will get less answer tables than InfoGather. Meanwhile, EACC will get higher precision and consistency. So, as the harmonic mean, the reliability difference between two methods on Song dataset will be smaller.

In summary, our method EACC performs much better than InfoGather in precision and consistency. Experimental results demonstrate that our entity augmentation framework has high accuracy and reliability, meanwhile ensuring also entity consistency.

5 RELATED WORK

At present, a large body of research work is about web search and data integration. Entity augmentation refers to extend attribute content based on entity or other known information, which helps people to obtain information they are interested in by web tables.

Web tables are important data source for gathering information by entity augmentation. Compared to other data sources such as knowledge base and crowdsourcing with human intelligence, web tables are more open and comprehensive. To our best knowledge, WEBTABLES system presented by Cafarella et al. is the first work on using the wealth of web tables [1, 2]. The authors extracted a large scale corpus of web tables and proposed several applications for such a corpus. They introduced AcsDB which enables several novel applications such as schema auto-complete, attribute synonym finding and join-graph traversal. According to user-supplied key-

words, WEBTABLES returns a list of web tables based on relevance ranking, and users can browse and filter useful information in the tables. WEBTABLES system gives us a chance to know the huge potential of web tables. Based on WEBTABLES system, Balakrishnan et al. display web table data as rich snippets in search engine results [15].

To integrate structured data, some researchers proposed to collect information from various data sources into a single table according to a set of keywords. Cafarella et al. proposed Octopus to integrate structured data [16]. Octopus used web search API to retrieve a ranked list of matching tables and integrated tables into a single table according to user interactions. MultiJoin algorithm was proposed to implement EXTEND operator in Octopus, which could find matching web tables for each queried entity independently and then cluster the web tables found. Pimplikar et al. presented a search engine which returned a multi-column result table in response to a keyword query without any known entities [17]. To achieve the table query, they mainly want to know if a web table is relevant to the query table, and if so, label each column of the web table with the query column to which it maps. The authors converted this task into a graphical model which took mappings of all candidate table columns into consideration. In the recent years, human intelligence has been introduced in the information processing and management. Park et al. implemented CrowdFill for collecting structured data from the crowd [18, 19], in which the interaction between workers and requesters is realized by Amazon Mechanical Turk. A partially-filled table is shown in CrowdFill to all participating workers, and workers contribute by filling in empty cells, as well as upvoting and downvoting data entered by other workers to return requesters the results collection.

There are also some studies for augmenting missing information in tables. Gupta et al. proposed an end to end system named WWT [20], which consolidated a table from a few example rows by harnessing the huge corpus of information-rich but unstructured lists on the web. InfoGather [7] is a system to augment binary tables. It identifies not only tables directly matched with the query table but also tables indirectly matched with the query table, what greatly improves the coverage of entity augmentation. InfoGather+ [21] is an improvement on InfoGather, which answers entity augmentation queries accurately for numeric and time varying attributes. It assigns labels for time and units of measurements of tables, and propagates labels among two connected nodes in semantic graph. However, InfoGather+ also argues that tables are entity-attribute binary relations, which hence leads to semantic fragmentation. Besides, Lehmberg et al. designed Search Join, a search engine achieving the search, join and composition of tables [8, 9, 10]. Eberius et al. use consistent set covering to solve the diversity of query results, and return users top-k result tables [22]. The authors proposed to process these queries in top-k fashion, in which they produced multiple minimal consistent solutions from which the user can choose to resolve the uncertainty of the data sources and methods used. However, all above systems consider the entity column as the only basis to augment another attribute column, ignoring the association between attributes and correlation between tuples.

For entity augmentation, a major challenge is the fact that many web tables have missing or non-informative column labels. To solve the problem, Braunschweig et al. proposed to identify and extract column specific information from the context of web tables [23]. They proposed a heuristic approach to extract column specific context information for relational tables on the Web. And, He et al. focus on automatic discovery of attribute synonyms [24]. They mainly consider attribute synonymy from query click logs and web table attribute name co-occurrences. The authors formalized the problem as an optimization problem on a graph, with the attribute names being the vertices and the positive and negative evidences from query logs and web table schemas as weighted edges. They developed a linear programming based algorithm to solve the problem. The method of discovering attribute synonyms is beneficial to improve the accuracy and coverage of entity augmentation. For some web tables with small size, Lehmberg et al. proposed that stitching web tables into a larger one could improve matching quality [25]. Above works are orthogonal to the problem of entity augmentation in this paper.

6 CONCLUSIONS

In this paper, we present the EACC framework to achieve consistent entity augmentation queries by using web tables as data source. In order to solve the entity inconsistency problem in existing technology, we propose the consistent matching relationship which should be hold between answer tables, and convert the problem of entity augmentation into the problem of building consistent γ-coverage cliques. Experimental results demonstrate that our entity augmentation framework has high accuracy and reliability, meanwhile ensuring the entity consistency. There are some future works, such as using parallel algorithms to improve the efficiency of big data processing, using the crowdsourcing platform to make full use of human intelligence to verify or correct result tables, and so on.

Acknowledgment

This work is supported by the National Key R & D Program of China (2018YFC0809800), and by the National Natural Science Foundation of China (61370060, 61673048).

REFERENCES

[1] Cafarella, M. J.—Halevy, A.—Wang, D. Z.—Wu, E.—Zhang, Y.: WebTables: Exploring the Power of Tables on the Web. Proceedings of the VLDB Endowment, Vol. 1, 2008, No. 1, pp. 538–549, doi: 10.14778/1453856.1453916.

[2] Cafarella, M. J.—Halevy, A.—Wang, D. Z.—Wu, E.—Zhang, Y.: Uncovering the Relational Web. Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008), Vancouver, Canada, 2008.

[3] Google Tables. Available at: https://research.google.com/tables.

[4] Gonzalez, H.—Halevy, A. Y.—Jensen, C. S.—Langen, A.—Madhavan, J.—Shapley, R.—Shen, W.—Goldberg-Kidon, J.: Google Fusion Tables: Web-Centered Data Management and Collaboration. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), 2010, pp. 1061–1066, doi: 10.1145/1807167.1807286.

[5] Gonzalez, H.—Halevy, A.—Jensen, C. S.—Madhavan, J.—Shapley, R.—Shen, W.: Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10), 2010, pp. 175–180, doi: 10.1145/1807128.1807158.

[6] Microsoft's Excel PowerQuery. Available at: https://office.microsoft.com/powerbi.

[7] Yakout, M.—Ganjam, K.—Chakrabarti, K.—Chaudhuri, S.: InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), 2012, pp. 97–108, doi: 10.1145/2213836.2213848.

[8] Bizer, C.: Search Joins with the Web. Christian Science Monitor, 2014.

[9] Lehmberg, O.—Ritze, D.—Ristoski, P.—Eckert, K.—Paulheim, H.—Bizer, C.: Extending Tables with Data from over a Million Websites. Semantic Web Challenge, International Semantic Web Conference, Riva del Garda, Italy, 2014.

[10] Lehmberg, O.—Ritze, D.—Ristoski, P.—Meusel, R.—Paulheim, H.—Bizer, C.: The Mannheim Search Join Engine. Journal of Web Semantics, Vol. 35, 2015, Part 3, pp. 159–166, doi: 10.1016/j.websem.2015.05.001.

[11] Sarma, A. D.—Fang, L.—Gupta, N.—Halevy, A.—Lee, H.—Wu, F.—Xin, R.—Yu, C.: Finding Related Tables. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), 2012, pp. 817–828, doi: 10.1145/2213836.2213962.

[12] Wu, W.—Li, H.—Wang, H.—Zhu, K. Q.: Probase: A Probabilistic Taxonomy for Text Understanding. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), 2012, pp. 481–492, doi: 10.1145/2213836.2213891.

[13] Wikipedia. Available at: http://www.wikipedia.org.

[14] Haveliwala, T. H.: Topic-Sensitive PageRank. Proceedings of the 11th International Conference on World Wide Web (WWW '02), 2002, pp. 517–526, doi: 10.1145/511446.511513.

[15] Balakrishnan, S.—Halevy, A.—Harb, B.—Lee, H. et al.: Applying WebTables in Practice. Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR '15), Asilomar, California, USA, 2015.

[16] Cafarella, M. J.—Halevy, A.—Khoussainova, N.: Data Integration for the Relational Web. Proceedings of the VLDB Endowment, Vol. 2, 2009, No. 1, pp. 1090–1101, doi: 10.14778/1687627.1687750.

[17] Pimplikar, R.—Sarawagi, S.: Answering Table Queries on the Web Using Column Keywords. Proceedings of the VLDB Endowment, Vol. 5, 2012, No. 10, pp. 908–919, doi: 10.14778/2336664.2336665.

[18] Park, H.—Widom, J.: CrowdFill: Collecting Structured Data from the Crowd. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14), 2014, pp. 577–588, doi: 10.1145/2588555.2610503.

[19] Park, H.—Widom, J.: CrowdFill: A System for Collecting Structured Data from the Crowd. Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion), 2014, pp. 87–90, doi: 10.1145/2567948.2577029.

[20] Gupta, R.—Sarawagi, S.: Answering Table Augmentation Queries from Unstructured Lists on the Web. Proceedings of the VLDB Endowment, Vol. 2, 2009, No. 1, pp. 289–300, doi: 10.14778/1687627.1687661.

[21] Zhang, M.—Chakrabarti, K.: InfoGather+: Semantic Matching and Annotation of Numeric and Time-Varying Attributes in Web Tables. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13), 2013, pp. 145–156, doi: 10.1145/2463676.2465276.

[22] Eberius, J.—Thiele, M.—Braunschweig, K.—Lehner, W.: Top-k Entity Augmentation Using Consistent Set Covering. Proceedings of the 27th International Conference on Scientific and Statistical Database Management (SSDBM '15), 2015, Art. No. 8, pp. 1–12, doi: 10.1145/2791347.2791353.

[23] Braunschweig, K.—Thiele, M.—Eberius, J.—Lehner, W.: Column-Specific Context Extraction for Web Tables. Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15), 2015, pp. 1072–1077, doi: 10.1145/2695664.2695794.

[24] He, Y.—Chakrabarti, K.—Cheng, T.: Automatic Discovery of Attribute Synonyms Using Query Logs and Table Corpora. Proceedings of WWW Conference, 2016, pp. 1429–1439, doi: 10.1145/2872427.2874816.

[25] Lehmberg, O.—Bizer, C.: Stitching Web Tables for Improving Matching Quality. Proceedings of the VLDB Endowment, Vol. 10, 2017, No. 11, pp. 1502–1513, doi: 10.14778/3137628.3137657.

Weijuan Sun received her Master degree in computer science from the Beijing Jiaotong University in 2018 and currently works in Baidu Netcom Science and Technology (Beijing) Co., Ltd. Her main research areas include web data integration and big data management.

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