Frontiers of Computer Science CN 10-1014/TP
DOI 10.1007/s11704-016-6319-3 ISSN 2095-2228
RESEARCH ARTICLE

Automatic Web-based Relational Data Imputation

Hailong LIU, Zhanhuai LI, Qun CHEN, Zhaoqiang CHEN

School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China

Front. Comput. Sci., **Just Accepted Manuscript** • 10.1007/s11704-016-6319-3
http://journal.hep.com.cn on December 8, 2016

Front.Comput.Sci.
DOI

RESEARCH ARTICLE

Automatic Web-based Relational Data Imputation

Hailong LIU (✉)[1], Zhanhuai LI [1], Qun CHEN [1], Zhaoqiang CHEN [1]

1 School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, P.R.China

Abstract Data incompleteness is one of the most important data quality problems in enterprise information systems. Most existing data imputing techniques just deduce approximate values for the incomplete attributes by means of some specific data quality rules or some mathematical methods. Unfortunately, approximation may be far away from the truth. Furthermore, when observed data is inadequate, they will not work well. The World Wide Web (WWW) has become the most important and the most widely used information source. Several current works have proven that using Web data can augment the quality of databases. In this paper, we propose a Web-based relational data imputing framework, which tries to automatically retrieve real values from the WWW for the incomplete attributes. In the paper, we try to take full advantage of relations among different kinds of objects based on the idea that the same kind of things must have the same kind of relations with their relatives in a specific world. Our proposed techniques consist of two automatic query formulation algorithms and one graph-based candidates extraction model. Several evaluations are proposed on two high-quality real datasets and one poor-quality real dataset to prove the effectiveness of our approaches.

Keywords data incompleteness, imputation, world wide web, query formulation, candidate selection, semantic relation

1 Introduction

Data incompleteness is one of the most important and inevitable problems in many areas like commercial, science computing, medical treatment. To improve data quality, incomplete data imputation techniques have attracted more attention in the recent decade. They aim at providing estimations of missing values by reasoning from observed data [1]. There are three kinds of data imputation methods. One imputes missing values only by virtue of the source data, the other two try to get the missing values from some extended information sources or some human experts. Most existing data imputation methods [2–6] are the former. They purely use some distribution characteristics or some constrains among different attributes to deduce approximate values and look on them as the final imputing values. Their goal is always to alleviate or eliminate the influence of some incomplete attribute values, but not to find real values for them. In many situations, approximation may be far away from the truth. Furthermore, most of these approaches require that there must be some duplicate data, or else they will not work well. Here we show some examples.

Example 1: Consider the *income* of a specific person in the tax systems. His/her annual bonus always accompany his/her normal salary in December. If we use his/her *average income* in the first 11 months to substitute for his/her *income* in December, the person's tax fees may be cut down to a large extent. It is obviously unreasonable.

Example 2: Consider the *publisher* of a specific book whose *isbn* value is *978-3-642-16261-9* in Table 1. We may find the functional dependency like *isbn→publisher*. However, we cannot use such constrains to deduce its *publisher* value because there is only one tuple with the *isbn* value *978-3-642-16261-9*. In this case, we can only use the most frequently appearing value as the imputing value. Obviously, it may be not correct.

Importing external information can compensate for the lack of enough information in relational tables and improve

Received month dd, yyyy; accepted month dd, yyyy

E-mail: *liuhailong@nwpu.edu.cn*

the accuracy of data imputing. External information source can be a human expert, a special knowledge base, a master data [7] database or the World Wide Web (WWW). According to Google's report, till July 2008, they had found 1 trillion Web sites and tens of billions of new Web pages appeared every day [8]. The WWW has become the biggest information source in the world. In comparison with other information sources and human experts, the WWW covers much more areas and contains much more information sources. Consequently, it is much more knowledgeable. If we can get needed information from the WWW efficiently, it can help us impute more incomplete data and impute more accurately.

Web search engines are the most frequently used information retrieval tools. There are always two main steps in the process of web-search-engine-based data retrieval approaches. One is to collect the most related Web documents by means of a specific search engine according to a set of keywords. The other is to extract and select the most relevant object values from the retrieved Web documents. Accordingly, to propose effective Web-based relational data imputation techniques, we must overcome the following two challenges.

1. As we know, different queries will bring different search results. To retrieve a missing value via Web search engines, we need effective queries. Although may be inadequate, existing information is the best option for improving data quality of a database. If we know sematic context relations among the complete attributes and the incomplete attributes, creating an efficient query may be not difficult. Unfortunately, such kind of relations are always unclear and cannot be determined from the schema data directly. As a result, how to use existing information to form an efficient query becomes a challenge. For example, when retrieving the *publisher* value of the tuple whose *isbn* value is *978-3-642-16261-9* in Table 1, the most proper query pattern which consists of other observed attributes (e.g., *isbn, title, author,* and *year*) is unknown.
2. As the most commonly used information retrieval tools, Web search engines are designed to retrieve the most related documents but not values. However, in relational data imputation tasks, we always need values, such as a *publisher* for a book in Table 1. Obviously, there is a gap. To the best of our knowledge, there are few effective tools for Web objects retrieving. Therefore, the second challenge is how to find the most related values from the retrieved snippets. For a user without any background knowledge, it is always difficult to define, select and get proper factors which will have effect on objects extraction and ranking. And if improper factors are selected, the users may get poor results. So a much more automatic approach should be provided.

To tackle these challenges, we propose a set of approaches in our Web-based relational data imputing system so as to automatically generate query keywords and automatically suggest the most probable candidate values for an incomplete relational database. We make the following contributions.

1. We propose two query formulation algorithms. One generates query keywords based on a functional depen-

Table 1 Incomplete book records extracted from DBLP dataset(part of)

isbn	title	author	publisher	year
978-3-642-16261-9	Nonlinear Dynamics in Human Behavior	Raoul Huys	NULL	2011
978-3-642-17553-4	Intelligent Video Event Analysis and Understanding	Jianguo Zhang	NULL	2010
NULL	Mobile Computation with Functions	Zeliha Dilsun Kirli	Springer	2002
.....	.....	.....	.....	.....
NULL	Omringd door Informatica	Bennie Mols	CWI	2010

Table 2 Complete book records extracted from DBLP dataset(part of)

isbn	title	author	publisher	year
978-0-387-30768-8	Encyclopedia of Machine Learning	Claude Sammut	Springer	2010
1-930708-38-6	Database Integrity: Challenges and Solutions	Jorge Horacio Doorn	Idea Group	2002
978-0-387-30236-2	Computer Viruses and Malware	John Aycock	Springer	2006
.....	.....	.....	.....	.....
978-0-521-56876-0	Computational principles of mobile robotics.	Michael R. M. Jenkin	Cambridge University Press	2000

dency whose right hand is the incomplete attribute. The other tries to generate query keywords by learning a proper composition of the observed values and attribute names. Our goal is to automatically output a query pattern with the highest *FITNESS* score, which is defined in Section 3.1.

2. We propose a graph-based entity ranking model to extract imputing candidates. Different from traditional entity extraction methods which accumulate the weights of different factors, our graph-based model tries to learn semantic relation patterns among different objects and use pattern matching methods to extract target values. The objects we used include the query keywords, the target attribute values and other related objects in the Web snippets. All relations are organized in graphs and graph pattern matching is used to find the most proper values. In this manner, users just need to input their tables with some complete tuples. Here a good knowledge of what factors will affect the selection of the candidates is not necessary.
3. We implement and evaluate a Web-based data imputing prototype system which employs our new query formulation methods and object extraction models. The evaluation results show that the proposed methods are effective and can bring more accurate imputation for a relational table.

The rest of this paper is organized as follows. We formulate data imputation problem in Section 2. We introduce the Web-based imputing framework, detail the query formulation techniques and candidates ranking model in Section 3. The experimental study is conducted in Section 4. We survey and analyze related works in Section 5. Finally, we conclude this paper in Section 6.

2 Problem Definition

Let U represent the set of the attributes of the table, t represent a tuple, *t[X]* represent the X attribute value of t, D represent a database, the completeness of D can be described as *Formula 1*.

$$Completeness(D) = \begin{cases} 0, if(\exists X \exists t(t \in D \wedge X \subset U \\ \quad \wedge t[X] = NULL)) \\ 1, otherwise \end{cases} \tag{1}$$

Let $t_{true}[X]$ represent the real value of the attribute X. ϵ is a small threshold value. Repairing the incompleteness error of *t[X]* means looking for a repairing model M to make the repaired value satisfy *Formula 2*. ϵ is the indicator of the bias of M. In this paper, we set $\epsilon = 0$. *It means that we want to retrieve the real but not an approximate value of each missing attribute value*.

$$\frac{|t_M[X] - t_{true}[X]|}{|t_{true}[X]|} \leq \varepsilon \tag{2}$$

Let $C(t, A)$ represent the condition of accurately imputation on the incomplete attribute A, R represent data dependency rules. In rule-based imputing techniques, two preconditions must be satisfied. One is that there must be at least one data dependency rule pointing to the target attribute. The other is that there must be at least one reference tuple t' which satisfies $t'[A] \neq NULL$. Otherwise, we can just use the distribution characteristics of A or other attributes in U to deduce an approximate value for $t[A]$. $C(t, A)$ is defined in *Formula 3*.

$$C(t, A) = \begin{cases} 1, if(\exists X \exists r((X \subset U \wedge r \in R \wedge r : X \rightarrow A) \\ \quad \wedge \exists t'(t' \in D \wedge t'[X] \neq NULL \wedge \\ \quad t'[A] \neq NULL \wedge t[X] = t'[X]))) \\ 0, otherwise \end{cases} \tag{3}$$

Because the attribute values of some tuples may be unique, which means it is difficult to find a reference tuple to impute the missing values, the preconditions in *Formula 3* are obviously too secrete. Missing values must hide somewhere, most possibly on the WWW. If we can find them, accurate imputation can be achieved. So we can relax the conditions and adjust the formula as *Formula 4*.

$$C(t, A) = \begin{cases} 1, if(\exists X \exists r(X \subset U \wedge r \in R \wedge r : X \rightarrow A \wedge \\ \quad t[X] \neq NULL) \wedge IsInWeb(t[A]) \\ 0, otherwise \end{cases} \tag{4}$$

$IsInWeb(t[A])$ is an indicator function. Only when *t[A]* exists in the WWW, it returns true. To retrieve the missing values, some observed values are necessary. So R should be relaxed to represent some relations between X and A. Obviously, Formula 4 is the necessary condition for Web-based attribute value imputation techniques. *Our goal is to develop a mechanism to learn R and pick out the missing values effectively and automatically from the WWW*.

We will use an example to help us explain our approaches more clearly in later parts of our paper. The example is as following.

Example 3: Table 1 and Table 2 are two subsets of the book record of DBLP dataset [9]. Table 1 contains some incomplete tuples and Table 2 is the training set with complete

tuples. *Our goal is to impute the missing values of the publisher attribute in Table 1.*

3 Web-based imputation approach

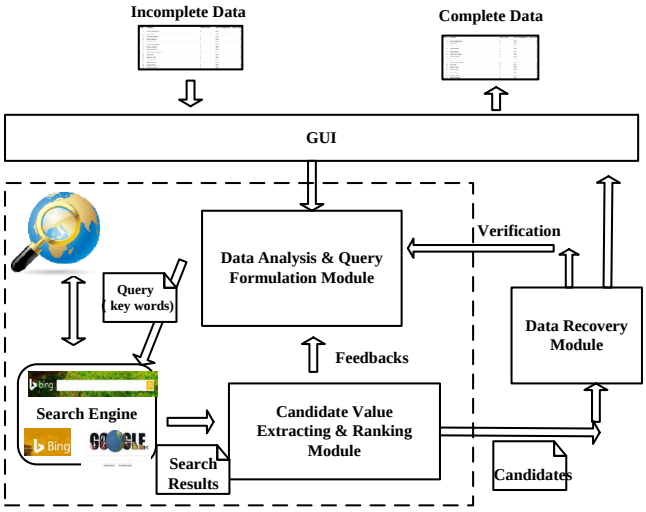


Fig. 1 Automatic Web-based relational data imputation framework

Our automatic Web-based imputation platform is illustrated in Figure 1. It contains three modules, including *Data Analysis & Query Formulation Module* for missing value detection and query formulation, *Candidate Value Extracting & Ranking Module* for candidate values extracting and ranking, *Data Recovery Module* for missing value substitution and verification. Next we will introduce each module one by one.

In the *Data Analysis & Query Formulation* module, we firstly detect whether there are some incomplete tuples. If so, we will construct a training set and then learn a proper query pattern based on it. In Example 3, the query pattern learnt may be (*"publisher"*, *t*[*isbn*], *t*[*year*]). Then, we can use *"publisher, Encyclopedia of Machine Learning, 2010"* as the keywords of Bing to fetch related Web pages. In this module, we will employ two new query formulation algorithms. One is rule-based, the other is genetic-based.

In the *Candidate Value Extracting & Ranking* module, we extract candidate values from the Web snippets generated in the former module and then rank these values. Here, we utilize some widely used techniques to extract related objects, like the *named entity identification techniques* used in Ephyra [10]. Our focus is on how to pick out the most proper ones from all the extracted candidates. To do this, we intend to make use of semantic relations among different kinds of objects across a relational table and the WWW. For instance, in Example 3 we will utilize the relations among the values of *publisher, isbn, year* and other objects from the WWW like *co-author names, editor names* to rank the candidates and finally pick out the top ranked candidate *"Springer"* for the first tuple in Table 1.

In the *Data Recovery* module, we will use the top ranked value to fix the corresponding tuples. For instance, *"Springer"* will be used as the final value of the incomplete tuple.

3.1 Query Formulation

3.1.1 Problem Definition

According to Formula 4, observed values in the same tuple are necessary. Consequently, the essence of query formulation problem is to find a proper composition from the observed values and the attribute names. The query formulation problem can be defined as *Definition 1*.

Definition 1. *Assume that the attribute names of a relational table are described in the form* $\mathcal{A}(A_1, A_2, ..., A_i)$ *and the attribute values of a tuple are described in the form* $\mathcal{V}(v_1, v_2, ..., v_i)$. *Assume that some values of* A_m *are missing. The* ***query formulation problem*** *can be defined as selecting a subset of* A_i *and* v_i *to create a query pattern which is very likely to obtain the missing values of* A_m.

When creating a query, the more items are selected from R, the more limitations are appended to. It may reduce down the possibilities to find the missing values from the WWW. And, the *attribute names* may not always co-appear with the *attribute values*. So, when formulating a query for a missing value, we only select the *attribute name* of the target missing value from $\mathcal{A}$. A ***query patten (QP)*** for A_m can be defined in *Definition 2*.

Definition 2. $QP(A_m) = A_m, v_i, ..., v_n$

Intuitively, two factors should be taken into consideration to judge whether the query pattern $QP(A_m)$ is proper. One is how many tuples it can cover and the other is how tightly it can cover. So the $fitness$ value of $QP(A_m)$ can be calculated as Formula 5. Here *SUPPORT* represents the ratio of the training tuples which supports the query pattern and *CONFIDENCE* represents the average ratio of the result snippets which contain the expected target values.

$$FITNESS(QP(A_m)) = \alpha \times SUPPORT + (1-\alpha) \times CONFIDENCE \quad (5)$$

Definition of *SUPPORT* is as Formula 6. In the formula, n represents the number of the tuples whose target values can be retrieved by the query pattern and N represents the total number of the training tuples. Assume 1000 tuples are selected from Table 2 as the training data and 800 tuples' publisher values are found by virtue of the query pattern q. The $SUPPORT$ value of q will be $800/1000 = 0.8$.

$$SUPPORT = \frac{n}{N} \tag{6}$$

Definition of *CONFIDENCE* is as Formula 7, in which n_i represents the number of the snippets which contain the expected value of the tuple i, N_S represents the number of the returned snippets. Given a query patter q. Assume each time we only consider 100 snippets returned by the web search engine. For the first tuple, 20 snippets contain its publisher value. For the second, 40 snippets do. For the third tuple, only 10 do. If the training data set only contains three tuples, the $CONFIDENCE$ value q is $\frac{\frac{20+40+10}{100}}{3}$.

$$CONFIDENCE = \frac{\sum_{i=1}^{N} \frac{n_i}{N_S}}{N} \tag{7}$$

The greater the *SUPPORT* value is, the more training tuples support the query pattern, which means the more probabilities we can retrieve the missing attribute values by the query pattern. The greater the *CONFIDENCE* is, the larger number of the result snippets contain the target value, which means the more confidence the values retrieved are the proper values. α and $(1-\alpha)$ are the coefficients of *SUPPORT* and *CONFIDENCE*. We think that the *SUPPORT* factor is more important than the *CONFIDENCE* factor, because it is difficult to require a query with high *CONFIDENCE* value, especially for the information that is not widely distributed in the WWW. Other factors like the ranking strategies of Web search engines and the number of selected retrieved snippets will also affect *CONFIDENCE* value. So we set α greater than $(1-\alpha)$ in our model, such as α=0.8 and $(1-\alpha)$=0.2.

We can simply try all compositions of the sets $\mathcal{A}$ and $\mathcal{V}$ for each training tuple, then select the composition with the highest *FITNESS* score as the final query pattern. Let $P(n,i)$ represent the number of $i-permutations$. The number of the compositions of a set which contains n attributes is $\sum_{i=1}^{n} P(n,i)$. Each composition determines a query pattern. To pick out the best one, we need compute the $FITNESS$ values of all query patterns. According to Formula 5, we need employee a query pattern on each training tuple. So, on m training tuples, the number of queries will reach $\sum_{i=1}^{n} P(n,i) \times m$. Since each try need call Bing or Google Web Search API and it is not free, this strategy is time-consuming and money-consuming. To guarantee imputation accuracy, a proper number of training tuples are need. It means that m is always fixed. On this occasion, we can only cut down $\sum_{i=1}^{n} P(n,i)$. In other words, we need try less compositions so as to cut down the cost.

The observed values and the missing one are always interconnected in some semantic manner on the WWW. So if we can identify these semantic inter-connected compositions, it can help us find a proper query quickly. In other words, a proper query should come from the most intensive semantic inter-connected compositions. Based on this idea, we propose two methods. One is *rule-based query formulation algorithm* which directly utilizes functional dependencies as query patterns to create queries. The other is *genetic-based query formulation algorithm* which uses machine learning techniques to find some hidden connections between the observed values and the missing one, then use a proper one of them to create queries.

3.1.2 Rule-based Query Formulation Algorithm

Algorithm 1 Rule-based query formulation algorithm

Input:
TA: target attribute
TD: training tuples
FT: fitness threshold
FD: functional dependencies

Output:
BI: best query pattern.

```
1:  FD = FindFDsByTane(TD, TA);
2:  bestFitness = 0.0;
3:  for each fd in FD do
4:    query = GenerateQueryPattern(fd, TA);
5:    for each t[i] in TD do
6:      keywords = GenerateQueryKeywords(query, t[i]);
7:      results[i] = BingSearchAPI(keywords);
8:    end for
9:    fitness = ComputeFitness(TD, results);
10:   if fitness >= FT and fitness > bestFitness then
11:     bestFitness = fitness;
12:     BI = query;
13:   end if
14: end for
15: return BI;
```

A functional dependency rule $fd : X \rightarrow Y$ means that a set of attributes X uniquely determine another set of attributes Y. If all attributes in X are complete and the incomplete attribute is in Y, we can consider values of the attributes in X to be the query keywords for retrieving values of the incomplete attribute. If we get a related functional dependen-

cy fd_1 : $isbn \rightarrow publisher$ in Example 3, we can generate the query pattern QP_1 : ("*publisher*", t[*isbn*]) and then use it to retrieve missing *publisher* values. There may be several such functional dependencies and consequently we will get several query patterns. On this occasion, we select the one with the highest $FITNESS$ value as the final query pattern. Assume we get another related functional dependency fd_2 : $(title, author, year) \rightarrow publisher$ in Example 3. Besides QP_1 : ("*publisher*", t[*isbn*]) , we can also get QP_2 : ("*publisher*", t[*title*], t[*author*], t[*year*]) according to fd_1 and fd_2. Based on the training tuples in Table 2, we can work out the *FITNESS* values of QP_1 and QP_2. If $FITNESS(QP_2) > FITNESS(QP_1)$, we select QP_2 as the final query pattern. The algorithm is detailed in Algorithm 1. In our work, we use TANE [11] to find functional dependencies and then use them directly to create queries.

3.1.3 Genetic-based Query Formulation Algorithm

Give a lower bound FT. If the *FITNESS* value of a query pattern overtakes FT, users will consider it is a proper candidate. Here query formulation problem changes to finding a composition of r and $\mathcal{R}$ whose *FITNESS* score is not lower than FT. Genetic algorithm [12, 13] is a good choice in the situations where the search space is large, complex and domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space. In this section, we try to adjust the traditional genetic algorithm to learn a proper query pattern.

Genetic algorithm generates solutions by means of the techniques inspired by natural evolution process. It always includes four main steps, such as *inheritance*, *mutation*, *selection*, and *crossover*. At the beginning of the genetic-based query formulation algorithm, we randomly set each attribute value in a tuple selected or not to generate a specified number of individuals. These individuals form the initial population. We use 0 and 1 to represent a attribute value is selected or not respectively. In Example 3, 0011 represents that *isbn* and *title* are not selected but *author* and *year* are. The initial population for retrieval *publisher* values in Table 1 can be $P = \{0011, 1100, 1001, 0001, 1001\}$. Then we look on each individual in P as a query pattern. We can calculate the $FITNESS$ value of each individual and pick out the best one. If the best $FITNESS$ dose not overtake FT, we will execute use *selection* operation to select excellent ones to generate a new population P_{new}. In this stage, we apply the *roulette betting strategy*. The idea is that the higher the $FITNESS$ value of an individual is, the more possibility it will be selected to reproduce later generations. Accordingly, the possibility of selecting a specific individual I can be calculated as $\frac{FITNESS_I}{\sum_I^{I \in P} FITNESS_I}$. In the *crossover* stage, we apply *uniform crossover strategy* and randomly cross two individuals in P_{new}. And then, we randomly reverse a bit in the *mutation* stage . When the best *FITNESS* value of the individuals in a population reaches a given threshold, the algorithm will exit. Assume that we set $FT = 0.7$ in Example 3. The *FITNESS* value of the composition *0101* exceeds 0.7, so the evolution will stop and the composition (*title,year*) should be outputted. It means that we can use the keywords (*"publisher", Nonlinear Dynamic in Human Behavior,2011*) to retrieve the *publisher* value of the first tuple in Table 1. The algorithm is detailed in Algorithm 2.

Algorithm 2 Genetic-based query formulation algorithm

Input:
TA: target attribute
TD: training tuples
FT : fitness threshold
MG: maximum number of generations
Output:
BI: best attribute subset having the best fitness.

```
1:  generate initial population P randomly;
2:  bestFitness = 0.0, currentGen = 0;
3:  while TRUE do
4:    for each I in P do
5:      query = GenerateQueryPattern(I, TA);
6:      for each t[i] in TD do
7:        keywords = generateKeywords(query, t[i]);
8:        results[i] = BingSearchAPI(keywords);
9:      end for
10:     fitness = ComputeFitness(TD, results);
11:     if fitness > bestFitness then
12:       bestFitness = fitness;
13:       BI = query;
14:     end if
15:   end for
16:   if bestFitness >= FT or currentGen > MG then
17:     break;
18:   else
19:     P = selection(P);
20:     P = crossover(P);
21:     P = mutaition(P);
22:     currentGen+ = 1;
23:   end if
24: end while
25: return BI;
```

3.2 Candidates Extraction

The attributes of a relational table are always context discrete in the WWW. Even it is not so, the context relations among them always cannot be extracted directly from the schema of a table. Consequently, it is difficult to automatically form an extraction pattern like [14]. In this section, we want to take full advantage of the relations among different kinds of objects in the WWW. We will propose a new graph-based information extraction approach. The main processes of our approach include *related objects collection* and *candidates selection*. In the first process, we will collect related objects from retrieved Web snippets using existing techniques. In the second process, we firstly *organize the collected objects into relation graphs* and then *rank them so as to find the target values*.

3.2.1 Related Objects Collection

We utilize natural language processing techniques [15] and named entity identification techniques [16] which are used in Ephyra [10] to collect related objects. These techniques can help us extract 154 kinds of entities [10].

The collected objects can be divided into three categories according to their roles, including *context entities(CE)*, *target entities(TE)* and *intermediate entities(iE)*. Their definitions are listed below.

Definition 3. ***Context Entities(CE)*** *are the entities which are involved in the query keywords. They will be marked with squares in our graph models.*

Definition 4. ***Target Entities(TE)*** *are the entities with the same type of the incomplete attributes. They will be marked with the composition of circles and triangles in our graph models.*

Definition 5. ***Intermediate Entities(iE)*** *are the entities that are not only excluded by the query keywords, but also with the different type of the* ***Target Entities(TE)****. They will be marked with circles in our graph models.*

Assume the query pattern learnt in Example 3 is (*"publisher", t[title], t[year]*). For the last tuple in Table 2, *"Computational principles of mobile robotics"* and *"2000"* are *CE* entities, collected publishers like *"Kluwer Academic"* and *"Cambridge University Press"* are *TE* entities, Other entities like the name of the editors are *iE* entities. Part of the entities of this example are listed in Figure 2.

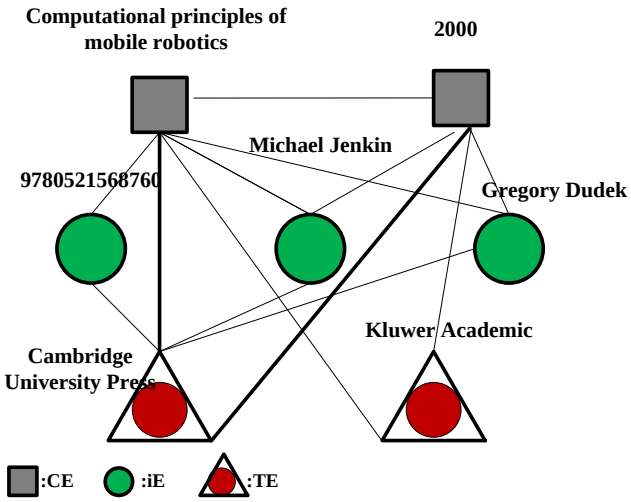


Fig. 2 A relation graph example (*co-appearance*)

3.2.2 Candidates Selection

As shown in Figure 2, we may get several *TE* entities. Only one of them should be selected as the target value of an incomplete tuple. A direct way is to select the candidate which appears most frequently in the WWW as the target one. In Example 3, if 20 snippets contain *"Springer"* and 10 snippets contain *"Kluwer Academic"*, we will rank the two publishers by the order *"Springer"*, *"Kluwer Academic"*. We name this method *weight-based ranking algorithm*.

Weight-based ranking algorithm can not always work well, because the candidates which appear most frequently may not be the correct candidates. The reason is that the snippets we retrieved by a Web search engine always contain not only the information we really needed but also other homologous information. The number of the homologous information may exceed the number of the target information. For example, we want to get *"Cambridge University Press"*, but *"Springer"* may appears more times since most other books in the same Web page are published by Springer. In this situation, we may get incorrect results.

In the physical world, if we cannot find something directly, we may try to find other things related to the target and then follow them to grasp the target thing. For example, the policemen always try to find a criminal's girl friend or parents and then follow them to catch him. This scenario appears frequently in many films. A tuple is a kind of description of an object or an event of the physical world. So we can retrieve missing values in a table in the same way. The idea is

that we firstly try to find the relation pattern between the observed attributes and the incomplete attributes, and then use the pattern to help us find the missing values.

In Section 1, we have learnt that which values may be related to the missing value but not known how do they associate with each other. Obviously, the vague relations in a query pattern will help us little in accurately candidate extraction. We need to find a pattern which can indicate more details in the relations between the observed values and the missing ones.

The same kind of objects always have the same kind of characteristics in the natural world. For example, all kinds of fruits are *juicy*, *sweet*, *rich in nutrients* and *come from some kind of plants*. We can generalize these characteristics from the attributes of some fruits like *apples*, *pears*, *oranges*, and conclude that all *peaches* must follow these generalized characteristics. All tuples in the same relational table are homogeneous. So, in the same way, we can learn relation patterns between complete attributes and incomplete attribute based on a set of complete training tuples. Relations between the observed values and the missing values of a specific incomplete tuple must follow the same relation pattern.

Graph is a proper and widely used model for data relation description, in which the nodes represent objects and the edges represent relations. Accordingly, we propose a graph-based candidates ranking model to help us select the most proper candidates. In our model, nodes represent the CE, iE and TE entities, and edges represent relations between these entities. And furthermore,each graph start from several CEs and end at several TEs. In all graphs, CEs are generated according to the same query pattern QP. It means that all graphs have the same number and kinds of CEs.

Because each tuple is related to a specific physical object or event, we need create a graph for each training tuple and name it a *training graph*. Although each graph may be special, we can generalize some common characteristics from a group of *training graphs* for the reason that they represent the same kinds of physical objects. Based on this idea, we propose an algorithm to generalize the graph pattern so as to find the general relations between the observed attribute values and the missing attribute values.

The framework of our graph-based ranking algorithm is shown in *Algorithm 3*. Main procedures include *creating relation graphs*, *generalizing the relation pattern from training graphs* and *ranking candidates for incomplete tuples*. Details of each procedure are introduced below.

Creating relation graphs. All entities and relations are extracted from the snippets return by Bing search API. If two objects relate to each other, firstly they must co-appear in the same snippet or Web document. So *co-appearance* is the basic semantic relation among the collected objects. There are always some objects which may not co-appear with others or co-appear just few times. In other words, they are not tightly connected with others. These objects may help us little. Consequently, we ignore these objects and the relations stemming from them when creating a graph. To do this, we firstly utilize *FPTree* algorithm to pick out the frequent item sets and just look on them as the nodes of each graph. After that, we mine k frequent appearing couples so as to find the objects really correlated with each other. Figure 2 is a training graph for Example 3. Some links between different nodes may appear several times in different snippets. We mark this kind of paths in a training graph with thick lines, just like Figure 2.

Algorithm 3 Graph-based candidates ranking algorithm

Input:
TA: target attribute
TD: training tuples
QP: the query pattern used to retrieve the candidates
t : the tuple whose incomplete attribute is TA

Output:
TEL: ranked proper values for TA

```
1: graphs=CreateTrainingGraph(TD,TA,QP);//Create the relation graphs of each training tuple
2: PG=Generalize(graphs);//Generalize training graphs
3: CG=CreateCandidateGraph(t,TA,QP);//create the relation graph of the incomplete tuple t
4: TEL=CandidatesSelection(PG,CG,QP,TA);//rank the imputing candidates
5: return TEL;
```

In Figure 2, relations attached to each edge are *co-appearance*. It is certain that the more detailed meanings the edge represents, the more precision we can get to extract the relation pattern. So if detail sematic relations like *is the member of* or *belongs to* can be extracted and attached, it will help us greatly. In our paper, we extended ReVerb [17] to extract binary relations expressed by verbs, prepositions, punctuation.

The whole process of creating a relation graph is detailed in Algorithm 4. Main procedures include *extracting related items, picking out k frequent items, extracting relation between the objects of a k frequent pair*. Mechanisms of these procedures have been mentioned earlier.

Generalizing the relation pattern from training graphs. After *training graphs* are created, we generalize the common characteristics of them so as to find the relation pattern. It is obvious that relation graph is composed of several paths and

Algorithm 4 Create a relation graph

Input:
ROL: Related Objects list
td: a training or target tuple
k: required frequency
QP: the query pattern used to retrieve the candidates

Output:
GRAPH: the relation graph

```
1: ROL=RetrieveEntitiesFromWeb(td,QP);
2: FrequentItems = FindKFrequentItmes(ROL,k,2);
   //Find k frequent item couples in ROL
3: for each FP IN FrequentItems.itempair do
4:    ExtractSemanticRelation(FP.first,FP.second)
      //Extract Sematic Relation between nodes
5:    ClusterSemanticRelation(FP.first,FP.second);
      //Clustering the relations in different forms
      edge=NewEdge(FP.first,FP.second);
6:    if edge IS NOT IN edges then
7:       edges=edges.addedges(edge);
         //add an new edge with semantic relation
8:       nodes=nodes.add(FP.first,FP.second);
         //add new nodes
9:    end if
10: end for
11: return GRAPH(nodes,edges);
```

each starts from a CE and ends at a TE. So *relation pattern generation problem* can be defined as *finding the path patten from a group of CEs to a specific TE*.

Let ce_i represent a CE entity and te represent the TE entity. From Figure 2, we can see that there are several paths from ce_i to te. Our goal is to pick out a path which most objects will follow, so only one path from ce_i to te will be selected when generalizing the path pattern. Obviously, the more times two items co-appear the more tightly they relate to each other. In this paper, we use *weight* to evaluate the co-appearance frequency of two items. The path with the largest *weight* value are the most possibly to be a path pattern. We use *PATH(ce_i,te)* to represent this kind of path and define it in *Definition 6*. We marked all members of *PATH(ce_i,te)* with thick lines, just like Figure 2.

Definition 6. PATH(ce_i, te) *is the path with the largest* ***weight*** *value in the relation graph and it is from a context entity node ce_i to the target entity te directly or passing by several intermediate entities.*

Let G_i:$ce_i \xrightarrow{G_i:r_1}$ ..... $\xrightarrow{G_i:r_i}$ te represent *PATH(ce_i,te)* in the graph G_i, $NodeType(ce_i)$ represent the type of ce_i and R_i represent the generalized form of the semantic relation. The form of the path pattern of *PATH(ce_i,te)* can be defined as $NodeType(ce_i) \xRightarrow{R_1}$ ..... $\xRightarrow{R_i} NodeType(te)$. Here, *path pattern generalization problem* can be defined as *finding a pattern which can represent the characteristics of the PATH(ce_i,te) in most graphs*.

PATH(ce_i,te) in different G_i may be different . We collect *PATH(ce_i, te)* from all training graphs and then divide them into several clusters. We select the biggest cluster of *PATH(ce_i, te)* to represent the general pattern of *PATH(ce_i, te)* . When clustering *PATH(ce_i, te)*, *Path Length*, *Node Type*, *Node Sequence* and *Semantic Relations* should be taken into account. We use *SIM(p_i, p_j)* in Formula 8 to compute the similarity of *PATH(ce_i, te)* in two different graphs. In Formula 8, we accumulate two kinds of similarities. One is the similarity of the path without edge semantics, in which we compare the similarities of the *nodetypes* sequently. The other is the similarity of the edge semantics, in which we compare the similarities of the edge semantics sequently. As known, semantics of an edge may be described in different forms in different graphs. Therefore, we will generalized semantics of the edges and use the generalized sematic pattern to represent the sematic of each edge when creating a graph. The generalized sematic pattern is a cluster. So, when computes the similarity of the edge semantics, we compute the similarities of two semantic clusters. Then we can divide all *PATH(ce_i, te)* into different clusters according to the similarities among them and a given threshold. We pick out the biggest cluster and look on the whole cluster as the path pattern. In Formula 8, $Ndis$ is used to compute the similarity of each pair of nodes and $Edis$ is used to compute the similarity of each pair of edges. When accumulating similarities, we give node similarities more weights. In this paper, we manually set $\alpha = 0.8$ and $\beta = 0.2$.

$$SIM(p_i, p_j) = \frac{\sum_{m=1}^{Max(p_i.length, p_j.length)} NDis(p_i.N_m, p_j.N_m)}{Max(p_i.length, p_j.length)} \times \alpha + \frac{\sum_{m=1}^{Max(p_i.length-1, p_j.length-1)} EDis(p_i.E_m, p_j.E_m)}{Max(p_i.length-1, p_j.length-1)} \times \beta \tag{8}$$

$$NDis(p_i.N_m, p_j.N_m) = \begin{cases} 1 & \text{if } p_i.N_m.\text{NEType}==p_j.N_m.\text{NEType} \\ 0 & \text{if others} \end{cases} \tag{9}$$

$$EDis(p_i.E_m, p_j.E_m) = \begin{cases} 1 & \text{if } p_i.E_m.\text{cluster}==p_j.E_m.\text{cluster} \\ 0 & \text{if others} \end{cases} \tag{10}$$

Candidate Selection. Till now, we have gotten the query pattern QP and the pattern graph PG. For a tuple t with a missing value te, we can use its observed values and QP to create a query and then a relation graph. The graph created from an incomplete tuple is named *candidate graph(CG)*. If the amount of the snippets is enough, the missing value must exist somewhere in CG. In this stage, our goal is to find it.

The path patterns from each CE to the incomplete attribute TE have been found and described in PG. The following work is to match the candidate graph and PG. To do this, pathes from each ce_i to all the members of TE which are with the highest *weight* values should be picked out from the CG. These pathes are recorded in the set $PATH_{CE} = \bigcup_{i=0,j=0}^{i=|CEs|,j=|TEs|}\{PATH(ce_i, te_j)\}$. Then, we compute the similarities between a member of $PATH_{CE}$ and the path pattern which starts from the same kind of CE in PG . The similarities are computed by the function $SIM(p_i, p_j)$ defined in Formula 8. The path $PATH(ce_i, te_j)$ with the biggest similarity is selected and the te_j in this path is marked as a candidate. All the candidates are ranked by the frequency of being marked and the top 3 ranked candidates are suggested to be the target imputing values.

4 Experiment Results

We have implemented all our presented approaches in Java and integrated them into a relational data imputation platform. In this section, we plan to evaluate the effectiveness of these approaches. It is obvious that the quality of observed values will have effects on the retrieval of the missing ones. In consideration of this, we select three real datasets. Two are with high original quality but one is not. The accuracy of imputing different types of values may be different. Consequently, we purposely set various types of values $NULL$ in all datasets, including different text values (*publisher*,*email*) and different numeric values (*isbn*, *year*, *zipcode*).

Details of these datasets are listed as follows.

1. **DBLP** [9]. DBLP dataset contains the information of some publications in various areas of computer science. We select five attributes of about 14498 books and import the refined tuples into MySQL. The attributes imported include *isbn,title,author,publisher,year of publication*. Then we randomly select 100 tuples from the imported table to generate incomplete datasets. In DBLP-publisher table we set all the values of the *publisher* attribute *NULL* and in DBLP-isbn table we set all the values of the *isbn* attribute *NULL*.
2. **Fortune 1000 Contact Information(2008)** [18]. This dataset contains general contact information of the Fortune 1000 companies, including *Company Name, Main Office Address, Main Office City, Main Office State, Main Office Zipcode, Contact Phone Number, Website, General Contact Email Address, CEO Name, CEO E-mail Address*. Because the *General Contact Email Address* or *CEO Email Addresses* in some tuples are blank and their formats are same, in our training table we only keep the *General Contact Email Address*. Similarly, we randomly select 100 tuples to generate incomplete datasets. In Fortune-zipcode table, we set all the values of the *Main Office Zipcode* attribute *NULL* and in Fortune-email table we set all the values of the *General Contact Email Address* attribute *NULL*.
3. **BX-Books** [19]. BX-Books dataset is crawled from Book-Crossing Community by Cai-Nicolas Ziegler. It contains 1,149,780 piece of remarks on 271,379 books. Attributes in the dataset include *ISBN, Book Title, Book Author, Year Of Publication, Publisher, Image_URL_S, Image_URL_M, Image_URL_L*. We delete the last three URL attributes to generate our own training tables and test tables. Different from the DBLP dataset and the Fortune 1000 dataset, the quality of BX-Books are not perfect. For example, in our randomly selected 100 tuples, about 22 tuples' *Publisher* values are wrong and 1 tuple's *Year Of Publication* is wrong.

We impute these incomplete tables using our presented approaches and evaluate the *accuracy* of our solutions. Let N_c represent the number of the tuples which have been correctly imputed and N_t represent the number of tuples which we want to impute. Here $accuracy = \frac{N_c}{N_t} \times 100\%$. For the first two datasets, we use the original complete table as the ground truth. For the last dataset, in order to compensate for the poor quality, we manually retrieve the values of *publisher* and *Year of Publication* from some confidential websites, such as Amazon, to create the ground truth. In the following experiments, if the candidates are listed in the TOP 3 candidate lists we think they are found, otherwise the are not. We use Bing Search API in our experiments.

4.1 The Web-based Imputation Technique vs. Rule-based Techniques

Each tuple in our datasets is unique, so statistical approaches are incapable of accurately dealing with them. For a relational table, we can apply functional dependency rules as the deduction rules to derive imputation values. Consequently,

we compare the effectiveness of our web-based imputation techniques with rule-based techniques here. Related functional dependency rules found by TANE are shown in Table 3. For DBLP dataset, no functional dependency rules can be found for deriving the values of *isbn* and *publisher*. And for the *website* attribute of Fortune 1000 Contact Information dataset and the *isbn* attribute of BX-Books dataset, because there are not enough duplicate attribute values in them, we can impute few missing values. Consequently, rule-based techniques are also incapable of dealing with the incompleteness of our datasets. The effectiveness of our web-based techniques will be evaluated in the subsequent sections.

4.2 Evaluation of the Query Formulation Algorithms

In this section, we will evaluate the performance of the rule-based query formulation algorithm and the genetic-based query formulation algorithm. We mainly compare the *FITNESS* values of them. The *FITNESS* values on different attributes are shown in Table 3. On most attributes, the genetic-based query formulation algorithm outdoes the rule-based query formulation algorithm. The reason is that functional dependency rules can only reflect the relations among the attributes within a table, and incapable of reflecting the relations of the attributes in the WWW. It can not always get good retrieval results. Besides this, there are no functional dependency rules in several situations. For example on *DBLP* dataset, we cannot find any functional dependency rules which determine the values of *isbn* and *publisher*. So the usage of rule-based query formulation algorithm is limited. Genetic-based query formulation algorithm tries to find the relations of different values by training, so it can get better query results.

4.3 Evaluation on the Influence of Different Queries

In this section, we will evaluate the influence of different queries. $Q1$, $Q2$, $Q3$ and $Q4$ are the queries which contain all observed values of the same tuple. $Q9$, $Q10$, $Q11$ and $Q12$ are the queries generated by our generic-based query formulation algorithm. We also randomly select an unfixed number of observed attribute values to generate $Q5$, $Q6$, $Q7$ and $Q8$. From the results in Figure 3, we can see that the queries generated by our generic-based approach can help us impute more data and get more accurate imputation results on most datasets. $Q1$, $Q2$, $Q3$ and $Q4$ perform poorly because they contains too many attribute values. Too many values mean too many limitations.

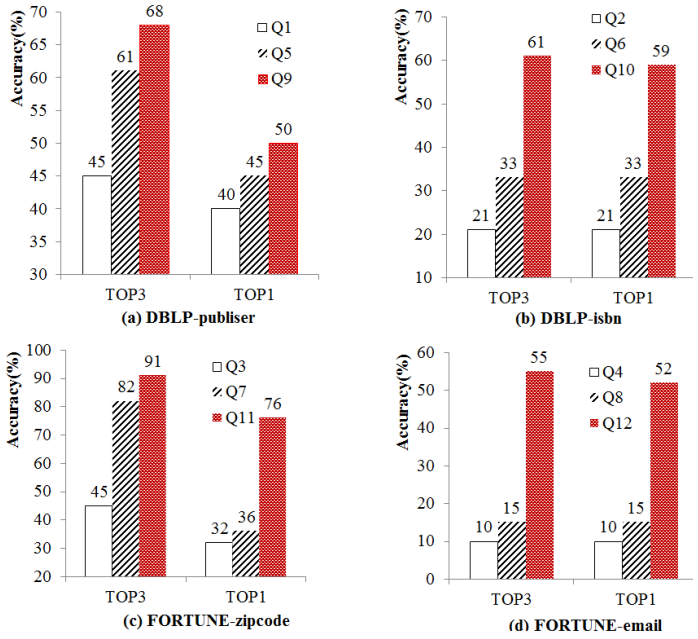


Fig. 3 Evaluation on the Influence of different queries

4.4 Evaluation of the Candidates Ranking Algorithms

In this section, we will evaluate the effectiveness of the weight-based ranking algorithm and the graph-based ranking algorithm. For the sake of fairness, in each pair of comparative experiments we use the same queries and the same entity extraction method. The results in Figure 4 show that the graph-based ranking algorithm outdoes the weight-based ranking method according to the number of the real values listed in TOP 3 and the number of the real values listed in TOP 1. The weight-based ranking method only tries to find the most frequently appearing candidates. As we know, the most frequently appearing ones may not be the correct candidates. The graph-based ranking algorithm tries to take advantage of the relations hidden in the WWW. It matches the real world better, so it works better.

4.5 Evaluation on the Influence of the Number of Snippets and Training Tuples

The amount of snippets retrieved and the amount of training tuples are two factors which will determine the possibilities of imputing missing attribute values.

From Figure 5, we can see that when we increase the amount of snippets retrieved by BING Search API, the imputation accuracy also increase. But more snippets mean more expense to use BING Search API and more compute expense to build the pattern graphs and target graphs. In our future work, we will try to build a model to determine how many

Table 3 Rule-based query formulation algorithm VS. genetic-based query formulation algorithm

Dataset	Target Attribute	Query Formulation Algorithm	Query	FITNESS
DBLP	isbn	FD		
		GA	"isbn"+author + year	0.668
	publisher	FD		
		GA	"publisher"+title + year	0.772
FORTUNE 1000	zipcode	FD	"zipecode"+address + state	0.946
		GA	"zipecode"+address + phone + website	0.924
	email	FD	"email"+website	0.027
		GA	"email"+address + website + ceoname	0.252
BX-BOOKS	year	FD	"year"+isbn	0.292
		GA	"year"+isbn + title + author	0.58
	publisher	FD	"publisher"+isbn	0.345
		GA	"publisher"+isbn + year	0.538

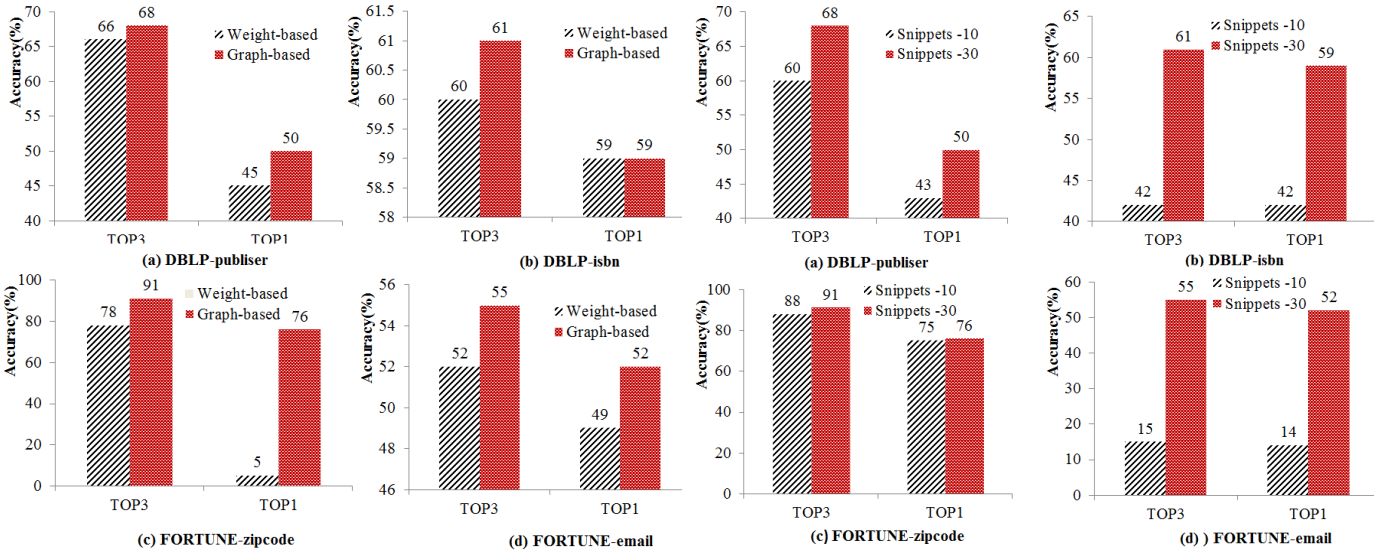


Fig. 4 Graph-based ranking algorithm VS. weight-based ranking algorithm

Fig. 5 Evaluation on the influence of the number of snippets

snippets we should get to reach the best balance between the expanse and imputation accuracy.

From Figure 6, we can see that when we increase the amount of training tuples which are used to form graph patterns, for some attributes imputation accuracy increases obviously, but for others no obvious change is observed. The reason is that the relation patterns can be determined by a proper amount of examples. After that, more examples can only strengthen the relation patterns but not change them. For different attributes, the proper amount may be different.

4.6 Accuracy Analysis of Our Web-based Data Imputation Platform

In this part, we evaluate the accuracy of our Web-based techniques on different kinds of incomplete values. Specifically, we select a textual attribute and a formatted numeric attribute in *DBLP* dataset, a normal numeric attribute and a formatted textual attribute in *FORTUNE 1000* dataset, a textual attribute and a short numeric attribute in *BX-BOOKs* dataset.

From Figure 7, we can see that our approach can impute about 60 percent of the missing values and about 50 percent can use the TOP1 candidates as the final value directly in the *DBLP* dataset and *FORTUNE 1000* dataset. The result shows that our approach is effective on all evaluated datasets.

Focusing on the accuracies on different kinds of attributes, we can find that our approaches perform differently on different attributes in Figure 7. Considering the TOP3 candidates, the imputation accuracy on *FORTUNE-zipcode* is much higher than the accuracies on other attributes, and the imputation accuracy on *FORTUNE-email* is the lowest. As is known to us, different kind of values are in different distribution in the WWW and the more widely the real candidates are distributed the more possibilities we can find them. We know that

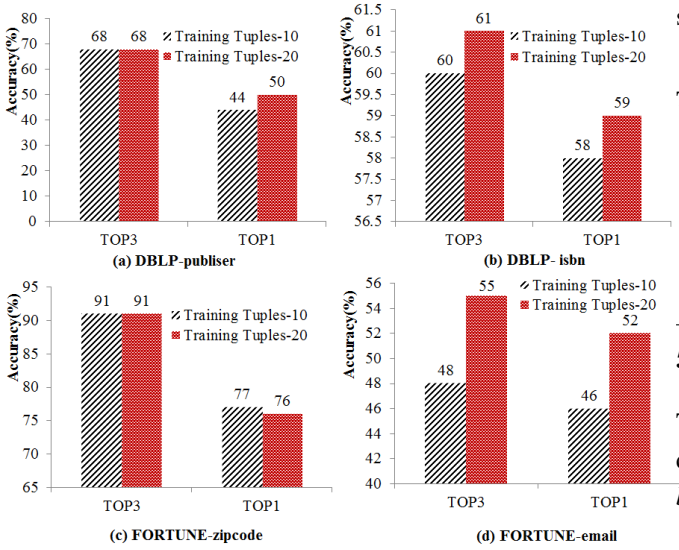


Fig. 6 Evaluation on the influence of the number of training tuples

there are a lot of Web-pages containing the *zipcode* of an address, but fewer Web-pages containing the general email of a specific company. Consequently, the imputation accuracy on *FORTUNE-zipcode* is much higher than the imputation accuracy on *FORTUNE-email*.

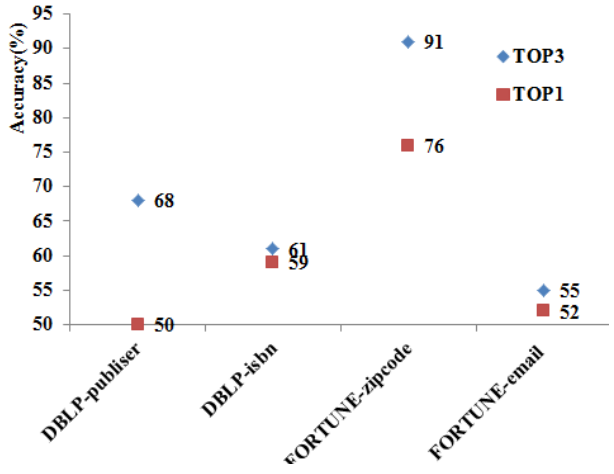


Fig. 7 Accuracy of our Web-based automatic data imputation techniques

We also evaluate our approach on the dataset *BX-BOOKS*, in which there are errors in some observed values. From Table 4, we can see that our approach can still retrieve good imputation candidates for the *publisher* attribute. But for the *Year of Publication* attribute, the imputation accuracy decreases to some extent. We conclude that there are two reasons. One is the influence of the errors in the *publisher* values, which will make us get unrelated snippets. The other is that there may be several related *year* values in the same snippet and they are much harder to be distinguished.

Table 4 Accuracy of BX-BOOKS imputation

Dataset	Target Attribute	TOP3	TOP1
BX-BOOKS	publisher	80%	54%
BX-BOOKS	Year of Publication	45%	36%

5 Related work

There exist three kinds of techniques to enhance information quality, including *self_info-based techniques* , *external_info-based techniques* and *human_based techniques* [20].

Self_info-based techniques just use intrinsic knowledge of the given database to improve the data quality of a database. Several statistical approaches have been introduced and compared in [1, 2, 4], like *mean imputation*, *selecting the most common attribute value as the missing value* , *assigning all possible values of the attribute restricted to the given concept*, *treating missing attribute values as special values* and so on. [21], [22] and [23] try to use data mining techniques, like machine learning, data classification, to find some approximate values of incomplete attributes. These methods purely rely on observed data to calculate or deduce missing values. So only when data is generally well organized and enough in the database, they may work well. Otherwise, they perform poorly, especially on the occasion that the amount of missing values exceeds a threshold. And what's more, their goal is to eliminate the influence of information missing but not to find real values. So these methods can not impute data accurately.

Human_based techniques improve the confidence of error detection and correction techniques by considering humanąs suggestions carefully. [24] and [25] try to utilize human's feedbacks to improve the confidence of data repairing operations. Human's feedbacks are the most effective and accurate way to repair data errors in a small specific area. But it is not efficient to deal with huge data and also incapable in unfamiliar areas. So, although these methods are effective, they can only be used in some specific areas.

External_info-based techniques try to use the information from external data sources, such as a knowledge base or the WWW. Using master data [7] to detect tuple missing errors in a database [26–28] is a typical work which introduce external information to improve data quality. However, in most enterprises master data only exists to record the most important information of the organization, such as the information of the assets, employees or projects. In other applications there

is always no master data. As the biggest information system in the world nowadays, the WWW is much more knowledgeable than its competitors, such as knowledge bases and master data databases in specific domains [20]. There are several Web-based works to complete a list or a relational table. Google Sets [29], SEAL [30, 31], NTT Set Expansion System [32] and WebSets [33] try to expand a list from several seeds with the help of the WWW. Their goal is to find the same kind of objects from related Web tables. Unfortunately, they are just effective on few regions till now, such as cars, fruits, et al. In other areas, they do not perform well. Because their goal is different from us, it can not be used in our paper. [34] and [35] try to integrate Web information into a knowledge bases. They are focused on information enriching but not quality enhancing. These techniques can not be used directly in our work. [36] proposes a Web-based technique to complete the relations between two datasets. It uses some known relations as seeds to learn the context pattern between two datasets and then use the context pattern to extract new relations between the target datasets. [14], [37] , [38] and [39] try to find real values of incomplete attributes of a relational table from the WWW. [37] proposes a web-based imputation method for improving the accuracy of Bayes classification networks. It tries to retrieve necessary numerical information from the WWW to compensate for the influence of incomplete values. [37] manually generates queries to retrieve documents and then utilize WHISK [40] to extract target information. Obviously, [37] is not an automatic method and can only be used to retrieve specific numeric data. In [14], pattern based data imputation(P-DI) and co-occurrence based data imputation strategies are proposed. P-DI and C-DI learn syntactic patterns and common context terms between two specific attributes respectively from some training tuples and then use them to retrieve missing values. In [14], the observed attributes which may help us retrieve the missing attributes are assumed known and its main work is to learn their relations. [38] and [39] try to improve the performance of [14]. These works are different from us for the reason that our input is just an incomplete table without any other knowledge. We have given a detail survey of these works in [41].

6 Conclusions

In this paper, we study on Web-based techniques for automatically imputing incomplete relational databases. To generate proper queries more efficiently, we propose two query formulation algorithms. One tires to utilize functional dependency rules directly and another tries to learn a proper pattern by genetic algorithms. To tackle the challenge on finding the most proper candidate imputing values, we propose a weight-based object extraction model and a graph-based object extraction model. Because the graph-based model takes full advantage of the semantic relations among different kinds of objects, it performs better. The detailed evaluations show that our approaches are effective and can dealing with the occasion that most tuples in a table are incomplete.

To cut down imputation cost, we will build some models to determine the proper amounts of the snippets and training tuples for each attribute in our future work. And further more, we will extend our approaches so as to deal with the situation that multiple incomplete attributes exist in one tuple.

7 Acknowledgments

The work was supported by the Ministry of Science and Technology of China, National Key Research and Development Program (No. 2016YFB1000700), the National Natural Science Foundation of China (No. 61502390, No. 61472321, No. 61402370, No. 61502392), and the Basic Research Fund of Northwestern Polytechnical University (No. 3102014JSJ0013, No. 3102014JSJ0005). Thanks a lot for the recommendation of ICYCSEE 2016.

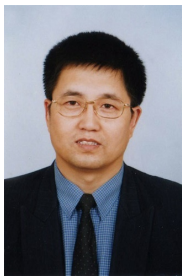
References

1. G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, 2003.
2. M. Magnani. Techniques for dealing with missing data in knowledge discovery tasks. *http://magnanim.web.cs.unibo.it/data/pdf/missingdata.pdf*, 2004.
3. M. Ramoni and P. Sebastiani. Robust learning with missing data. *Machine Learning*, 45(2):147–170, 2001.
4. J. W. Grzymala-Busse and M. Hu. *A Comparison of Several Approaches to Missing Attribute Values in Data Mining*, pages 378–385. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
5. X. F. Zhu, S. C. Zhang, Z. Jin, Z. L. Zhang, and Z. M. Xu. Missing value estimation for mixed-attribute data sets. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):110–121, 2011.
6. S. Ghosh. Statistical analysis with missing data. *Technometrics*, 30(4):455–455, 2012.
7. D. Loshin. *Master data management*. Morgan Kaufmann, 2010.
8. http://www.google.cn/intl/zh-CN/about/company/history/, 2008.
9. http://dblp.uni-trier.de/xml/, 2014.

10. N. Schlaefer, J. Ko, J. Betteridge, G. Sautter, M. Pathak, and E. Nyberg. Semantic extensions of the ephyra qa system for trec 2007. In *Proceedings of the Sixteenth Text REtrieval Conference*, volume 35, page 36, 2007.
11. Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.
12. J.H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. MIT press, 1992.
13. D. E Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.
14. Z. X. Li, M. A. Sharaf, L. Sitbon, S. Sadiq, M. Indulska, and X. F. Zhou. Webput: efficient web-based data imputation. In *13th International Conference on Web Information Systems Engineering*, pages 243–256. Springer, 2012.
15. D. Jurafsky and H. James. *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech*. Pearson Education, 2000.
16. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.
17. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.
18. http://www.cs.cmu.edu/ pavlo/datasets/fortune1000/.
19. http://grouplens.org/datasets/book-crossing/.
20. H.L. Liu, Z.H. Li, C.Q. Jin, and Q. Chen. Web-based techniques for automatically detecting and correcting information errors in a database. In *3rd International Conference on Big Data and Smart Computing*, pages 261–264, 2016.
21. K. Lakshminarayan, S. A. Harp, R. Goldman, and T. Samad. Imputation of missing data using machine learning techniques. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 140–145. AAAI Press, 1996.
22. Q.H. Wang and J. Rao. Empirical likelihood-based inference in linear models with missing data. *Scandinavian Journal of Statistics*, 29(3):563–576, 2002.
23. S.C. Zhang, J.L. Zhang, Z.F. Zhu, Y.S. Qin, and C.Q. Zhang. Missing value imputation based on data clustering. *Transactions on Computational Science*, pages 128–138, 2008.
24. M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *Proceedings of the VLDB Endowment*, 4(5):279–289, 2011.
25. Y.X. Tong, C. C Cao, C. J. Zhang, Y.T. Li, and L. Chen. Crowdcleaner: Data cleaning for multi-version data on the web via crowdsourcing. In *2014 IEEE 30th International Conference on Data Engineering (ICDE)*, pages 1182–1185. IEEE, 2014.
26. W. F. Fan and F. Geerts. Capturing missing tuples and missing values. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 169–178. ACM, 2010.
27. W. F. Fan and F. Geerts. Relative information completeness. *ACM Transactions on Database Systems (TODS)*, 35(4):27, 2010.
28. W. F. Fan, J. Z. Li, S. Ma, N. Tang, and W. Y. Yu. Towards certain fixes with editing rules and master data. *Proceedings of the VLDB Endowment*, 3(1-2):173–184, 2010.
29. J. Cirasella. Google sets, google suggest, and google search history: Three more tools for the reference librarian's bag of tricks. *The Reference Librarian*, 48(1):57–65, 2007.
30. R. C. Wang and W. W. Cohen. Language-independent set expansion of named entities using the web. In *Seventh IEEE International Conference on Data Mining*, pages 342–350. IEEE, 2007.
31. R. C. Wang and W. W. Cohen. Iterative set expansion of named entities using the web. In *Eighth IEEE International Conference on Data Mining*, pages 1091–1096. IEEE, 2008.
32. K. Sadamitsu, K. Saito, K. Imamura, and G. Kikui. Entity set expansion using topic information. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 726–731, 2011.
33. B. B. Dalvi, W. W. Cohen, and J. Callan. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 243–252. ACM, 2012.
34. H.Q. Bian, Y.G. Chen, X.Y. Du, and X.L. Zhang. Metkb: enriching rdf knowledge bases with web entity-attribute tables. In *22nd ACM International Conference on Information and Knowledge Managemen-t(CIKM)*, pages 2461–2464, 2013.
35. X.L. Zhang, Y.G. Chen, J.C. Chen, X.Y. Du, and L. Zou. Mapping entity-attribute web tables to web-scale knowledge bases. In *The 18th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 108–122, 2013.
36. Z.X. Li, M.A. Sharaf, L. Sitbon, X.Y. Du, and X.F. Zhou. Core: A context-aware relation extraction method for relation completion. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):836–849, 2014.
37. N. Tang and V. R. Vemuri. Web-based knowledge acquisition to impute missing values for classification. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 124–130. IEEE Computer Society, 2004.
38. Z.X. Li, M.A. Sharaf, L. Sitbon, S. Sadiq, M. Indulska, and X.F. Zhou. A web-based approach to data imputation. *World Wide Web*, 17(5):873–897, 2014.
39. Z.X. Li, S. Shang, Q. Xie, and X.L. Zhang. Cost reduction for web-based data imputation. In *19th International Conference on Database Systems for Advanced Applications*, pages 438–452, 2014.
40. S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.
41. H.L. Liu, Z.H. Li, Q. Chen, and Z.Q. Chen. A review on web-based techniques for automatically detecting and correcting information errors in relational databases. *Chinese Journal of Computers,*

http://www.cnki.net/kcms/detail/11.1826.TP.20161029.0118.008.html, 2016.

Hailong LIU is a lecturer at School of Computer Science, Northwestern Polytechnical University, Xi'an, China. He received his M.Sc. and Ph.D. degrees from Northwestern Polytechnical University. He is the member of China Computer Federation and ACM SIGMOD China. His research interests include data management and data quality.



Zhanhuai LI is a professor at School of Computer Science , Northwestern Polytechnical University, Xi'an, China. He is the vice-chairman of Database Technical Committee of China Computer Federation. He received his M.Sc. and Ph.D. degrees from Northwestern Polytechnical University. His research interests include data management and data quality.

Qun CHEN is currently a professor at School of Computer, Northwestern Polytechnical University in Xi'an, China. He received his Ph.D. degree from National University of Singapore. Between 2004 and 2006, he was a research associate in Hong Kong University of Science and Technology. His research interests include data management and data quality.

Zhaoqiang CHEN is a Ph.D. candidate at School of Computer, Northwestern Polytechnical University in Xi'an, China. His research interests include data management and data quality.