Cost Reduction for Web-Based Data Imputation

Zhixu Li[1], Shuo Shang[2], Qing Xie[1], and Xiangliang Zhang[1]

[1] King Abdullah University of Science and Technology, Saudi Arabia
[2] Department of Software Engineering, China University of Petroleum-Beijing, P.R. China
{zhixu.li,qing.xie,xiangliang.zhang}@kaust.edu.sa,
jedi.shang@gmail.com

Abstract. Web-based Data Imputation enables the completion of incomplete data sets by retrieving absent field values from the Web. In particular, complete fields can be used as keywords in imputation queries for absent fields. However, due to the ambiguity of these keywords and the data complexity on the Web, different queries may retrieve different answers to the same absent field value. To decide the most probable right answer to each absent filed value, existing method issues quite a few available imputation queries for each absent value, and then vote on deciding the most probable right answer. As a result, we have to issue a large number of imputation queries for filling all absent values in an incomplete data set, which brings a large overhead. In this paper, we work on reducing the cost of Web-based Data Imputation in two aspects: First, we propose a query execution scheme which can secure the most probable right answer to an absent field value by issuing as few imputation queries as possible. Second, we recognize and prune queries that probably will fail to return any answers a priori. Our extensive experimental evaluation shows that our proposed techniques substantially reduce the cost of Web-based Imputation without hurting its high imputation accuracy.

Keywords: Web-based Data Imputation, Imputation Query, Cost Reduction.

1 Introduction

Data incompleteness is a pervasive problem in all kinds of databases [19]. The process of filling in absent field/attribute values is well-known as *Data imputation* [3]. Recently, a new generation of Web-based Data Imputation approaches were proposed to retrieve the absent field values from the web [16, 9, 23, 4, 13, 20]. In particular, external data sources on the Web are typically rich enough to answer absent fields in a wide range of incomplete data sets, while complete fields in a data set can be utilized as keywords in imputation queries for absent fields in the same data set.

However, the *high cost* remains a big problem with the Web-based Data Imputation approaches. In particular, due to the ambiguity of existing information about absent values in the local data set, various answers might be retrieved for each absent value. In order to find out the most probable correct answer for an absent value, existing method has to trigger all available imputation queries to this value, and then decides the most probable correct answer according to the voting results proposed by all issued queries. In particular, existing method relies on a confidence scheme to estimate a confidence score for every issued query [15, 16] indicating the probability that the query

S.S. Bhowmick et al. (Eds.): DASFAA 2014, Part II, LNCS 8422, pp. 438–452, 2014.

Table 1. An Example Incomplete Table with Absent Values

Title	Name	Uni.	Street	City	Zip
Prof.	Jack Davis	NYU	W. 4th Str.	NYC	10012
Dr.	Tom Smith	NYU		NYC	10012
Mr.	Bill Wilson	Columbia	Broadway	NYC	10027
Mr.	Wei Wang	Cornell	East Ave.	Ithaca	14850
Ms.	Ama Jones		W. 37th Str.		90089
	Lank Hanks	UCLA	W. Boulevard	LA	90024
	Wei Wang	UIUC	W. Illinois Str.	Urbana	61801

can retrieve a right answer. Then it employs a confidence combination scheme, such as Noisy-All [17, 1], to decide a joint confidence score to each distinct retrieved answer. For example, for the absent `Uni` Value in the 5th tuple in Table 1, assume we have multiple queries with different existing attribute values as keywords, which are listed with their probabilistic scores and retrieved answers as follows:

Query (Keywords $\rightarrow$ Target Absent Value)	Confidence Score	Answer
"Ms. + Ama Jones + W. 37th Str."$\rightarrow$ `Uni` .	0.90	"USC"
"Ms. + Ama Jones + 90089" $\rightarrow$ `Uni` .	0.85	"UCLA"
"Ms. + Ama Jones + LA" $\rightarrow$ `Uni` .	0.80	"UCLA"
"Ms. + Ama Jones" $\rightarrow$ `Uni` .	0.65	"USC"

According to Noisy-All, the joint confidence score of "USC" is: $1-(1-0.90)*(1-0.65)=0.965$, while that of "UCLA" is: $1-(1-0.85)*(1-0.80)=0.97$. After all, the one with highest confidence score (i.e., "UCLA") will be taken as the most probable right answer, or what we call as the *winner answer*, to an absent value.

Given the above, to decide the winner answer to an absent value, existing method has to fire quite a few available imputation queries for each absent value in order for calculating the joint confidence of each distinct answer, which, on the other hand, brings a large overhead. In addition, among all fired queries, a significant part of them probably will not return any answer, which we call as *empty-result queries*. For example, for the absent `Uni` value in the fifth tuple in Table 1, the query "Ms. + Ama Jones + W. 37 th Str + 90089" $\rightarrow$ `Uni` may not return any results, as there is no Web documents mentioning all these values together with the `Uni` value. Apparently, issuing these empty-result queries without getting any results is totally a waste of resources.

In this paper, we work on reducing the number of issued queries by Web-based Data Imputation based on two intuitions below: First, an answer can be secured as the winner answer to an absent value as long as we can ensure that the joint confidence of this answer must be higher than that of all the other answers. Particularly, we can calculate a *lower-bound joint confidence* and *upper-bound joint confidence* to each distinct answer, indicating the lowest and highest joint confidence that this answer can achieve respectively, based on the results of already issued queries and the number and confidence of the left un-issued queries. Once we have the lower-bound joint confidence of an answer higher than the upper-bound joint confidence of all the other answers, we can secure this answer as the winner answer to the absent value. As a result, we don't need to issue the left un-issued queries. In addition, in order for securing the winner answer as early as possible (i.e., with as few issued queries as possible), queries with higher confidence

should be issued earlier than those with lower confidence, given that the cost of each query is more or less the same, but those with higher confidence have greater impact to the final imputation result to an absent value.

Secondly, a query w.r.t. an absent value in one tuple can be predicted as an empty-result query a priori, if it already failed to retrieve any answer to absent values in several other tuples, especially when these tuples share some common attribute values. The intuition here is, queries of the same format, that is, leveraging the same set of attributes (with complete values) as keywords and target at absent values under the same attribute, tend to have similar performance to absent values in similar instances. For example, after we learned that queries leveraging values under attributes `Title + Name + Zip` as keywords returned no answer to the absent `Street` value in several processed instances, it is very possible that query of the same format is also an empty-result query to an absent `Street` value in a new instance, when this new instance shares the same `Uni` values with the processed instances. Based on this intuition, we rely on a supervised learning method to construct an empty-result query prediction model for each query format. However, the drawback of supervised learning method lies on the requirement of a large training set. Fortunately, we can automatically generate training sets with complete instances of the data set.

In summary, the main contributions of this paper are as follows:

1) We propose to secure the winner answers to an absent value with as few issued queries as possible in Web-based Data Imputation through working out the condition for securing an answer as the winner answer to an absent value.
2) We devise empty-result queries predictors to predict empty-result queries a priori. Besides, an empty-result queries pre-pruning rule is also proposed for further identifying empty-result queries.
3) We perform experiments on several real-world data collections. The results demonstrate that the proposed two techniques could greatly decrease the number of executed queries by up to 85% comparing to existing method.

In the rest of this paper, we give preliminaries and then define our problems in Section 2. We present how we secure the winner answers early in Section 3, and then introduce the failure queries predictors in Section 4. The experiments are reported in Section 5, followed with related work in Section 6. We conclude in Section 7.

2 Preliminaries and Problems Definition

In this section, we give preliminaries on existing Web-based Data Imputation approaches, and then define our problems in this paper.

2.1 Preliminaries on Web-Based Data Imputation

Web-based Data Imputation formulates *Imputation Queries* to retrieve absent values from the Web [15, 16]. Specifically, for each absent value, imputation queries are issued to retrieve relevant web documents, from which we then extract the target absent value with Named Entity Extraction techniques [8, 5].

Definition 1. (Imputation Query). *Given an instance with an absent value under attribute Y and complete values under attributes $\mathbb{X}$, an imputation query to the absent*

Y value can leverage values under a set of attributes $\mathbf{X} \subseteq \mathbb{X}$ as keywords, denoted as $\mathbf{X} \rightarrow Y$, in order for retrieving an answer for the absent Y value from the Web.

Theoretically, any subset of existing attribute values in an instance can be used as keywords in an imputation query to any absent value in the same instance. In reality, however, only those queries with a relatively high *confidence* (which was set to ≥ 0.4 in [15, 16]) are qualified to be used for imputation.

In particular, the confidence of an imputation query format q: $(\mathbf{X} \rightarrow Y)$ indicates its probability in retrieving the right answer for an absent value under Y. The confidence of each query can be easily estimated based on a set of complete instances in the same data set [15]. For example, if queries of the same format $(\mathbf{X} \rightarrow Y)$ can retrieve correct answer for 8 complete instances out of 10, then its confidence can be calculated as 0.8. In reverse, the confidence of a retrieved answer equals to its corresponding imputation query, given the premise that all initially existing attribute values in the data set are correct.

However, due to the data complexity on the Web and the ambiguity of the used keywords in imputation queries, different queries may return different answers to the same absent value. In order to find out the most probable right answer to an absent value, the existing method has to trigger all imputation queries for this absent value, and then employs a confidence combination scheme to decide a probabilistic score to each distinct retrieved answer. Finally, only the answer with the highest *Joint Confidence* will be taken as the most probable right answer, or what we call the *Winner Answer*, to the absent value.

More specifically, assume an answer y was returned by a number of queries $Q(y)$ to an absent value under attribute Y, three different confidence combination schemes can be utilized as follows:

- *Max-Conf:* The highest confidence of a retrieved answer is taken as its joint confidence, that is,
$$C(y) = \max_{q \in Q(y)} C(q, y) \tag{1}$$
where $Q(y)$ is the set of all imputation queries that retrieve y as the answer to the absent value, and $C(q, y)$ is the confidence of y retrieved by q.
- *Sum-Conf:* The sum of the confidence of a retrieved answer is taken as its joint confidence, that is,
$$C(y) = \mathcal{N} * \sum_{q \in Q(y)} C(q, y) \tag{2}$$
Here $\mathcal{N}$ is a normaliser to prevent the confidence getting larger than 1. Usually, we set $\mathcal{N} = \frac{1}{\sum_{y' \in V(Y)} C(y')}$, where $V(Y)$ is the set of retrieved answers for this absent value.
- *Noisy-All:* We can also adopt the noisy-all scheme [17, 1] in estimating the joint confidence of a retrieved answer. That is,
$$C(y) = 1 - \prod_{q \in Q(y)} (1 - C(q, y)) \tag{3}$$

Example 1. Assume two distinct values y_1 and y_2 are retrieved for one absent value, where y_1 is retrieved by a query with confidence 0.8, and y_2 is retrieved by three other

queries with confidence 0.5, 0.4 and 0.4 respectively. The joint confidences of y_1 and y_2 according to different calculation schemes are listed in Table 2. After all, with the Max-Conf scheme, the winner answer is y_1, while with the other two schemes, the winner answer is y_2.

Table 2. Computing Joint Confidence for an Absent Value

Distinct Values with Retrieved Confidences	Max-Conf	Sum-Conf	Noisy-All
y_1: 0.8	0.80	0.38	0.80
y_2: 0.5; 0.4; 0.4	0.50	0.62	0.82
The Winner Answer	y_1	y_2	y_2

2.2 Problems Definition

In order to find the winner answer for an absent value, the existing method triggers all available imputation queries to the value, which, as a result, brings a large overhead. In addition, among all fired queries, a part of them probably will not return any answer, which we call as *empty-result queries*. In order to reduce the cost of Web-based Data Imputation, our work presented in this paper addresses and proposes solutions to the following two challenging problems:

1) How we secure the winner answer for an absent value by issuing as few queries as possible, when any one of the three confidence combination schemes above is applied? (Section 3)
2) How we decrease the number of issued empty-result queries? (Section 4)

3 Securing Winner Answers Early

This sections focus on securing the winner answer to an absent value with as few issued queries as possible. We first introduce how we secure winner answer with Max-Conf, and then present how we secure winner answer with Sum-Conf or Noisy-All.

3.1 Securing Winner Answers with Max-Conf: Confidence-Prior Execution

When the Max-Conf scheme is employed, we can secure the winner answer for an absent value as long as we find out the non-empty-result query with the highest confidence to this absent value. According to Max-Conf scheme, the answer retrieved by this particular query must be the winner answer to this absent value.

In this light, we can employ a simple *Confidence-Prior Execution Strategy*, which always selects the query with the highest confidence for execution at a time. Following this execution strategy, the first retrieved answer must be the winner answer to an absent value according to the Mac-Conf scheme, and as a result, we do not need to issue/execute the left un-issued queries. Given the above, we have:

Lemma 1. *With the confidence-prior execution strategy, the first retrieved answer to an absent value must be the winner answer to this absent value when the Max-Conf scheme is employed.*

Proof. Let q_1 be the first issued query with a returned answer y_1 for an absent value under Y. According to Max-Conf scheme, the joint confidence of y_1 must be: $C(y_1) = C(q_1, y_1) = C(q_1)$. For all the other queries, since their confidence is lower than $C(q_1)$, so they are impossible to return an answer with a confidence higher than $C(y_1)$. Thus, y_1 must be the winner answer to the absent value.

3.2 Securing Winner Answers with Sum-Conf and Noisy-All

Different from Max-Conf, when either Sum-Conf or Noisy-All is employed, the joint confidence of every answer to an absent value can only be calculated when all corresponding queries are executed. Nonetheless, we can secure an answer, say y_1, as the winner answer if no other answer can beat y_1 with a higher joint confidence w.r.t. a target absent value. This condition requires us to compute the *upper-bound joint confidence* and *lower-bound joint confidence* of every distinct answer w.r.t. an absent value. Specifically, the upper-bound joint confidence of an answer w.r.t. an absent value is the highest possible confidence that this answer can achieve after all imputation queries to this absent value are executed, while the lower-bound joint confidence of an answer w.r.t. an absent value is the lowest possible confidence that this answer can achieve after all imputation queries to this absent value are executed.

Based on the definitions above, we can have the following condition for securing the winner answer to an absent value without executing all corresponding queries:

Lemma 2. *Once the lower-bound joint confidence of an answer y_1 is higher than the upper-bound joint confidence of all the other answers w.r.t. an absent value, we can secure y_1 as the winner answer to this absent value a priori.*

Note that the upper-bound and lower-bound joint confidence of every answer w.r.t. an absent value will be updated every time a new query to this absent value is executed. Once the winner answer to an absent value is secured, we don't need to execute the left yet executed queries for this absent value.

In the following, we introduce how we calculate the upper-bound and lower-bound join confidence to an answer with either Sum-Conf or Noisy-All scheme.

1. Calculating Upper-bound and Lower-bound with Sum-Conf: Let Q_t denote the set of executed queries to an absent value under Y, and $Q_t(y) \subset Q_t$ denote the set of executed queries that retrieve the answer y. Let Q_u denote the set of yet executed queries to the absent value. The upper-bound joint confidence of answer y w.r.t. this absent value can be achieved when all the unexecuted queries in Q_u will retrieve y. Hence, we have the upper-bound joint confidence to y as:

$$\overline{C(y)} = \frac{\sum_{q \in Q_t(y)} C(q) + \sum_{q \in Q_u} C(q)}{\sum_{q \in Q_t} C(q) + \sum_{q \in Q_u} C(q)} \tag{4}$$

where $C(q)$ denotes the confidence of a query q.

On the contrary, the lower-bound joint confidence of answer y w.r.t. the absent value can be gotten when all the unexecuted queries in Q_u will not retrieve y for the absent value. Hence, we have the lower-bound joint confidence to y as:

$$\underline{C(y)} = \frac{\sum_{q \in Q_t(y)} C(q)}{\sum_{q \in Q_t} C(q) + \sum_{q \in Q_u} C(q)} \tag{5}$$

Let $C_t(y) = \sum_{q \in Q_t(y)} C(q)$ denote the *Temporary Confidence* that an answer y holds w.r.t. an absent value according to the Sum-Conf scheme. Then we have the condition for securing the winner answer as:

Lemma 3. *Assume answer y_1 holds the highest temporary confidence $C_t(y_1)$ w.r.t. an absent value at a moment, and answer y_2 holds the second highest temporary confidence $C_t(y_2)$, we say y_1 can be secured as the winner answer if $C_t(y_1) - C_t(y_2) > \sum_{q \in Q_u} C(q)$ when the Sum-Conf scheme is employed.*

Proof. Let $\overline{C(y_2)}$ denote the upper-bound join confidence of y_2, and $\underline{C(y_1)}$ denote the lower-bound join confidence of y_1, both of which will be achieved when all the unexecuted queries retrieve y_2 for the absent value. According to Lemma 2, as long as we have $\overline{C(y_2)} < \underline{C(y_1)}$, we can secure y_1 as the winner answer. Finally, we have: $C_t(y_1) - C_t(y_2) > \sum_{q \in Q_u} C(q)$ as the condition to secure y_1 as the winner answer.

2. Calculating Upper-bound and Lower-bound with Noisy-All: With the Noisy-All scheme, the upper-bound joint confidence to an answer y w.r.t. an absent value can also be achieved when all the unexecuted queries in Q_u will retrieve y for the absent value. That is,

$$\overline{C(y)} = 1 - \prod_{q \in Q_t(y)} (1 - C(q)) * \prod_{q \in Q_u} (1 - C(q)) \tag{6}$$

, while the lower-bound joint confidence to y is also gotten when all the unexecuted queries in Q_u will not retrieve y for the absent value. More specifically,

$$\underline{C(y)} = 1 - \prod_{q \in Q_t(y)} (1 - C(q)) \tag{7}$$

Let $C_t(y) = 1 - \prod_{q \in Q_t(y)} (1 - C(q))$ denote the temporary confidence that an answer y holds w.r.t. an absent value, we have the condition for securing the winner answer with the Noisy-All scheme as:

Lemma 4. *With the Noisy-All scheme, an answer y_1 can be secured as the winner answer to an absent value if $\frac{1 - C_t(y_1)}{1 - C_t(y_2)} < \prod_{q \in Q_u} (1 - C(q))$, where y_1 is the answer with the highest temporary confidence $C_t(y_1)$, while y_2 is the answer with the second highest temporary confidence $C_t(y_2)$.*

Proof. Let $\overline{C(y_2)}$ denote the upper-bound join confidence of y_2, and $\underline{C(y_1)}$ denote the lower-bound join confidence of y_1, both of which will be achieved when all the unexecuted queries retrieve y_2 for the absent value. According to Lemma 2, as long as

we have $\overline{C(y_2)} < \underline{C(y_1)}$, we can secure y_1 as the winner answer. Finally, we have: $\frac{1-C_t(y_1)}{1-C_t(y_2)} < \prod_{q \in Q_u} (1 - C(q))$ as the condition to secure y_1 as the winner answer.

Note that Lemma 3 and Lemma 4 should work in conjunction with the confidence-prior execution strategy, such that the winner answer to an absent value can be identified as early as possible.

4 Pre-pruning Empty-Result Queries

The confidence-prior execution strategy proposed in Section 3.1 tends to give priority to queries with high confidence. However, queries with high confidence also have high risks to become *empty-result queries*, as these queries usually involve many attribute values which might not appear together with the absent value in one web page. As a result, the existing method may have to bear the cost of executing a large number of empty-result queries, while garnering non of the benefits.

In this section, we propose to reduce the number of issued empty-result queries by identifying them a priori with two orthogonal techniques below: First, we build an *Empty-Result Queries Predictor* for each query format in Section 4.1, which is actually a classification model that can classify queries into empty-result category and effective category. If the prediction result is positive (i.e., an effective query), we will execute this query, otherwise, we skip the query. Second, we introduce a heuristic *Empty-Result Queries Pre-pruning Rule* developed based on the relationship between queries in Section 4.2.

4.1 Empty-Result Queries Predictor

In this subsection, we introduce how we construct empty-result query predictors. Intuitively, queries targeting at the same target attribute Y by leveraging the same set of leveraged attribute $\mathbf{X}$ tend to have similar performance when applying them to similar instances.

Based on this intuition, we rely on a supervised model to predict empty-result queries a priori. More specifically, for all queries of the same format: q: $(\mathbf{X} \rightarrow Y)$, one supervised empty-result query predictor will be formulated, which will predict the imputation result of the query to an instance as either (1) *Negative*, which means the query is an empty-result query to the instance; or (2) *Positive*, which means the query is an effective query to the instance.

In the following, before presenting how we perform supervised learning to a query predictor, we first introduce the features that designed for each instance, and describe how we generate the training set with complete instances in the data set.

1. Features: Basically, we utilize two groups of features for each instance based on the intuitions (or observations) below: (1) instances having complete values under the same set of attributes are more likely to be in the same category. (2) instances sharing some complete attribute values are more likely to be in the same category.

Firstly, instances having complete values under the same set of attributes are covered by the same set of queries, which is the basic information required by the query predictor. Therefore, the first group of features are defined on whether an instance t has an existing value under each attribute A. If $t[A]$ is not absent, then the feature value corresponding to this attribute is "TRUE", otherwise "FALSE". This group of features provide the basic information of each instance.

Secondly, sharing some attribute values means instances having some common properties in some aspects, which may have more or less impact on whether a query can retrieve any answer for an instance. For example, if two instances in Table 1 share the same `University` value, which means the two people are from the same university, then there is perhaps some unified format for home pages of all staffs in this university, and as a result, their home pages may always mention the street address and Zip code of the university. However, some other shared attribute values may be less important such as `Title` values. On the other hand, taking every attribute value as a feature will let the feature space be very large, and the feature space be very sparse.

In order to find out "useful" shared attribute values (or their combinations) for constructing the second group of features, we propose to learn some special kind of "association rules" holding between attribute values under $\mathbf{X}$, and the retrieving result (Positive or Negative) of a query format to an instance. For example, if we learn that when `City`="NYC", queries of the format $\{$ `Name`, `Uni`, `City`$\} \rightarrow$ `Zip` are not empty-result queries to 95% of training instances, then "NYC" will be taken as a "useful" shared attribute value to the predictor. Here we can set a confidence threshold (such as 75% in our experiments which is demonstrated as a good one) to divide "useful" attribute values (or combinations of them) from the others. That is, when an attribute value or a combination of attribute values appears in an "association rule" like the example one above, it will be taken as an "useful" attribute values, and the confidence of the association rule will be taken as the weight of the value (or the combination of values) in training predictors. Finally, each detected "useful" attribute value or combination of values will be taken as a feature. For each instance, if it contains an "useful" value or combination of values, it will have a "TRUE" value under corresponding feature, otherwise, "FALSE".

2. Training set: Now we introduce how we automatically generate the training set for a query format q: $\mathbf{X} \rightarrow Y$ based on a set of complete instances. For each complete instance t, we take its attribute values under $\mathbf{X}$ attributes, denoted as $t[\mathbf{X}]$, as keywords to formulate a web-based imputation query q like $(\mathbf{X} = t[\mathbf{X}]) \rightarrow (Y = ?)$. We execute this query to get retrieved answers $Y_{t[\mathbf{X}]}$. If $Y_{t[\mathbf{X}]} = \emptyset$, this instance t will be labeled as "Negative". Otherwise, "Positive".

Note that when the table is large, we probably have a large number of complete instances to a query format. To save the cost, we only sample a small number of instances for training. However, simple sampling methods are not sufficient, given that the unbalanced distributions of $\mathbf{X}$ values in the data set. Let instances sharing the same $\mathbf{X}$ attribute values as one group. There might be several very large groups, and many small groups. As a result, either uniform row sampling or uniform group sampling will generate a biased training set. Here we adopt a *Two-Pass Sampling Scheme* [6] for estimating the confidence of each query pattern in a local table. In the first pass, we uniformly sample a

number of rows and put into a sample set $\mathcal{S}$. In the second pass, we put instances sharing the same $\mathbf{X}$ attribute values in $\mathcal{S}$ into one group, and then uniformly sample some rows from each group. With this two-pass sampling, the contributions of large groups are not lost in the first pass, and enough information is collected about each of the sampled small groups to correctly compute their contributions to the overall training set [6].

3. Predictor Learning: We start with a small set of training data set, and then update query predictors with new completed instances. In particular, each time we execute a query to an instance, we will see if this query is a positive or negative one to this instance. This information can then be used to update the corresponding predictor according to an on-line learning algorithm.

Here we adopt a popular on-line learning algorithm, *winnow* [18], which is simple but effective when it is applied to many on-line learning problems. In our problem, all features are boolean-valued features, that is, $f_i(t) \in \{1, 0\}$ $(1 \leq i \leq n)$, and the label of each instance is also boolean-valued, that is, $label_p(t) = \{\text{True, False}\}$. The algorithm maintains non-negative weights w_i for the i-th feature. Initially, the first group of features have uniformed weights, and the second group of features have their calculated weights respectively. For each instance t with a feature vector $[f_1(t), f_2(t), ..., f_n(t)]$, the algorithm uses a linear classifier for prediction as:

$$label_p(t) = \begin{cases} True & (\sum_{i=1}^{n} w_i f_i(t) > \delta) \\ \\ False & (otherwise) \end{cases} \tag{8}$$

where δ is a threshold that defines a dividing hyperplane in the instance space. Good bounds are obtained if $\delta = n/2$.

Then, for each instance that the classifier has predicted, after executing the query corresponding the query pattern to this instance and get the true label $label(t)$ for it, we apply the following update rule to the classifier:

- If the instance is correctly classified, that is, $label(t) = label_p(t)$, we do nothing to the classifier.
- If the instance is incorrectly classified, and $label(t) = True$, but $label_p(t) = False$, all of the weights of the features with value 1 in the mistake instance are multiplied by a value α (promotion step).
- If the instance is incorrectly classified, and $label(t) = False$, but $label_p(t) = True$, all of the weights of the features with value 1 in the mistake instance are divided by a value α (demotion step).

We set a typical value 2 to α. Initially, we use the generated training data set to build the predictor for each query pattern. Later, in the imputation process, each time we executed an imputation query which returns empty result, we will update its corresponding predictor according to the rules above.

4.2 Query Pre-Pruning Rule

Besides the empty-result queries predictors, an empty-result query pre-pruning rule can also be developed based on the relationship between queries to the same absent value as follows:

Lemma 1. *For one absent value, if a query q_1: $(\mathbf{X}_1 \rightarrow Y)$ fails to retrieve any value for an instance, then another query q_2: $(\mathbf{X}_2 \rightarrow Y)$, where $\mathbf{X}_1 \subseteq \mathbf{X}_2$, will also be an empty-result query to this instance.*

Proof. We assume q_2 can also retrieve an answer, say y', for the absent value in an instance, there should be some web pages containing both y' and values under $\mathbf{X}_2$ in this instance. Since $\mathbf{X}_1 \subseteq \mathbf{X}_2$, in no doubt, these web pages should also contain values under $\mathbf{X}_1$ in this instance, thus q_1 should also return y' as an answer to the absent value in this instance. However, since we know that q_1 fails to retrieve any answer to this absent value, thus the assumption that q_2 can retrieve an answer for the absent value in this instance is incorrect.

According to Lemma 1, each time a query q fails to retrieve any answer for an instance, we could prune some other queries for the same instance, whose leveraged attribute set is a superset of q's leveraged attribute set.

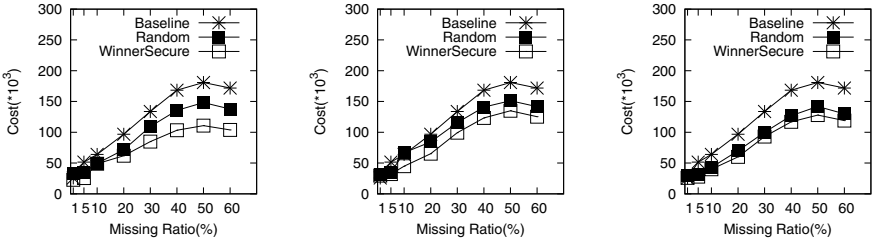
5 Experiments

We have implemented a Web-based Data Imputation prototype in Java which uses Google API to answer data imputation queries [15, 16]. To demonstrate the effectiveness of our techniques, we have experimented with two large real-world data sets and one small but interesting real-world data set. We say the 3rd data set is interesting since it is a multilingual data set which could evaluate the performance of our techniques in multilingual scenario:

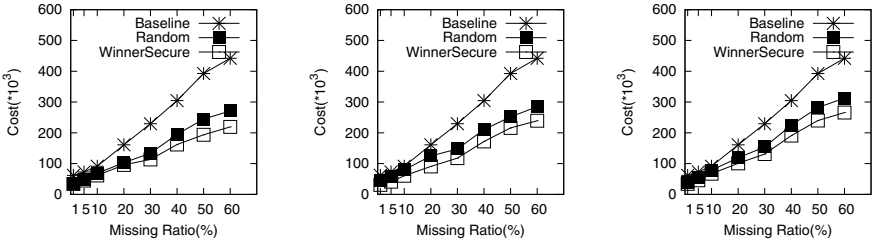
- *Personal Information Table (PersonInfo):* This is a 50k-tuples, 9-attributes table, which contains contact information for academics including name, email, title, university, street, city, state, country and zip code. This information is collected from more than 1000 different universities in the USA, UK, Canada and Australia.
- *DBLP Publication Table (DBLP):* This is a 100k-tuples, 5-attributes table. Each tuple contains information about a published paper, including its title, first author, conference name, year and venue. All papers are randomly selected from DBLP.
- *Multilingual Disney Cartoon Table (Disney):* This table contains names of 51 classical disney cartoons in 8 different languages collected from Wikipedia.

All the three data sets are complete relational tables. To generate incomplete tables for our experiments, we remove attribute values at random positions from the complete table, while making sure that at least one key attribute value will be kept in each tuple. For the PersonInfo dataset, the name and email are key attributes. For the DBLP dataset, the paper title is the only key attribute. For the Disney dataset, all attributes are key attributes.

Each reported result is the average of 5 evaluations, that is, for each absent value percentage (1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%), 5 incomplete tables will be generated with 5 random seeds, and the experimental results we present are the average results based on the 5 generated incomplete tables. We then impute these generated incomplete tables using our methods and evaluate the performance of our solutions by using the original complete table as the ground truth.



(a) PersonInfo (Max-Conf) (b) PersonInfo (Sum-Conf) (c) PersonInfo (Noisy-All)



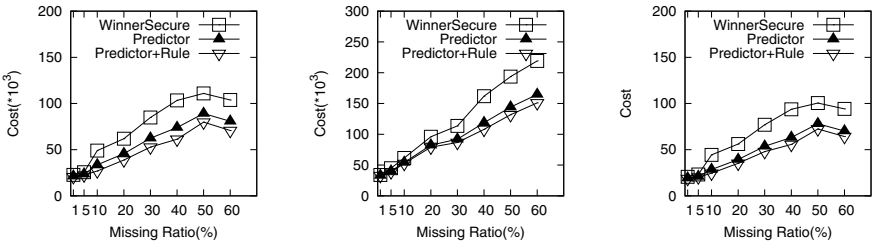
(d) DBLP (Max-Conf) (e) DBLP (Sum-Conf) (f) DBLP (Noisy-All)

Fig. 1. The Effectiveness of Winner Answer Securing Techniques

5.1 Evaluating Winner Answers Securing Techniques

To evaluate the effectiveness of the winner answer securing techniques, we compare the imputation cost of the following three methods based on the PersonInfo and DBLP data sets: (1) a *Baseline* method which issues all available queries for each absent value; (2) the *Random* method which randomly decides the next query to execute, and also applies the condition for securing winner answers; and (3) the *WinnerSecure* method, which uses the conditions for securing winner answers together with the confidence-prior execution strategy.

As demonstrated in Figure 1, with the winner answer securing condition only, the Random method can decrease around 30%-40% issued queries. In addition, with the confidence-prior execution strategy, the WinnerSecure method can further decrease about 15% issued queries when the Max-Conf scheme is applied, and can decrease



(a) PersonInfo (b) DBLP (c) Disney

Fig. 2. The Effectiveness of Empty-Answer Queries Pre-Pruning Techniques

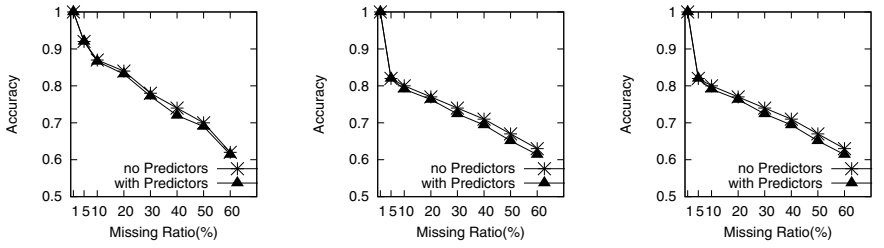
about 5-10% issued queries when either one of the other two confidence combination schemes is applied.

5.2 Evaluating Empty-Result Queries Pre-pruning Techniques

To evaluate the effectiveness of the empty-result queries pre-pruning techniques, we compare the cost of the following three methods on all the three data sets with MaxConf scheme: (1) *WinnerSecure* method; (2) *Predictor* method which uses the empty-result queries predictor based on the *WinnerSecure* method; (3) *Predictor+Rule* method which uses both empty-result queries predictor and query pruning rule based on the *WinnerSecure* method. As described in Figure 2, by using the empty-result queries predictor, the *Predictor* method can decrease about 20% cost of the *WinnerSecure* method. In addition, the *Predictor+Rule* method can further decrease about 5-10% cost of the *Predictor* method.

Overall, the combination of *WinnerSecure* and *Predictor+Rule* can decrease the number of issued queries by up to 85% of the *Baseline* method. As a result, imputing each absent value averagely costs only 1.2-1.5 imputation queries.

Finally, we evaluate whether the empty-result queries predictors will decrease the imputation accuracy by killing some positive queries. As depicted in Figure 3, by using the predictors, the imputation accuracy will be slightly decreased by no more than 5%. Apparently, compared with the improvement of the imputation efficiency, it is cost-efficient to use the predictors in doing web-based imputation.



(a) PersonInfo (b) DBLP (c) Disney

Fig. 3. The Side-Effect of Imputation Results Predictors

6 Related Work

The process of filling in absent attribute values is well-known as Data Imputation [3]. While most of the previous efforts focus on recovering absent attribute values from the complete part of the data set [11, 24, 26, 25, 2, 21], which we call as table-based data imputation methods, there are also many recent work conducted on retrieving absent attribute values from external resources, such as the Web [15, 14, 7, 23, 13, 16], i.e., web-based data imputation methods.

Existing table-based data imputation methods can be divided into two categories: (1) substitute-based data imputation; and (2) model-based data imputation. More specifically, the substitute-based data imputation approaches find a substitute value for the absent one from the same data set [11, 24, 25], such as selecting the most common

attribute value [11] or the k-Nearest Neighbor [10, 25] as the absent value. Besides, association rules between attribute values are also learned and used to find a close-fit value for an absent one from a similar context [22]. However, they are unlikely to reach a high imputation precision and recall, since the right absent value can only be inferred when the value is in the complete part of the data set. Differently, the model-based data imputation approaches [2, 21, 26] aim at building a prediction model based on the complete part of the data set for estimating an answer to each absent value. However, close estimations can not replace the absent original values, especially for non-quantitive string attribute values like email addresses.

Web-based data imputation methods resort to the World Wide Web for answering absent values. Many work has been conducted to harvest absent values from either web lists [13, 7], or web tables [14, 12, 23, 20]. More recently, a general Web-based approach, WebPut, was proposed to retrieve missing data from all kinds of web documents [15, 16]. In principle, WebPut utilizes the available information in an incomplete database in conjunction with the data consistency principle. It extends popular Information Extraction (IE) methods for the purpose of formulating data imputation queries that are capable of effectively retrieving absent values with high accuracy. However, the cost of web-based retrieving is much higher than that of the table-based inference, as all absent values need to be retrieved from the web. Although some schedule algorithms were proposed in WebPut, they are insufficient to solve the high cost problem. This paper is an extended work on WebPut, which aims at using much fewer queries to reach the same high imputation accuracy.

7 Conclusions and Future Work

In this paper, we mainly focus on minimizing the number of issued imputation queries in doing Web-based Data Imputation. We not only propose to secure the winner answers by executing as few imputation queries as possible in Web-based Data Imputation through working out the conditions for securing an answer as the winner answer to an absent value, but also devise empty-result queries predictors to predict empty-result queries a priori. The experiments based on several real-world data collections demonstrate that we can greatly decrease the number of issued queries by up to 85% comparing to the baseline.

An underlying assumption of the work is that all existing attribute values are faultless (i.e., clean data), meanwhile we leave the problem of data imputation in the presence of incorrect and dirty data as part of our future work. We will also work on a hybrid approach that combines and integrates our web-based approach with previous model-based data imputation methods.

References

[1] Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: ACM DL, pp. 85–94 (2000)
[2] Barnard, J., Rubin, D.: Small-sample degrees of freedom with multiple imputation. Biometrika 86(4), 948–955 (1999)
[3] Batista, G., Monard, M.: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5-6), 519–533 (2003)
[4] Chaudhuri, S.: What next?: A half-dozen data management research goals for big data and the cloud. In: PODS, pp. 1–4 (2012)

[5] Cohen, W., Sarawagi, S.: Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In: ACM SIGKDD, pp. 89–98 (2004)
[6] Cormode, G., Golab, L., Flip, K., McGregor, A., Srivastava, D., Zhang, X.: Estimating the confidence of conditional functional dependencies. In: SIGMOD, pp. 469–482 (2009)
[7] Elmeleegy, H., Madhavan, J., Halevy, A.: Harvesting relational tables from lists on the web. PVLDB 2(1), 1078–1089 (2009)
[8] Finkel, J., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: ACL, pp. 363–370 (2005)
[9] Gomadam, K., Yeh, P.Z., Verma, K.: Data enrichment using data sources on the web. In: 2012 AAAI Spring Symposium Series (2012)
[10] Grzymala-Busse, J., Grzymala-Busse, W., Goodwin, L.: Coping with missing attribute values based on closest fit in preterm birth data: A rough set approach. Computational Intelligence 17(3), 425–434 (2001)
[11] Grzymała-Busse, J.W., Hu, M.: A comparison of several approaches to missing attribute values in data mining. In: Ziarko, W.P., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 378–385. Springer, Heidelberg (2001)
[12] Gummadi, R., Khulbe, A., Kalavagattu, A., Salvi, S., Kambhampati, S.: Smartint: Using mined attribute dependencies to integrate fragmented web databases. Journal of Intelligent Information Systems, 1–25 (2012)
[13] Gupta, R., Sarawagi, S.: Answering table augmentation queries from unstructured lists on the web. PVLDB 2(1), 289–300 (2009)
[14] Koutrika, G.: Entity Reconstruction: Putting the pieces of the puzzle back together. HP Labs, Palo Alto, USA (2012)
[15] Li, Z., Sharaf, M.A., Sitbon, L., Sadiq, S., Indulska, M., Zhou, X.: Webput: Efficient web-based data imputation. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 243–256. Springer, Heidelberg (2012)
[16] Li, Z., Sharaf, M.A., Sitbon, L., Sadiq, S., Indulska, M., Zhou, X.: A web-based approach to data imputation. In: WWW (2013)
[17] Lin, W., Yangarber, R., Grishman, R.: Bootstrapped learning of semantic classes from positive and negative examples. In: ICML Workshop on The Continuum from Labeled to Unlabeled Data, vol. 1, p. 21 (2003)
[18] Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2(4), 285–318 (1988)
[19] Loshin, D.: The Data Quality Business Case: Projecting Return on Investment. Informatica (2006)
[20] Pantel, P., Crestan, E., Borkovsky, A., Popescu, A., Vyas, V.: Web-scale distributional similarity and entity set expansion. In: EMNLP, pp. 938–947 (2009)
[21] Quinlan, J.: C4. 5: Programs for machine learning. Morgan kaufmann (1993)
[22] Wu, C., Wun, C., Chou, H.: Using association rules for completing missing data. In: HIS, pp. 236–241 (2004)
[23] Yakout, M., Ganjam, K., Chakrabarti, K., Chaudhuri, S.: Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In: SIGMOD, pp. 97–108. ACM (2012)
[24] Zhang, S.: Shell-neighbor method and its application in missing data imputation. Applied Intelligence 35(1), 123–133 (2011)
[25] Zhang, S.: Nearest neighbor selection for iteratively knn imputation. Journal of Systems and Software 85(11), 2541–2552 (2012)
[26] Zhu, X., Zhang, S., Jin, Z., Zhang, Z., Xu, Z.: Missing value estimation for mixed-attribute data sets. IEEE TKDE 23(1), 110–121 (2011)