WebPut: Efficient Web-based Data Imputation

Zhixu Li[1], Mohamed A. Sharaf[1], Laurianne Sitbon[2],
Shazia Sadiq[1], Marta Indulska[1], Xiaofang Zhou[1]

[1] The University of Queensland, QLD 4072 Australia
[2] Queensland University of Technology, QLD 4000 Australia
{zhixuli, shazia, zxf}@itee.uq.edu.au, m.sharaf@uq.edu.au,
laurianne.sitbon@qut.edu.au, m.indulska@business.uq.edu.au

Abstract. In this paper, we present WebPut, a prototype system that adopts a novel web-based approach to the data imputation problem. Towards this, Webput utilizes the available information in an incomplete database in conjunction with the data consistency principle. Moreover, WebPut extends effective Information Extraction (IE) methods for the purpose of formulating web search queries that are capable of effectively retrieving missing values with high accuracy. WebPut employs a confidence-based scheme that efficiently leverages our suite of data imputation queries to automatically select the most effective imputation query for each missing value. A greedy iterative algorithm is also proposed to schedule the imputation order of the different missing values in a database, and in turn the issuing of their corresponding imputation queries, for improving the accuracy and efficiency of WebPut. Experiments based on several real-world data collections demonstrate that WebPut outperforms existing approaches.

Keywords: Web-based Data Imputation, WebPut, Incomplete Data

1 Introduction

Data incompleteness is one of the most pervasive data quality problems especially for web databases [13]. The process of filling in missing attribute values is well-known as *Data Imputation* [5, 16]. Commonly used data imputation approaches assign a missing value to "the most common attribute value", "a special value" [8], or a "closest-fit" value from the most similar context within the data set [6, 11]. Another line of data imputation approaches attempts to predict an estimation for the missing values using models built on the incomplete data set [15, 18, 21]. Such approaches could effectively smooth the influence of missing attribute values on some statistical data analysis. Both approaches, however, typically fall short in replacing the individual missing attribute values, especially when those values are unique within the data set (e.g., the missing `Email` addresses in Table 1), which is precisely the context we address in this paper.

The premise underlying our work is that most of the missing data in a wide range of online databases is typically available from some external data sources on the world wide web. The effective and efficient retrieval and extraction of this data, however, remains a challenging task, which motivated us to propose our novel *Web-based data imputation (WebPut)* approach.

Table 1. A Personal Information Table with Missing Data

	Name(N)	Email(E)	Title(T)	University(U)	State(S)
1	Jack Davis	jdavis@mit.edu	Professor	MIT	MA
2	Tom Smith	tomsmith2@cs.cmu.edu		CMU	PA
3	Bill Wilson		Doctor	UIUC	IL
4	Bob Brown	bbrown7@yale.edu	A/Professor	Yale	NY
5	Ama Jones		Ms		CA
6		lank@ucla.edu			

In principle, WebPut could be perceived as a novel Information Extraction (IE) approach. In particular, classical IE tasks typically process web documents for the purpose of recognizing a category of entities (such as *locations*), or relations (such as *company-headquarter pairs*) [1, 4, 19, 17]. WebPut, however, formulates the extraction tasks around the missing attribute values in a database, which greatly challenges the existing IE techniques. Towards this, WebPut utilizes the available information in the incomplete data set in conjunction with the data consistency principle. That is, the values in the same tuple are of the same instance, while the values in the same column are of the same domain. To this end, WebPut extends effective IE methods for the purpose of formulating web search queries that are capable of effectively retrieving the missing values in a database. The main contributions of this paper are summarized as follows:

1. We implement a WebPut prototype system that employs and extends a suite of traditional IE methods for the purpose of formulating effective web-based data imputation queries.
2. We propose a confidence-based scheme that efficiently leverages our suite of data imputation queries so that to automatically select the most effective imputation query for each missing value.
3. We propose a novel greedy iterative algorithm to schedule the imputation order of the different missing values in a database, and in turn the issuing of their corresponding imputation queries, for improving the accuracy and efficiency of WebPut.

Our experimental results on several real-world data collections show that WebPut outperforms all previous approaches and can achieve up to 80% accuracy. In addition, our proposed query scheduling techniques improve the efficiency of WebPut by up to 50%, as compared to the baseline approach.

Roadmap: We discuss both our system model and problem definition in Sec. 2, then present our proposed WebPut approach in Sec. 3. We report on our experimental results in Sec. 4, and then cover related work in Sec. 5. Then we conclude this paper in Sec. 6.

2 Model and Problem Definition

Our Web-based approach for data imputation (*WebPut*) leverages traditional *Information Extraction* (IE) methods together with the capabilities of Web search engines towards the goal of completing missing attribute values in relational tables such as the one shown in Table 1. Towards this, WebPut introduces what we call a *Web-based data imputation query*, which is basically a web search query specially formulated for the purpose of data imputation. Such data imputation query is formally defined as follows:

Definition 1. *For a relational tuple t, a* **data imputation query** $q(X \rightarrow y)$ *is a web search query formulated to utilize the existing values of a certain set of attributes* $\{X\} = \{x_1, x_2, ...\}$ *to retrieve the missing value of a certain attribute y. Hence, in* $q(X \rightarrow y)$*, X is denoted as the* **utilized attributes** *and y is denoted as the* **target attribute**.

In addition to formulating effective data imputation queries, the distribution of complete and missing values throughout the relational table makes the data imputation problem further challenging. Specifically, in a typical incomplete table, each tuple t might contain multiple target attributes: $y_1, y_2, ...$ and for each one of those target attributes there might exist multiple sets of possible utilized attributes: $X_1, X_2, ...$. For instance, in the 5th tuple of Table 1, there are three existing attribute values (Name, Title, State) and two missing values (Email, University). Any combination of the three existing attributes $\{N\}$, $\{T\}$, $\{S\}$, $\{N, T\}$, $\{N, S\}$, $\{T, S\}$ or $\{N, T, S\}$ could be utilized in a data imputation query targeting one of the two missing values.

Problem Definition: Given the above, our work presented in this paper addresses and proposes solutions to the following critical questions:

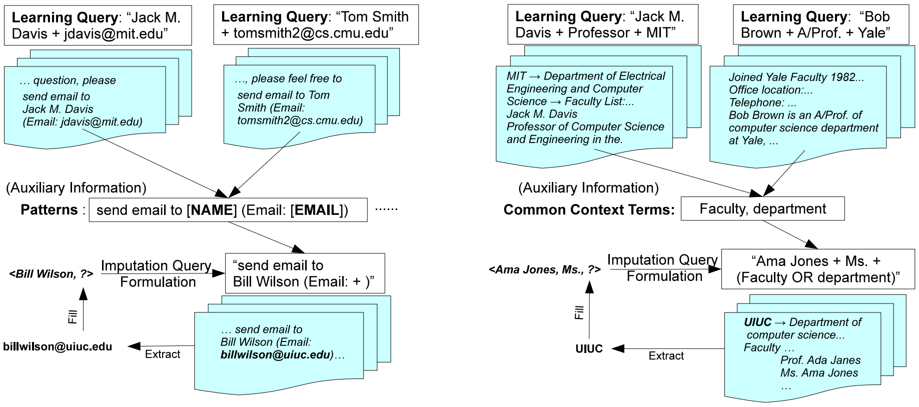
1. Given a pair $< X, y >$, how to **formulate** $q(X \rightarrow y)$ to effectively retrieve the missing value y? (Sec. 3.1)
2. In the presence of multiple sets of possible utilized attributes: $X_1, X_2, ...$ for the same target attribute y, how to **select** which set X is the best to impute y? (Sec. 3.2), and
3. In the presence of multiple target attributes: $y_1, y_2, ...$, how to **schedule** the imputation order of each y_j? (Sec. 3.3)

3 The WebPut Approach

In this section, we present WebPut, our web-based approach for data imputation. WebPut integrates several novel methods that provide effective and efficient solutions to the challenges listed above.

3.1 Web-based Data Imputation

WebPut extends popular and effective IE methods for the purpose of formulating effective data imputation queries. In particular, WebPut leverages existing complete tuples together with IE methods to *learn* the different query formulations suitable for the imputation of each missing value in a relational table. Central to that idea is searching for Web documents that contain some of the data in those complete tuples and extracting some *auxiliary information* from those documents to use in future data imputation queries. Given that approach, our previous definition of data imputation query could be further refined as: $q(X, A \rightarrow y)$ where A is the auxiliary information required for query formulation. The particular nature of this auxiliary information depends on the adopted IE method. While our approach described above is general enough to accommodate and extend any relevant IE method towards our goal of Web-based data imputation. In this paper, however, we focus on two particular IE methods, namely: the Pattern based IE method [1] and the Co-occurrence based IE method [12]. Our choice is based on the high effectiveness of these methods as shown in [12]. Investigating other methods remains part of our future work.


Learning Query: "Jack M. Davis + jdavis@mit.edu"
Learning Query: "Tom Smith + tomsmith2@cs.cmu.edu"
… question, please send email to Jack M. Davis (Email: jdavis@mit.edu)
…, please feel free to send email to Tom Smith (Email: tomsmith2@cs.cmu.edu)
(Auxiliary Information)
Patterns : send email to [NAME] (Email: [EMAIL]) ……
<Bill Wilson, ?>
Imputation Query Formulation
"send email to Bill Wilson (Email: +)"
Fill
billwilson@uiuc.edu
Extract
… send email to Bill Wilson (Email: billwilson@uiuc.edu)…
Learning Query: "Jack M. Davis + Professor + MIT"
Learning Query: "Bob Brown + A/Prof. + Yale"
MIT → Department of Electrical Engineering and Computer Science → Faculty List:... Jack M. Davis Professor of Computer Science and Engineering in the.
Joined Yale Faculty 1982... Office location:... Telephone: ... Bob Brown is an A/Prof. of computer science department at Yale, ...
(Auxiliary Information)
Common Context Terms: Faculty, department
<Ama Jones, Ms., ?>
Imputation Query Formulation
"Ama Jones + Ms. + (Faculty OR department)"
Fill
UIUC
Extract
UIUC → Department of computer science... Faculty … Prof. Ada Janes Ms. Ama Jones …

(a) P-DI (b) C-DI

Fig. 1. Example Learning and Retrieving Process of the Two Retrieval Ways

Pattern based Data Imputation (P-DI) Our pattern based data imputation extends the classical Pattern based IE method [10, 1], which relies on syntactic patterns to identify instances of a given entity type (such as *University*) or relation type (such as *(University, State)*). Early research in that area focused only on high-quality patterns known as Hearst Patterns (such as *"... such as..."*) to identify new entities or relations [10]. Later, bootstrapping Pattern based IE methods were proposed, in which patterns learned from seed instances are used to find more instances of the same type (such as *Snowball* [1]).

Applying pattern based data imputation in WebPut involves learning and using auxiliary information in the form of patterns, which is accomplished via the following three tasks: (1) identifying all possible utilized and target attributes pairs as $< X_i, y_j >$ based on the existing attribute values and missing values per tuple, (2) learning auxiliary information $A_{i,j}$ for each possible pair $< X_i, y_j >$ in the set of complete tuples, and (3) applying those learned auxiliary information to formulate data imputation queries for incomplete tuples.

For example, as shown in Figure 1(a), to learn auxiliary information (i.e., patterns) based on complete instances such as *("Jack M. Davis", "jdavis @mit.edu")* and *("Tom Smith", "tomsmith2 @cs.cmu.edu")*, we issue a *Learning Query* based on each one of those complete tuples. A learning query is a Web search query that returns a set of documents that are further utilized for pattern extraction. In particular, from the retrieved documents, we may learn patterns corresponding to $< \{Name\}, Email >$ such as: “send email to `[NAME]` (Email: `[EMAIL]`)” (as shown in Figure 1(a)). Finally, we can easily formulate a data imputation query for each tuple with a missing `Email` value using the values of `Name` and the extracted pattern. For example *“send email to Bill wilson (Email:”* + *“)”*(The string in a quotation is taken as an unseparated keyword) and extract the missing `Email` value from the retrieved documents.

Co-occurrence based Data Imputation (C-DI) Co-occurrence based data imputation extends the co-occurrence based IE method [12]. In the context of IE, the co-occurrence based method was proposed to circumvent the limitations of the pattern based approach, given that patterns are sometimes too *strict* to capture most of the existing entities or relations on the web. In particular, a Co-occurrence based extraction method learns *common context terms* instead of patterns from seed instances of a given relation [12]. For example, from instances such as (MIT, MA), (CMU, PA), (Yale, NY), we could learn some common context terms for the relation (University, Location) such as “located at”, which are mentioned closely and frequently with these instance pairs in some web pages. With these context terms, we expect to find the Location of another university like (UIUC, ?). Different from pattern-based extraction, the co-occurrence based extraction method relies on *Named Entity Recognition(NER)* [14] to extract the entity or relation from the documents.

Employing C-DI in WebPut involves almost the same three steps as the P-DI. There are three minor differences, however, which could be observed in the example in Figure 1(b): (1) C-DI learns common context terms, instead of patterns, as the auxiliary information, (2) The formulated imputation query only requires the retrieved documents to contain the common context terms at any position, (3) From the retrieved documents, we apply NER to identify all Universities, the one with the highest frequency will be taken as the missing value we are looking for.

Multiple Data Imputation Queries: To simplify the presentation, so far we have assumed that all the learning queries for each $< X_i, y_j >$ pair will return the same auxiliary information, being a pattern or a set of context terms, and in turn result in formulating a single data imputation query. For instance, Figure 1(a) shows that all the learning queries return the same pattern *“send email to* `[Name]` *(Email:*`[Email]`*)”*. In reality, however, WebPut could extract multiple auxiliary information for the same $< X_i, y_j >$ pair from the retrieved documents. For instance, the learning queries in Figure 1(a) could have returned other patterns such as: *“contact* `[Name]` *through* `[Email]`*”* and *“Name:* `[Name]`*, Email:* `[Name]`.”. Similarly, for each target attribute y_i there might exist multiple sets of possible utilized attributes: $X_1, X_2, ...$ For example, for the $\{$`Email`$\}$ target attribute, $\{$`Name`$\}$ could be a utilized attribute (as shown Figure 1(a)) but also all other combinations of attributes form a valid set of utilized attributes (e.g., $\{$`Name`, `University`$\}$, etc.). As such, for each y_j WebPut can formulate a significantly large number of data imputation queries. The decision of which of these queries to use brings additional complexity to our WebPut approach which is addressed in the next section.

3.2 Confidence-based Selection of Data Imputation Queries

In this section, we propose the use of confidence values as a method for effective and efficient data imputation in the presence of multiple applicable data imputation queries. As discussed in the previous section, it is typically the case that WebPut can formulate a significantly large number of data imputation queries for each target attribute y_j. In particular, for each y_j WebPut formulates a set of imputation queries $\mathbb{Q}_j$ where the number of queries in $\mathbb{Q}_j$ is equal to the number of valid combinations of utilized attributes and auxiliary information that is applicable to the target attribute y_j and each $q(y_j) \in \mathbb{Q}_j$ represents exactly one of those valid combinations.

In a naive implementation, WebPut would fire all queries in $\mathbb{Q}_j$ for the imputation of each y_j and select for imputation the value agreed upon by most of the queries in $\mathbb{Q}_j$ (i.e., high frequency). This approach is inefficient as it incurs a large overhead in terms of the number of issued queries and their corresponding delays. Moreover, applying a voting mechanism based on frequency might jeopardize the accuracy of imputation since frequency does not necessarily translate into high-quality.

In the rest of this section we focus on our proposed confidence-based scheme for effective imputation. In summary, our scheme *ranks* each candidate value returned by a a data imputation query $q \in \mathbb{Q}_j$ according to two factors: 1) the confidence in the data imputation query, and 2) the confidence in the values utilized by that query. Then in turn selects the value with the highest rank.

The first factor (i.e., confidence of data imputation query) is simply based on the query's success in retrieving values similar to those already existing in the complete tuples during the learning phase. Hence, our scheme naturally relies on the concepts of *coverage* and *support*, which are defined as:

Definition 1. *A data imputation query $q(y_j)$ **covers** a target attribute y_j in tuple t, if and only if: (1) all the utilized attribute values of $q(y_j)$ exist in t; and (2) $q(y_j)$ could retrieve a value for y_j.*

Definition 2. *A complete tuple t **supports** a data imputation query $q(y_j)$, if and only if: (1) $q(y_j)$ covers t; and (2) the missing value returned by $q(y_j)$ equals to the existing value of the target attribute y_j in the complete tuple t.*

The second condition in Definition 1 is necessary because sometimes a query might fail to retrieve any value for its target attribute. Moreover, notice that even though the definition of support is based on that of cover, it only applies for complete tuples.

1. Confidence of Data Imputation Queries Given the definitions of both coverage and support, the *confidence* of a data imputation query is estimated as:

$$Conf(q) = \frac{|Support(q)|}{|Cover(q)|} \tag{1}$$

where $|Cover(q)|$ is the number of complete tuples that are covered by q, whereas $|Support(q)|$ is the number of complete tuples that support q, as defined above.

Notice, however, that a data imputation query q is only *valid* (i.e., can be applied to impute missing values), if and only its confidence and cover are higher than predefined thresholds τ and η respectively. This approach is mainly to avoid employing outlier queries (i.e., low cover) or poor-quality queries (i.e., low support). The values of τ and η are system's parameters and their settings are discussed in Section 4.

2. Confidence of Imputed Values The confidence of a new imputed value is directly computed according to the confidence of its corresponding data imputation query q as well as the set of utilized attribute values X. More specifically,

$$Conf(y_{j,t}, q) = Conf(q) \prod_{x_i \in X} Conf(x_{i,t}) \tag{2}$$

where the confidence of each utilized attribute value $x_{i,t}$ is either pre-determined (for existing values) or it is computed (for already imputed values) in the same way. When several data imputation queries are applied to a missing value, and different data imputation queries lead to different values, only the value with the highest confidence will be used to impute the missing value.

3.3 Efficient Scheduling of Data Imputation Queries

WebPut aims to impute each missing value with the highest-confidence value while at the same time minimizing the number of issued data imputation queries. In this section, we propose efficient algorithms for the scheduling of data imputation queries that achieve these goals. As opposed to the naive approach discussed in the previous section (Sec. 3.2), the following algorithms share the principle idea of leveraging the confidence of data imputation queries to achieve high efficiency and effectiveness.

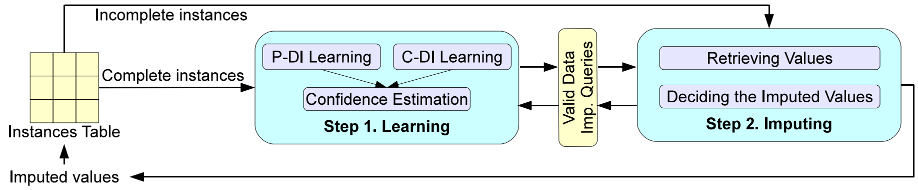
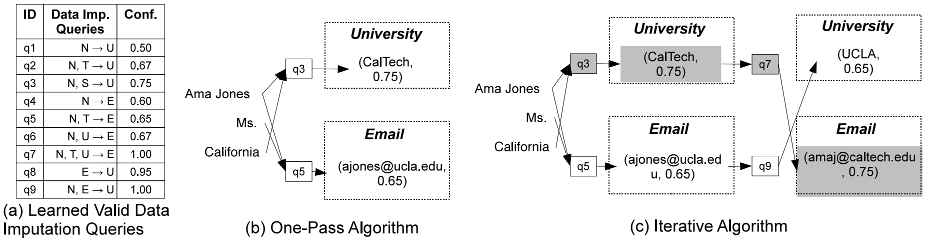


Fig. 2. The Architectures of the Iterative WebPut Algorithm

1. One-Pass Scheduling Algorithm (Baseline) In this baseline algorithm, WebPut imputes all missing values in one pass. First, based on existing complete tuples, it estimates the confidence of all the data imputation queries targeting any incomplete attributes in the given database. Second, it uses those valid data imputation queries to retrieve values for missing values. However, when several queries are available for the same missing value, it selects the one with the highest confidence to be issued under the assumption that it should provide the highest confidence value.



ID	Data Imp. Queries	Conf.
q1	N → U	0.50
q2	N, T → U	0.67
q3	N, S → U	0.75
q4	N → E	0.60
q5	N, T → E	0.65
q6	N, U → E	0.67
q7	N, T, U → E	1.00
q8	E → U	0.95
q9	N, E → U	1.00

(a) Learned Valid Data Imputation Queries

(b) One-Pass Algorithm

(c) Iterative Algorithm

Fig. 3. Imputation Results in an Example Tuple with Two Different Algorithms

For example, consider the 5th tuple in Table 1 in which there are two missing values (=Email and University) and three existing values (=Name, Title, State). As shown in Figure 3(a), data imputation queries q_1, q_2, q_3 can be applied to the University missing value, among which q_3 has the highest confidence. Meanwhile q_4 and q_5 can be applied to the Email missing value, among which q_5 has the highest confidence. By applying both of the high-confidence queries (i.e., q_3 and q_5), we assign “CalTech” with 0.75 confidence for the University missing value; and “ajones@ucla.edu” with 0.65 confidence for the Email missing value as shown in Figure 3(b). While both values have a reasonable confidence, they are inconsistent with each other: people from one University are unlikely to use an email belonging to another university. This example shows that one-pass algorithm suffers from one major drawback: the imputed value of one imputed missing data is not further utilized to impute values of other missing data. Consequently, it may lose the chance to retrieve a better value with a higher confidence for some missing values.

2. Iterative Scheduling Algorithm To address the drawbacks of the one-pass algorithm described above, we introduce an iterative scheduling algorithm. In each iteration of this algorithm, we impute the missing values, similar to the one-pass algorithm. However, after each iteration, the imputed values are stored in an *intermediate* instance database (Figure 2). In the next iteration, some of those imputed values are further utilized to impute already imputed ones if they are of high confidence and they support high-confidence queries.

To further illustrate the iterative algorithm, let’s consider again the 5th tuple in Table 1. After the 1st iteration (Figure 3(c)), we fill the two missing missing values (Email and University missing values) with two imputed values, which are similar to those provided by the one-pass algorithm. In the 2nd iteration, however, two more queries (=q_8 and q_9) that target the University missing value become available as they can utilize the Email value imputed in the 1st iteration. Similarly, queries q_6 and q_7 targeting the Email missing value become available as they can utilize the University value imputed in the 1st iteration. According to our query rankings, we apply q_7 and q_9 in this iteration, and we get “amaj@caltech.edu” with 0.75 confidence for the Email missing value; and “UCLA” with 0.65 confidence for the University missing value. Since the confidence of the new value “amaj@caltech.edu” is higher than the value “ajones@ucla.edu” obtained in the 1st iteration, we replace the imputed value of Email with the new one. However, note that our schedule prevents cycles, that is, the value of missing value y_1 that is imputed by utilizing the value of missing value y_2 cannot be utilized in imputing values for missing value y_2. Thus, we could stop the iterative algorithm at this moment. Now we have more consistent imputed values for the two missing values.

3. Greedy Iterative Scheduling Algorithm In the above example, four data imputation queries (q_3, q_5, q_7, q_9) were applied, while only two (q_3 and q_7) provided the highest-confidence values for the two missing values. Motivated by this observation, we present a greedy shceduling algorithm, which identified “effective” data imputation queries that provide the highest-confidence values for missing values apriori, and in turn, minimizes the number of issued queries. For an incomplete tuple, all identified data imputation queries are processed in a particular order so they form a *greedy schedule* for each incomplete tuple.

The optimal schedule is *acyclic* and associates all missing values with the minimum number of data imputation queries, as shown in the highlighted path in Figure 3(c).

To generate that optimal schedule, in each iteration we only select one data imputation query that is estimated to provide the highest-confidence attribute value for a not imputed missing value. As long as the data imputation query could provide a value for that missing data, we impute the missing data with the already imputed values. However, if no value is provided, WebPut drops that query and re-selects a new data imputation query from the remaining unused imputing queries based on the same rule. Finally, when all missing values are imputed or there are no more unprocessed data imputation queries, the algorithm stops. The optimal schedule for the 5th tuple in Table 1 is highlighted (shaded boxes) in Figure 3(c).

The greedy version of the iterative algorithm always provides the same imputed values as the pure iterative algorithm described earlier. However, in the best case scenario, the greedy algorithm requires only one data imputation query to apply per missing value, which is more efficient than the iterative algorithm.

4 Experimental Evaluation

We have implemented a WebPut prototype in Java which uses Google API to answer data imputation queries. We have experimented with three real-world data sets:

- *Multilingual Disney Cartoon Table (Disney):* This table contains names of 51 classical disney cartoons in 8 different languages collected from Wikipedia.
- *Personal Information Table (PersonInfo):* This is a 2.5k-tuples, 5-attributes table, which contains contact information for academics including name, email, title, university and country. This information is collected from 20 different universities in the USA, UK and Australia.

- *DBLP Publication Table (DBLP):* This is a 10k-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.

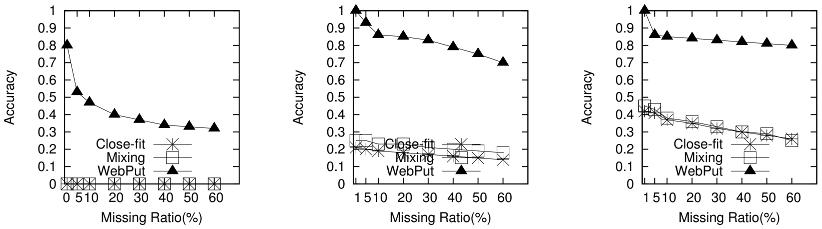
The three data sets above are complete relational tables. To generate incomplete tables for our experiments, we remove attribute values at random positions from the complete table, while making sure that at least one key attribute value will be kept in each tuple. (For the Disney dataset, all attributes are key attributes. For the PersonInfo dataset, the name and email are key attributes. For the DBLP dataset, the paper title is the only key attribute.) Each reported result is the average of 5 evaluations, that is, for each missing 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 WebPut and evaluate the performance of our solutions by using the original complete table as the ground truth.

4.1 WebPut v.s. related approaches

We compare the accuracy of WebPut against two state-of-the-art data imputation approaches: (1) *Close-fit*: a substitute-based approach which finds a “close-fit” value from a similar context using association rules [20]; (2) *Mixing*: a model-based approach where a mixture-kernel based iterative estimator is advocated to impute mixed-attribute datasets [23]. The accuracy of an algorithm is the percentage of missing values filled with correct values among all missing values in the table.

As shown in Figure 4, WebPut (using the greedy iterative scheduling algorithm) reaches a much higher accuracy than both of the other two algorithms for the 3 data sets at all the missing ratios. For the 1st data set both *Close-fit* and *Mixing* can not impute any missing values at all, since all values in this dataset are unique and thus missing values are unlikely to be imputed either from a similar context within the dataset or through a established model built on the dataset. For the same reason, the two methods impute no more than 50% of missing values on the other two datasets.

The performance of WebPut is determined by whether the missing values could be retrieved from the web based on existing values. For the 1st data set, since only the English name of a Disney movie is usually mentioned together with names in other languages, once the English name is missing, it is pretty hard to retrieve missing names in other languages through existing names in other languages. With the iterative algorithm, we may impute the missing English name first, and then identify other names through the imputed English name. But once the English name is incorrect, all the other names that imputed based on the English names are also incorrect. Therefore, the accuracy on the 1st data set is not high. For the other two data sets, as long as a key value of a tuple exists, such as the name in PersonInfo, or the paper title in DBLP, it is not difficult to impute the missing values based on the key values. As a result, WebPut retrieves approximately 70-80% of missing values for the two data sets.



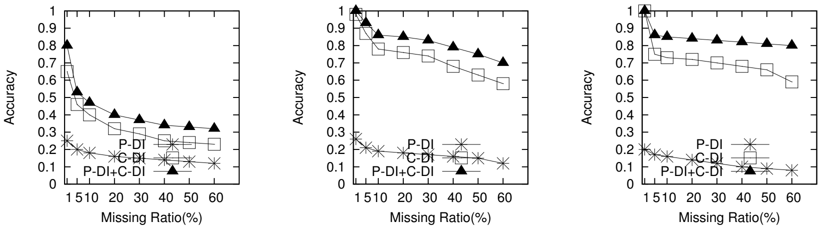
(a) Disney (b) PersonInfo (c) DBLP

Fig. 4. Comparing the Accuracy of WebPut against Previous Approaches

4.2 P-DI v.s. C-DI

We measure the effectiveness of WebPut when the data imputation queries are formulated based on: 1) only P-DI; 2) only C-DI; and 3) a suite of P-DI and C-DI formulated queries (as described in Section 3.1). In this experiment, for a query to be valid it must be supported by at least 3 complete tuples and its confidence must be more than 0.3. For P-DI, the length of Prefix and Suffix in each pattern must be at least 5 characters long. Finally, when applying a pattern to extract the value of an entity, that candidate value must be within 300 characters. For C-DI, we adopt all the settings from [12] to get the best expansion terms for each query.

As shown in Figure 5, the accuracy of using both P-DI and C-DI queries in WebPut is always higher than that of using either one alone. For instance, P-DI can only fill up to 20% of the missing values due to the strictness of pattern matching. While C-DI can fill much more missing values than P-DI due to its flexibility, the combination of both still achieves the highest accuracy.



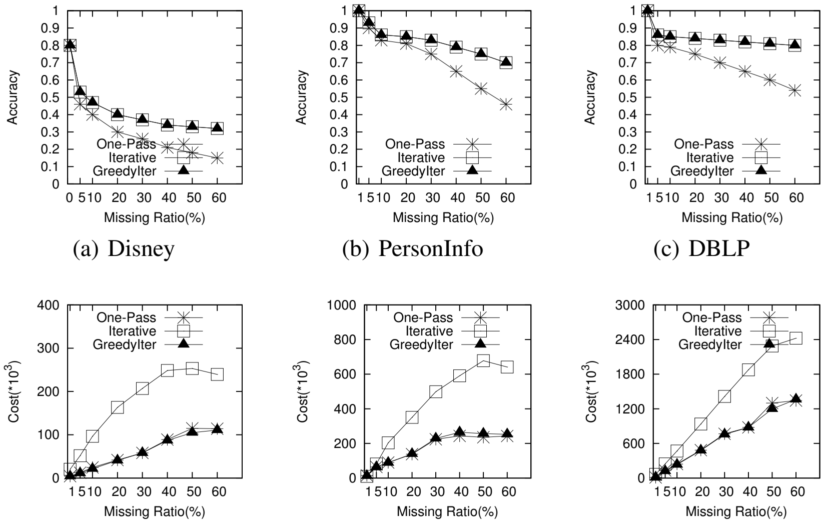
(a) Disney (b) PersonInfo (c) DBLP

Fig. 5. Comparing the Accuracy of the different Query Formulation Methods

4.3 Evaluation of the Scheduling Algorithms

In this experiment, we first evaluate the accuracy of three WebPut algorithms proposed in Section 3. As shown in Figure 6(a)(b)(c), when the missing ratio is low (<5%), the accuracy of the three algorithms is almost the same. As the missing ratio increases from 5% to around 50%, the accuracy of *One-Pass* drops faster than *Iterative* for all the three data sets. When the missing ratio becomes more than 50%, the accuracy of *Iterative* is about 20% higher than that of *One-Pass*. The figure also shows that the accuracy of *Greedy Iterative (GreedyIter)* is always the same as that of *Iterative* as discussed in Section 3.3.

In Figure 6(d)(e)(f) we compare the costs of the 3 algorithms measured in terms of the number of data imputation queries issued by WebPut. The figure shows that the imputation costs of both *GreedyIter* and *One-Pass* is approximately proportional to the missing ratio, since both of them need only one query to impute each missing value. The cost of *Iterative*, however, increases much faster than *GreedyIter* and *One-Pass*. For instance, when the missing ratio is larger than 50%, the cost of *Iterative* is about 2 times that of *GreedyIter*. From the two figures, it is clear that *GreedyIter* reaches the best accuracy at the minimum cost as intended.



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

(d) Disney (e) PersonInfo (f) DBLP

Fig. 6. Comparing the Accuracy and Cost of the Scheduling Algorithms

5 Related Work

Data imputation aims at providing estimations for missing attribute values by reasoning from observed data on a data set [3]. Existing data imputation approaches can be divided into three categories: (1) substitute-based data imputation; (2) model-based data imputation; (3) external resource based data imputation.

The substitute-data imputation approaches arbitrarily find a substitute value for the missing one from the same data set. Nine approaches have been introduced and compared in [8], such as selecting the "most common attribute value" as the missing value, or "assigning all possible values of the attribute restricted to the given concept", or "treating missing attribute values as special values" and so on. Later, k-Nearest Neighbor [7] and association rules [20] were also used to find a "close-fit" value from a similar context. The substitute-based data imputation approaches attempt to smooth the influence of missing values on statistical data analysis results by replacing them with suitable substitutes, but are unlikely to find the right missing attribute values for missing values, especially when the missing value is a unique one within the data set.

The model-based data imputation approaches build a prediction model based on the data set, and then estimate a value for a missing value based on the model. Among the proposed approaches, some [2, 18] were developed for continuous attributes only, while others [15, 22] were designed to deal with discrete attributes. There are also some approaches [21, 23] targeting imputing mixed attributes, which either take the discrete attributes as continuous ones, or smooth the mixed regressors. The model-based data imputation approaches have advantage on estimating a vey close value for the missing one, which could greatly smooth the influence brought by incomplete data. However, close estimations can not replace the missed original values, especially the values of discrete attributes.

The third type of data imputation approaches aim at finding missing attribute values from external resources. The WebPut belongs to this category. This type of data imputation approach do not find substitutes or estimations for the missing ones, but the missing values themselves from external resources. There were some effects on augmenting a table with very few example rows by constructing new rows from unstructured lists on the web [9]. Although table augmentation does not have the same purpose with data imputation, it is also extracting data from external sources and put them into local table.

6 Conclusions and Future Work

In this paper, we present WebPut - an approach to impute missing data from external data sources on the world wide web. In that sense, WebPut could be perceived as a novel Information Extraction (IE) approach, which formulates the extraction tasks around the missing attribute values in a database. Our experimental results based on several real-world data collections demonstrate that WebPut can effectively retrieve a large percentage (approximately 70-80%) of correct missing values in an incomplete table, outperforming previous approaches.

An underlying assumption of the work presented in this paper 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 are also working on a hybrid approach that combines and integrates our web-based approach with previous model-based data imputation methods.

Acknowledgments: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by Australian Research Council grant DP110102777.

References

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *ACM DL*, 2000.

[2] J. Barnard and D. Rubin. Small-sample degrees of freedom with multiple imputation. *Biometrika*, 86(4):948–955, 1999.

[3] G. Batista and M. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, 2003.

[4] S. Brin. Extracting patterns and relations from the world wide web. *The World Wide Web and Databases*, pages 172–183, 1999.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[6] J. Grzymala-Busse. Three approaches to missing attribute values: A rough set perspective. *Data Mining: Foundations and Practice*, pages 139–152, 2008.

[7] J. Grzymala-Busse, W. Grzymala-Busse, and L. Goodwin. Coping with missing attribute values based on closest fit in preterm birth data: A rough set approach. *Computational Intelligence*, 17(3):425–434, 2001.

[8] J. Grzymala-Busse and M. Hu. A comparison of several approaches to missing attribute values in data mining. In *RSCTC*, 2001.

[9] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *PVLDB*, 2(1):289–300, 2009.

[10] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, 1992.

[11] J. Li and N. Cercone. Assigning missing attribute values based on rough sets theory. *IEEE Granular Computing*, pages 607–610, 2006.

[12] Z. Li, L. Sitbon, and X. Zhou. Learning-based Relevance Feedback for Web-based Relation Completion. In *CIKM*, 2011.

[13] D. Loshin. *The Data Quality Business Case: Projecting Return on Investment*. Informatica, 2006.

[14] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *EACL*, 1999.

[15] J. Quinlan. *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[16] M. Ramoni and P. Sebastiani. Robust learning with missing data. *Machine Learning*, 45(2):147–170, 2001.

[17] S. Shi, H. Zhang, X. Yuan, and J. Wen. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *COLING*, 2010.

[18] Q. Wang and J. Rao. Empirical likelihood-based inference under imputation for missing response data. *The Annals of Statistics*, 30(3):896–924, 2002.

[19] R. Wang and W. Cohen. Automatic set instance extraction using the web. In *ACL/AFNLP*, 2009.

[20] C. Wu, C. Wun, and H. Chou. Using association rules for completing missing data. In *HIS*, 2004.

[21] S. Zhang. Parimputation: From imputation and null-imputation to partially imputation. *IEEE Intelligent Informatics Bulletin*, 9(1):32–38, 2008.

[22] S. Zhang. Shell-neighbor method and its application in missing data imputation. *Applied Intelligence*, 35(1):123–133, 2011.

[23] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu. Missing value estimation for mixed-attribute data sets. *IEEE TKDE*, 23(1):110–121, 2011.