WebPut: A Web-Aided Data Imputation System for the General Type of Missing String Attribute Values

Shuangli Shan
The School of Computer Science and Technology, Soochow University, China
sldan@stu.suda.edu.cn

Zhixu Li*
The School of Computer Science and Technology, Soochow University, China
zhixuli@suda.edu.cn

Yang Li
The School of Computer Science and Technology, Soochow University, China
xxxxx

Qiang Yang
The School of Computer Science and Technology, Soochow University, China
qiangyanghm@hotmail.com

Jia Zhu
South China Normal University, China
jzhu@m.scnu.edu.cn

Mohamed Sharaf
The University of Queensland, Brisbane, Australia
fmsharaf@itee.uq.edu.au

Xiaofang Zhou
The University of Queensland, Brisbane, Australia
zxf@itee.uq.edu.au

ABSTRACT

In this demonstration, we present an end-to-end web-aided data imputation prototype system named WebPut. WebPut consults the Web for imputing the missing values in a local database when the traditional inferring-based imputation method has difficulties in getting the right answers. Specifically, WebPut investigates the interaction between the local inferring-based imputation methods and the web-based retrieving methods and shows that retrieving a small number of selected missing values can greatly improve the imputation recall of the inferring-based methods. Besides, WebPut also incorporates a crowd intervention component that can get advice from humans in case that the web-based imputation methods may have difficulties in making the right decisions. We will demonstrate, step by step, how WebPut fills an incomplete table with each of its components.

1 INTRODUCTION

Having missing values is a common problem for databases, and the process of filling in the missing attribute values is well known as data imputation [8, 11]. So far, various imputation methods have been developed for missing quantitive data (either continuous or discrete data), while only limited attention has been paid to non-quantitive data (pure string data with a large scope of values) [6]. However, pure string data takes up a large part of the missing data in many databases, and it can be seen as a general type of data since all kinds of data can be taken as a special type of string data.

Existing imputation methods to the general type of non-quantitive string data mainly rely on some local data constraints (such as FD/CFDs) [1, 10, 12] or some prediction models [11, 14] to infer substitutes or estimations for the missing values. However, the inferring-based methods are always failing to get the right missing value for the lack of enough knowledge, especially when the missing value is unique in the data set. In view of the drawbacks of the previous methods, recent systems turn to involve crowdsourcing for help, such as outsourcing the task to a crowdsourcing platform to obtain answers from crowd workers [3, 13]. Crowd-based imputation methods have shown their effectiveness on improving imputation precision and recall, as well as on reducing the cost of human interventions. However, the cost of human resources are always expensive. In addition, some required human interventions from experts may not always be available, which brings further obstacles to the applicability of the imputation systems.

Why not use a much cheaper and within reach knowledge depository, i.e., the Web, in data imputation? Compared with crowdsourcing, the Web has at least four advantages: First, it is almost free. Usually there is no need to pay for getting information from the surface Web. Second, consulting the Web through web search engines has no restrictions on time and locations since the Web does not sleep and can be accessed everywhere. Third, consulting the Web can be very fast especially when it is performed in parallel. Fourth, as the world's largest repository of human's knowledge, the Web naturally contains enough raw data and knowledge, either structured (such as Wikipedia) or unstructured, to support the imputation to a wide range of datum in databases of different domains.

The four advantages above motivate us to develop a Web-Aided Data Imputation (or **WebPut** for short) system, which implements a web-based data imputation module, and finds ways to integrate the web-based imputation with existing inferring-based data imputation methods. Besides, a crowdsourcing component is incorporated to seek advice from human to help control the quality of imputation results from the Web in a cost-efficient way. To make WebPut a general imputation tool for various types of missing textual values, the current system takes every textual value of a particular type (like numeric, email address, currency) as a kind of string, based on which we design all our imputation techniques on the string data only. As a future work, we will integrate the imputation techniques that are specially designed for some particular data type into the system.

*Corresponding author

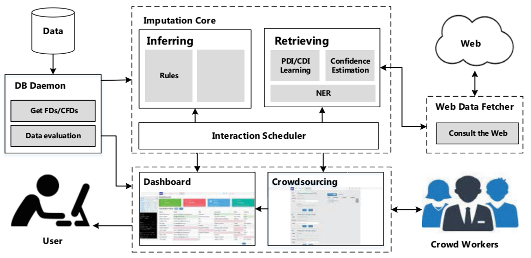


Figure 1: The System Architecture of WebPut

2 SYSTEM OVERVIEW

Figure 1 shows the building blocks of WebPut. Basically, there are five main components: the DB Daemon, the Imputation Core, the Web Data Fetcher, the Interaction Scheduler, and the Crowdsourcing Interface. We introduce the design of these components respectively as follows.

1) DB Daemon: This module is responsible for importing data from databases and collecting quality rules such as FD/CFDs and user-defined rules. In addition, the DB daemon verifies the completeness of all tuples and identifies the missing values for imputation.

2) Imputation Core: The imputation approaches we adopt in WebPut include both traditional inferring-based imputation method and retrieving-based imputation method.

(1) Inferring-based Imputation: The Inferring-based imputation APPROACH we adopt in WebPut relies on data constraints, such as FD/CFDs, to infer the missing values from the complete part of the dataset [2, 5].

(2) Retrieving-based Imputation: The retrieving-based imputation approach can be perceived as a novel Information Extraction (IE) approach, which formulates the extraction tasks around the missing attribute values in a database. It utilizes the available information in an incomplete database in conjunction with the data consistency principles, and extends effective IE methods for the purpose of formulating web search queries that are capable of effectively retrieving missing values with high accuracy. Details of this part will be covered in Sec. 3.1.

3) Web Data Fetcher: This component fetches required data (including web pages and attribute values) from the Web according to the input queries and information extractors. It mainly works for the retrieving-based imputation, but is separated as an independent part since it may be adjusted more adaptively to the external Web environments. Brief details on this component will be given in Sec. 3.2.

4) Interaction Scheduler: This module is responsible for the interaction between retrieving-based imputation and inferring-based imputation, given that the interaction between the two can benefit from each other in reaching a higher imputation recall while minimizing the Web consultation cost by having the least number of values for retrieving-based imputation. Details will be given in Sec. 3.3.

5) Crowdsourcing Module: We incorporate a crowdsourcing module that gets advice from humans in case that the web-based imputation may have difficulties in making the right decisions. Basically, there are two ways for the crowds to input suggestions depending on the web-based/inf-erring-based imputation results to a certain missing value. One way is to make selection decisions when there are multiple candidates, while the other is to input the answer directly when there is no good candidate. We briefly introduce the two ways below, and will give more details in Sec. 3.4.

(1) Selection Making: In a basic situation, multiple candidate imputation values might be gotten for one missing value. Crowd workers are assigned to a selection task, in which a drop-down list will be generated containing all alternative imputation values, while the relevant information about the missing value is provided as hints.

(2) Direct Value Suggestion: In contrast to selection making, crowd workers can also make direct value suggestion in two situations: (1) the crowd worker may want to provide another answer that is not listed as an option; (2) no candidate value is provided by the inferring-based and retrieving-based imputation methods. In both situations, crowd workers are asked to fill out a form where an input box is provided for the missing value. Hint and relevant information about the missing value is presented likewise.

3 CORE TECHNIQUES

In this section, we introduce the challenges we meet in WebPut towards effective and efficient data imputation and the corresponding solutions we propose.

3.1 Retrieving Query Formulation

The Web is not clean at all, thus we may take the risk to bring web noises into the local database. Existing work on web data extraction estimates the quality of the extraction according to various factors , such as the employed patterns and the confidence of various web sources. But in our case, the quality of each formulated retrieving query also needs to be taken into account. To effectively retrieve a missing attribute value in an incomplete tuple from the Web, the retrieving-based method utilizes the available information in the incomplete tuple in conjunction with the data consistency principles, and extends effective IE methods for the purpose of formulating web search queries. Specifically, we face two challenges here: (1) For a missing attribute value v in an incomplete tuple $t = [x_1, x_2, ...\square, ...x_n]$ (where $\square$ denotes the position of v in t), all the existing attribute values could possibly be utilized in various combinations as the "keywords" in a imputation query, but will bring much noises with a large overhead as most of these queries are not very selective. As an alternative, we find ways to estimate the effectiveness of every possible combinations, by which we can identify a number of combinations that work the best for the missing value v in t. (2) For each combination of existing attribute values, we also need some "auxiliary information" to bridge between the query keywords and the target missing value v, such that we can define the purpose of an imputation query and improve the selectivity of the query towards the purpose.

1) Query "Keywords" Selection: We estimate the effectiveness of every query template (such as [`TITLE, NAME` $\rightarrow$ `EMAIL`]) that is formulated by a combination of leveraged attributes (such as TITLE, NAME here) and the corresponding target attribute (such as EMAIL here) based on complete instances. More specifically, a concrete search query is an instantiation of a query template on a single tuple, by which we retrieve the corresponding imputation value from the Web and then check if the imputation value equals to the existing target value in the tuple. In particular, we say a query template q covers a complete tuple t if its instantiation on t can retrieve values for the target attribute, and we say the tuple t supports a query template q if the instantiation of q on t can retrieve the right missing value for the target attribute. Straightforwardly, the effectiveness of a query template q is the percentage of tuple that support q among all tuples that covered by q in the trained set. That is,

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

where $|Cover(q)|$ denotes the number of tuples that covered by q, and $|Support(q)|$ denotes the number of tuples that support q. Naturally, a query template with the highest confidence is believed to be more capable of retrieving the right value, and should be employed at first.

2) Getting "Auxiliary Information": Currently, we extend two effective information extraction methods, namely pattern based [4] and co-occurrence based [7] methods, to optimize our imputation queries.

(1) Pattern based method: Given a query template say [`TITLE, NAME` $\rightarrow$ `EMAIL`], this method learns common patterns from complete tuples, such as"contact [`TITLE`] [`NAME`] through email [`EMAIL`]", such that we can instantiate the query template in the format of such pattern say "contact Dr Jack M.Davis through email" to get Jack's email from the Web.

(2) Co-occurrence based method: This method learns common terms that frequently appear with leveraged and targeted attributes, such as the term "locate in", when is used right after an `UNIVERSITY`, usually indicates that a `LOCATION` is likely to follow up. Accordingly we can instantiate the query template [`TITLE, NAME` $\rightarrow$ `UNIVERSITY`] with learned common term "university" in the format of "Ms. Ama Jones+ university". For more details of our implementation, please refer to [8].

3.2 Web Data Fetcher

The web data fetcher module serves for the retrieving-based imputation method by: (1) retrieving web documents from the Web according to an input query; and then (2) extracting the target attribute value from the retrieved web documents. In general, web data fetcher encodes input queries together with optional parameters and submits to Bing or Google search API [1]. Latter, feedbacks in JSON format are returned, containing abstract information of relevant webpages. Then we relay on Stanford Name Entity Recognition [9] to extract substrings denoting the expected type of entities from those documents. However, the quality of an imputed value depends on not only the employed query template, but also the quality of the leveraged values that are used in instantiating the query template. Having this observation, we define the confidence of the imputed value y_t to help rule out impractical imputed results as follows.

$$Conf(y_t) = Conf(q) \prod_{x_i \in X} Conf(x_i) \quad (2)$$

where q denotes the query and X denotes all the attribute values utilized to form the query.

To avoid erroneously adopting low-confidence imputed values, both the coverage and confidence of the queries as well as the confidence of the imputed values should be higher than predefined thresholds respectively.

3.3 Interactive Inferring and Retrieving

Compared to the inferring-based imputation, the retrieving-based imputation is able to reach a higher imputation recall, but with a much larger overhead spend in consulting the Web (retrieving web documents and extracting the target values from the retrieved documents). To take advantage of both methods, we propose to combine the two methods together in doing the imputation.

In particular, we perform inferring and retrieving alternatively based on a simple principle: identifying the minimum set of missing values for retrieving to maximize the number of values that can be inferred.

The inferring step fills all inferred missing values to the maximum extent, and the subsequent retrieval step retrieves a selected set of missing values so that a set of unfilled missing values can be inferred at the next inferring step. We keep on repeating the two steps until there is no more missing values can be imputed. However, some missing values are only inferable when certain blanks are filled with specific imputed values, which can not be predicted a prior. So the selection of values for retrieving will have an impact on the succeeding inferring and retrieving step.

To achieve the optimal scheduling scheme that can fill all missing string values at the minimum web consultation cost, the basic rule is to retrieve only two kinds of values: 1) those can't be inferred even all the other missing values are imputed (un-inferable), and 2) those form an inference deadlock with some other missing values, where an inference deadlock refers to a group of missing values, in which each value in the group is possible to be inferred only when some other values in the group are imputed firstly. In our implementation, we build an inference dependency graph when it comes to the retrieving step, and then introduce a scheduling algorithm to identify the minimum set of missing values for retrieving. More details about this have to be omitted due to space limitation and can instead be found in [6].

3.4 Crowdsourced Interventions

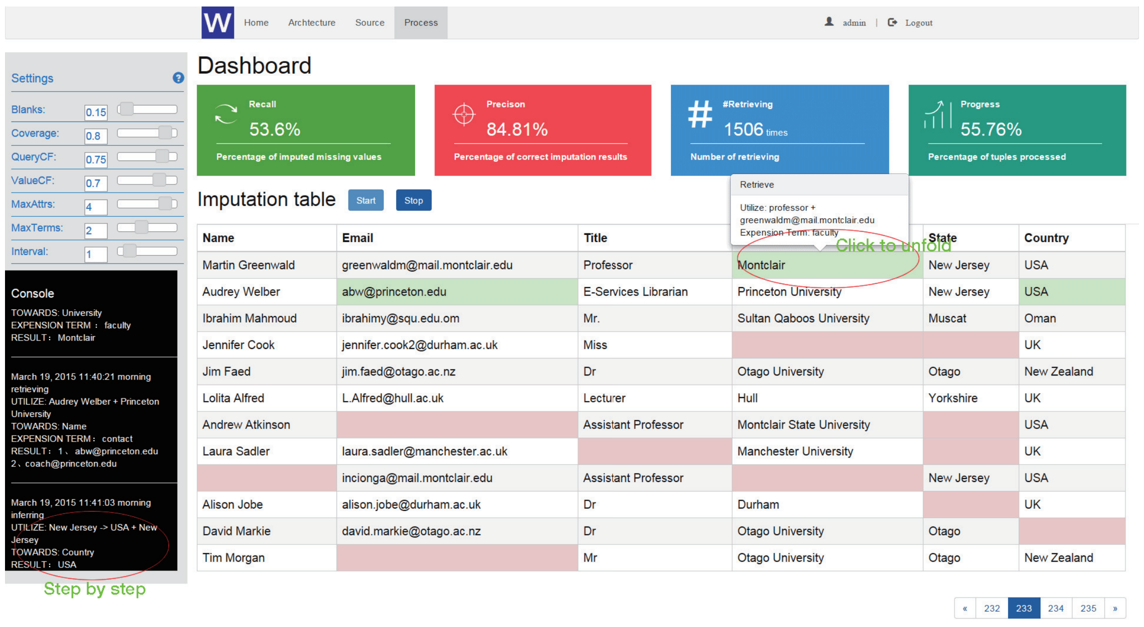
A crowdsourcing module is incorporated to have the crowds help make decisions when we have multiple qualified answers or directly input an answer for a particular missing value. The basic workflow can be described as follows: when multiple answers with reasonable confidence are gotten, the one(s) with the highest confidence will be temporarily adopted in an intermediate table, and a crowdsourcing task will be generated with all these answers. Crowd workers who login the system will be presented with a user-friendly interface for each particular task containing: 1) a selection box for users to select from a list of limited options; 2) a pure input box for inputting other answer, and 3) some extra information about the missing value in the task, such as the existing attribute values in the same tuple.

WebPut incorporates mechanisms for checking the correctness of the input values such as the data type and format. Furthermore, in the presence of malicious workers, we deliberately add some evaluation questions in the list of questionnaire, answers of which are already known, and the credibility of each crowd worker is assessed by how well the input answers fit the right answers. WebPut is designed to outsource crowdsourcing tasks in parallel, thus multiple workers can asynchronously fulfill the imputing task. When there are conflicts between the input answers from different workers, a voting mechanism is employed to consider both the popularity of an answer and the credibility of relevant crowd workers.

4 DEMONSTRATION

We plan an end-to-end demonstration, which visualizes the whole workflow from setting up the system's parameters all the way to exhibiting the final imputation results and some detailed reports on how each value is imputed and the overall statistical reports.

[1]www.bing.com, www.google.com



Settings

Blanks:	0.15
Coverage:	0.8
QueryCF:	0.75
ValueCF:	0.7
MaxAttrs:	4
MaxTerms:	2
Interval:	1

Console

TOWARDS: University
EXPENSION TERM： faculty
RESULT： Montclair

March 19, 2015 11:40:21 morning
retrieving
UTILIZE: Audrey Welber + Princeton University
TOWARDS: Name
EXPENSION TERM： contact
RESULT： 1、abw@princeton.edu
2、coach@princeton.edu

March 19, 2015 11:41:03 morning
inferring
UTILIZE: New Jersey -> USA + New Jersey
TOWARDS: Country
RESULT： USA

Recall	Precison	#Retrieving	Progress
53.6%	84.81%	1506 times	55.76%
Percentage of imputed missing values	Percentage of correct imputation results	Number of retrieving	Percentage of tuples processed

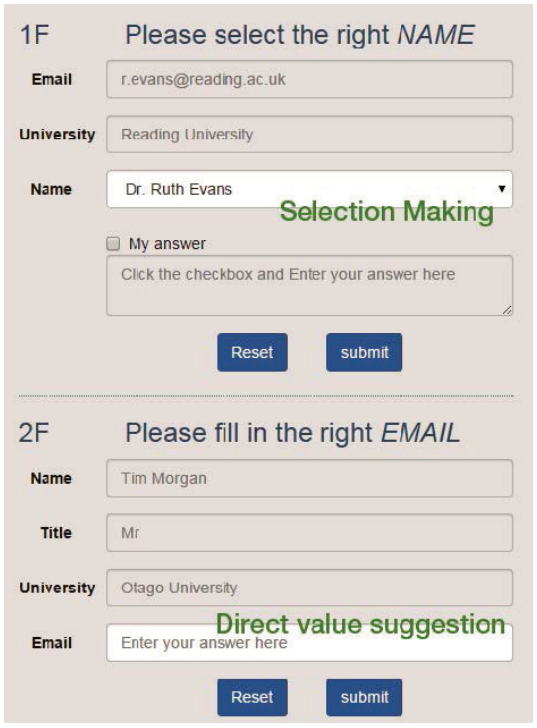
Imputation table

Retrieve

Utilize: professor + greenwaldm@mail.montclair.edu
Expension Term: faculty

Name	Email	Title		State	Country
Martin Greenwald	greenwaldm@mail.montclair.edu	Professor	Montclair	New Jersey	USA
Audrey Welber	abw@princeton.edu	E-Services Librarian	Princeton University	New Jersey	USA
Ibrahim Mahmoud	ibrahimy@squ.edu.om	Mr.	Sultan Qaboos University	Muscat	Oman
Jennifer Cook	jennifer.cook2@durham.ac.uk	Miss			UK
Jim Faed	jim.faed@otago.ac.nz	Dr	Otago University	Otago	New Zealand
Lolita Alfred	L.Alfred@hull.ac.uk	Lecturer	Hull	Yorkshire	UK
Andrew Atkinson		Assistant Professor	Montclair State University		USA
Laura Sadler	laura.sadler@manchester.ac.uk		Manchester University		UK
	incionga@mail.montclair.edu	Assistant Professor		New Jersey	USA
Alison Jobe	alison.jobe@durham.ac.uk	Dr	Durham		UK
David Markie	david.markie@otago.ac.nz	Dr	Otago University	Otago	
Tim Morgan		Mr	Otago University	Otago	New Zealand

Figure 2: Demonstration of Imputing Process of WebPut



1F Please select the right *NAME*

Email	r.evans@reading.ac.uk
University	Reading University
Name	Dr. Ruth Evans

☐ My answer

Click the checkbox and Enter your answer here

2F Please fill in the right *EMAIL*

Name	Tim Morgan
Title	Mr
University	Otago University
Email	Enter your answer here

Figure 3: Two Types of Crowdsoucing Tasks

The Personal Contact Information data set we employ for demonstration is a 7-attributes (Name, Email, Title, University, Country, State, Postcode), 50k-tuples relational table, collected from 100+ universities worldwide.

1) Web-based Operations Demonstration: We will show the fresh users some background knowledge of data imputation and then describe in a top-down style how WebPut works in fetching expected web document that hopefully contains the desired attribute values and extracting the desired candidate values from the web pages.

2) Step-by-Step Interactive Imputation: WebPut can exhibit the process of interactive inferring and retrieving step by step as shown in Figure 2. By default, we set the imputation interval to 1s for visualization purpose, but the users can change the time slots by simply dragging on the interval bar and the speed of imputation will be changed accordingly. Likewise, all the thresholds can be customized during the demonstration.

3) Unfold Imputation Process: Besides displaying imputed results to users, WebPut keeps track of how each imputation value comes into being. As a result, if users are curious about how a specific missing value is imputed, they can double click on the data item and imputation information related to that data will popup.

4) Live Interaction (User or Crowd Worker Input): This is the most interesting part. Every CIKM attendee is encouraged to participate in the demonstration either as a user or as a crowd worker. As a user, one can name a new data set for imputation or input any new person's name and email to the current data set to see if WebPut can get all the rest contact information for the person. As a crowd worker, one can experience how a crowd worker works in the system. Also, we will make our system online after the conference.

REFERENCES

[1] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In ICDE. 746–755.

[2] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional functional dependencies for capturing data inconsistencies. TODS 33, 2 (2008), 6.

[3] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: answering queries with crowdsourcing. In SIGMOD. 61–72.

[4] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In Coling. 539–545.

[5] Solmaz Kolahi and Laks VS Lakshmanan. 2009. On approximating optimum repairs for functional dependency violations. In ICDT. 53–62.

[6] Zhixu Li, Lu Qin, Hong Cheng, Xiangliang Zhang, and Xiaofang Zhou. 2008. TRIP: An Interactive Retrieving-Inferring Data Imputation Approach. TKDE (2008).

[7] Zhixu Li, Mohamed A Sharaf, Laurianne Sitbon, Xiaoyong Du, and Xiaofang Zhou. 2014. CoRE: a context-aware relation extraction method for relation completion. TKDE 26, 4 (2014), 836–849.

[8] Zhixu Li, Mohamed A Sharaf, Laurianne Sitbon, Shazia Sadiq, Marta Indulska, and Xiaofang Zhou. 2012. Webput: Efficient web-based data imputation. In WISE. 243–256.

[9] Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In ACL. 1–8.

[10] Jau-Ji Shen, Chin-Chen Chang, and Yu-Chiang Li. 2007. Combined association rules for dealing with missing values. Journal of Information Science (2007).

[11] Qihua Wang, JNK Rao, et al. 2002. Empirical likelihood-based inference under imputation for missing response data. The Annals of Statistics 30, 3 (2002), 896–924.

[12] Chih-Hung Wu, Chian-Huei Wun, and Hung-Ju Chou. 2004. Using association rules for completing missing data. In HIS. 236–241.

[13] Chen Ye and Hongzhi Wang. 2014. Capture Missing Values Based on Crowdsourcing. In Wireless Algorithms, Systems, and Applications. 783–792.

[14] Shichao Zhang. 2008. Parimputation: From Imputation and Null-Imputation to Partially Imputation. IEEE Intelligent Informatics Bulletin 9, 1 (2008), 32–38.