Dimensional Data KNN-Based Imputation

Yuzhao Yang, Jérôme Darmont, Franck Ravat, Olivier Teste

Dimensional Data KNN-based Imputation*

Yuzhao Yang[1][0000−0002−6552−4812], Jérôme Darmont[2][0000−0003−1491−384X], Franck Ravat[1][0000−0003−4820−841X], and Olivier Teste[1][0000−0003−0338−9886]

[1] IRIT-CNRS (UMR 5505), Université de Toulouse, France
{Yuzhao.Yang, Franck.Ravat, Olivier.Teste}@irit.fr
[2] Université de Lyon, Lyon 2, UR ERIC, France
jerome.darmont@univ-lyon2.fr

Abstract. Data Warehouses (DWs) are core components of Business Intelligence (BI). Missing data in DWs have a great impact on data analyses. Therefore, missing data need to be completed. Unlike other existing data imputation methods mainly adapted for facts, we propose a new imputation method for dimensions. This method contains two steps: 1) a hierarchical imputation and 2) a k-nearest neighbors (KNN) based imputation. Our solution has the advantage of taking into account the DW structure and dependency constraints. Experimental assessments validate our method in terms of effectiveness and efficiency.

Keywords: Data Imputation · Data Warehouses · Dimensions · KNN

1 Introduction

Data warehouses (DWs) are widely used in companies and organizations as a significant Business Intelligence (BI) tool to help them building their decision support systems. Data in DWs are usually modelled in a multidimensional way, which allows the user to analyse data through On Line Analytical Processing (OLAP). An OLAP model organizes data according to analysis subjects (facts) associated to analysis axis (dimensions). Each fact is composed of measures. Each dimension contains one or several analysis viewpoints (hierarchies).

Missing data may exist in a DW. There are 2 types of DW missing data: **dimensional missing data** which are missing data in the dimensions and **factual missing data** which are in the facts. These missing data have impact on OLAP analyses. It is important to complete the missing data for the sake of a better data analysis.

Data imputation is the process of replacing the missing values by some plausible values based on information available in the data [12]. The current DW data imputation research mainly focuses on factual data [25,21,4]. Yet the dimensional missing data make aggregated data incomplete and make it hard to analyse them with respect to hierarchy levels. Therefore the imputation for DW dimensions is also necessary. However the DW dimension has a complex structure containing

* This work is supported by the French National Research Agency (ANR), Project ANR-19-CE23-0005 BI4people (Business intelligence for the people).

different hierarchies with different granularity levels having their dependency relationships. When we complete the dimensional missing data, we have to take the DW structure and the dependency constraints into account. We proposed a hierarchical imputation based on the inter- and intra-dimensional hierarchical dependency relationships [27] for the imputation of dimensional missing data. To the best of our knowledge, there is no other specific data imputation method for DW dimensions. The hierarchical imputation is convincible because we use accurate data based on real functional dependency relationships. However, this method is limited owing to the sparsity problem which means that for an instance to be completed, there may not be an instance sharing the same value on a lower-granularity level of the hierarchy.

In order to complete as many values as possible, in this paper, we propose H-OLAPKNN, an imputation method for DW dimensions by extending the hierarchical imputation with a novel dimension imputation method called OLAPKNN. OLAPKNN is based on K-nearest neighbours (KNN) algorithm. KNN imputation finds the K nearest neighbors of an instance with missing data then fills in the missing data based on the mean or mode of the neighbors' value [23]. We choose KNN because it is a non-parametric and instance-based algorithm, which is widely applied for data imputation [3] and has been proved to have relatively high accuracy [2,23]. Compared to the basic KNN imputation, OLAPKNN considers the structure complexity and the dependency constraints of the dimension hierarchies. Moreover, the dimensional data are usually qualitative on which we focus in this paper.

The remainder of this paper is organized as follows. In Section 2, we review the related work about data imputation algorithms. In Section 3, we formalize the DW dimension model. In Section 4, we propose a distance calculation method for dimension instances. In Section 5, we explain in detail our proposed dimension imputation algorithm. In Section 9, we validate our proposal by some experiments. In Section 7, we conclude this paper and hint at future research.

2 Related Work

There are various data imputation methods [16]: statistic based imputation, machine-learning based imputation, rule based imputation, external source based imputation and hybrid methods etc. The statistic based imputation completes the missing values by applying the statistical methods like filling average, the most frequent value or with the value of the most similar record; there are also methods using the regression to predict the missing values [19]. The machine learning based imputation methods use algorithms like k-nearest neighbor (KNN) [2,23,10,17], regression models [13], Naive Bayes [9] to predict the missing values. The rule based imputation methods [8,22,5] complete the missing values by some business rules, similarity rules or dependency rules. Concerning the external source based methods, the crowdsourcing [14] can be applied for the data imputation by putting forward the queries in the crowdsourcing frameworks and collecting answers to complete the missing data. There are also methods which

realize the imputation through web information [29,26] like web pages, web lists and web tables. What's more, there are hybrid methods which mix different imputation methods to provide a higher performance.

The statistic and machine learning based methods mainly focus on the numerical data, which fit for the imputation of facts where the data are mostly numerical. However, in the dimensions, there are mainly qualitative data which make it difficult to process the data imputation by such imputation methods. The rule based and external source based imputation methods may be suitable for the imputation of dimensions, but they need time and efforts to create rules or find the appropriate sources. Hence we propose H-OLAPKNN which combines the hierarchical imputation with a KNN-based imputation method.

3 DW Dimension

As a DW is composed of dimensions and facts and we focus on the dimension imputation, we introduce the DW dimension concepts used in this paper [20].

Definition 1 (Dimension). *In a data warehouse, a* ***dimension****, denoted by* D*, is defined as* (A^D, H^D, I^D)*.* $A^D = \{a_1, ..., a_u\} \cup \{id\}$ *is a set of attributes, where id represents the dimension's identifier;* $H^D = \{H_1, ..., H_v\}$ *is a set of hierarchies;* I^D *is a matrix of dimension instances, for a given row* r*, the row instance vector is denoted as* i_r*; for a given attribute* a_u*, their joint instance value is denoted as* i_{r,a_u}*.*

Definition 2 (Hierarchy). *A* ***hierarchy*** *of dimension* D*, denoted by* $H \in H^D$*, is defined as* $(Param^H, Weak^H)$*.* $Param^H =< id^D, p_2^H, ..., p_v^H >$ *is an ordered set of dimension attributes, called* ***parameters****, which set granularity levels along the dimensions,* $\forall k \in [1...v], p_k^H \in A^D$*. Parameter* p_1^H *rolls up to* p_2^H *in* H *is denoted as* $p_1^H \preceq_H p_2^H$*;* $Weak^H = Param^H \rightarrow 2^{(A^D - Param^H)}$ *is a mapping possibly associating each parameter with one or several* ***weak attributes****, which are also dimension attributes providing additional information; All parameters and weak attributes of* H *constitute the hierarchy attributes of* H*, denoted by* $A^H = Param^H \cup (\bigcup_{p_v^H \in Param^H} Weak^H[p_v^H])$

There exists different types of hierarchy, but the most basic and common one is the strict hierarchy [15] where a value at a hierarchy's lower-granularity belongs to only one higher-granularity value [24]. Thus in this paper, we only consider the case of the strict hierarchy.

4 Distance Between Dimension Instances

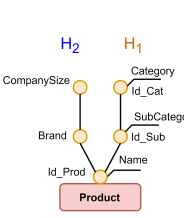
Since the KNN imputation select the k-nearest neighbors of the missing data instance for the imputation, we should calculate the distance between dimension instances containing missing data to be completed and other instances. In a

dimension D, for an instance $i_1 \in I^D$ containing missing data on a hierarchy $H_1 \in H^D$, and another instance $i_2 \in I^D$, we propose to calculate their distance by 4 levels:

- The **dimension instance distance** is the final distance between two instances i_1 and i_2, denoted by $\Delta(i_1, i_2)$. Since the attributes on the same hierarchy have their dependency relationships, we consider the attributes of a hierarchy as an entirety. $\Delta(i_1, i_2)$ is thus calculated by the weighted sum of the **hierarchy instance distances**.
- The **hierarchy instance distance** is the distance of the attributes of a hierarchy $H_2 \in H^D$ i.e. distance between $\{i_{1,a_1} \in i_1 : a_1 \in A^{H_2}\}$ and $\{i_{2,a_1} \in i_2 : a_1 \in A^{H_2}\}$, denoted by $\Delta_{H_2}(i_1, i_2)$. It is calculated by the weighted sum of the **hierarchy level instance distances**. The lowest-granularity level of each hierarchy is the same i.e. id with its weak attributes, so we consider the hierarchy instance distance from the second level of the hierarchy and we regard each weak attribute of id as a hierarchy containing only one parameter.
- The **hierarchy level instance distance** is the instance distance between the attributes of a level l on a hierarchy H_2 i.e. distance between $\{i_{1,a_2} \in i_1 : a_2 \in p_l^{H_2} \cup Weak^{H_2}[p_l^{H_2}]\}$ and $\{i_{2,a_2} \in i_2 : a_2 \in p_l^{H_2} \cup Weak^{H_2}[p_l^{H_2}]\}$, denoted by $\Delta_{p_l^{H_2}}(i_1, i_2)$. It is calculated by the average of the instance distances of the level's parameter and weak attributes (**attribute distances**).
- The **attribute distance** is the instance distance of an individual attribute $a_u \in A^D$ i.e. distance between i_{1,a_u} and i_{2,a_u}, denoted by $\Delta(i_{1,a_u}, i_{2,a_u})$.

Based on the explanation of the distances, we then give the formulas and some examples to illustrate them in detail.

Example 1. Given a dimension *Product* containing two hierarchies H_1 and H_2 whose schema and instances are shown in Fig. 1. Instance i_1 contains missing values on H_1, Fig. 2 shows the calculation of the distance $\Delta(i_1, i_2)$ between i_1 and another instance i_2.



(a) Schema

	H2				H1			
Instance	Brand	CompanySize	Id	Name	Id_Sub	Subcategory	Id_Cat	Category
i_1	Apple	Large	p1	Iphone 13	?	?	Tn	Technology
i_2	Samsung	Large	p2	Gaxaly S10	Ph	Phones	Tn	Technology
i_3	Homefine	Medium	p3	Office Table	Tb	Table	Of	Office
i_4	Homefine	Medium	p4	Computer Table	Tb	Table	Of	Office
i_5	Sumsang	Large	p5	Galaxy Book	Pc	Laptop	Tn	Technology

(b) Instances

Fig. 1: Schema and instances of dimension *Product*

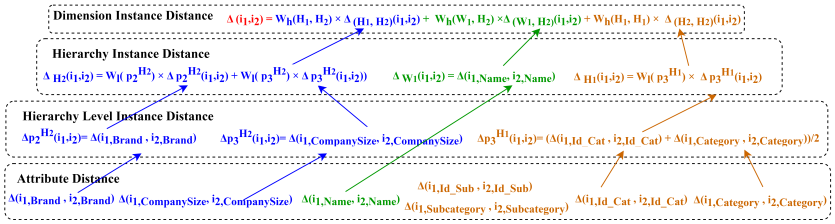


Fig. 2: Distance between i_1 and i_2

4.1 Attribute Distance

There are different attribute data types which can be mainly classified into numerical and textual. For numerical data, we use normalized distance of numerical data [1]. For textual data, we first apply semantic distance e.g. cosine distance based on word2vec [11]. If the attribute value cannot be found in the model, we can then use the syntactic distance e.g. normalized Levenshtein Distance [28].

For an attribute a_{u1}, if $i_{1,a_{u1}}$ is missing, then $\Delta(i_{1,a_{u1}}, i_{2,a_{u1}})$ cannot be calculated and is not taken into count for the distance calculation. For an attribute a_{u2}, if $i_{2,a_{u2}}$ is missing, then $\Delta(i_{1,a_{u2}}, i_{2,a_{u2}})$ is obtained by the average distance between $i_{1,a_{u2}}$ and other instances whose value of a_{u2} is not missing.

Example 2. Following the calculation rules of the attribute distance, we obtain $\Delta(i_{1,Brand}, i_{2,Brand}) = 0.71$, $\Delta(i_{1,CompanySize}, i_{2,CompanySize}) = 0$, $\Delta(i_{1,Name}, i_{2,Name}) = 0.8$, $\Delta(i_{1,Id\_Cat}, i_{2,Id\_Cat}) = 0$, $\Delta(i_{1,Category}, i_{2,Category}) = 0$. Since i_{1,Id_Sub} and $i_{1,Subcategory}$ are missing, $\Delta(i_{1,Id\_Sub}, i_{2,Id\_Sub})$ and $\Delta(i_{1,Subcategory}, i_{2,Subcategory})$ cannot be calculated and are not taken into count for the calculation of $\Delta(i_1, i_2)$.

4.2 Hierarchy Level Instance Distance

The hierarchy level instance distance $\Delta_{p_l^{H_2}}(i_1, i_2)$ is calculated as (1).

$$\Delta_{p_l^{H_2}}(i_1, i_2) = \frac{\Delta(i_{1,p_l^{H_2}}, i_{2,p_l^{H_2}}) + \sum_{w \in Weak[p_l^{H_2}]} \Delta(i_{1,w}, i_{2,w})}{1 + |Weak[p_l^{H_2}]|} \tag{1}$$

As we mentioned that we only consider the levels from the second level of each hierarchy, we do not calculate the distance for the first level of hierarchies. The weak attributes of the first hierarchy levels are regarded as hierarchies containing only one parameter, so their level distance is not needed to be calculated neither.

Example 3. According to (1), for the levels in H_1, we have $\Delta_{p_3}^{H_1}(i_1, i_2) = (0 + 0)/2 = 0$. As the parameter and weak attribute value of the second level i_{1,Id_Sub} and $i_{1,Subcategory}$ are missing, the distance of this level is not taken into account. For H_2, since the two levels contain only one parameter without weak attribute, their hierarchy level is equal to the attribute distance of the parameter, so we have $\Delta_{p_2}^{H_2}(i_1, i_2) = 0.71$, $\Delta_{p_3}^{H_2}(i_1, i_2) = 0$.

4.3 Hierarchy Instance Distance

The hierarchy instance distance is calculated as (2), where $W_l(p_l^{H_2})$ is the hierarchy level weight.

$$\Delta_{H_2}(i_1, i_2) = \sum_{p_l^{H_2} \in H_2 \setminus \{id\}} W_l(p_l^{H_2}) \Delta_{p_l^{H_2}}(i_1, i_2) \tag{2}$$

For a weak attribute $w \in Weak^{H_2}[id]$ of the first hierarchy level, $\Delta_w(i1, i_2) = \Delta(i_{1,w}, i_{2,w})$.

Hierarchy Level Weight Since the parameters on the lower levels have thinner granularity, their weight for measuring the hierarchy instance distance should be higher. Here, we propose two hierarchy level weights: one is based on the cardinalities of the parameters and another is an incremental weight.

- For the cardianlity-based weight, we consider the number of the distinct values of the level as the portion of the weight. Thus for the cardianlity-based hierarchy level weight of the lth level at H_2 is calculated as (3), where $dv(n)$ denotes the number of distinct values of the nth level.

$$W_l^c(p_l^{H_2}) = \frac{dv(l)}{\sum_{j=2}^{|Param^{H_2}|} dv(j)} \tag{3}$$

- However, when the cardinality ratio between certain parameters is very large, the cardinality-based weight may be biased. So we also propose another type of incremental hierarchy level distance weight. For the incremental weight, we consider the weight of the highest-granularity as one portion and it increases by one portion for each neighboring lower-granularity level. The total weight should be equal to 1, thus the incremental hierarchy level weight of the lth level at H_2 is calculated as (4).

$$W_l^i(p_l^{H_2}) = \frac{2(|Param^{H_2}| - l + 1)}{|Param^{H_2}|^2 - |Param^{H_2}|} \tag{4}$$

Example 4. Our example has only 5 instances, so we can use cardinality-based weight to get hierachy level weight. We thus have for H_1: $W_l(p_2^{H_1}) = 3/(3 + 2) = 0.6$ and $W_l(p_3^{H_1}) = 2/(3 + 2) = 0.4$. For H_2: $W_l(p_2^{H_2}) = 3/(3 + 2) = 0.6$ and $W_l(p_3^{H_2}) = 2/(3 + 2) = 0.4$. We can then calculate the hierarchy instance distances: $\Delta_{H_1}(i_1, i_2) = 0.4 \times 0 = 0$, $\Delta_{H_2}(i_1, i_2) = 0.6 \times 0.71 + 0.4 \times 0 = 0.426$, $\Delta_{w_1}(i_1, i_2) = 0.8$.

4.4 Dimension Instance Weight

The dimension instance weight $\Delta(i_1, i_2)$ is calculated as (5), where $W_h(H_1, H_2)$ and $W_h(H_1, w)$ are hierarchy weights of H_2 and w with respect to H_1.

$$\Delta(i_1, i_2) = \sum_{H_2 \in H^D} W_h(H_1, H_2)\Delta_{H_2}(i_1, i_2) + \sum_{w \in Weak^{H_2}[id]} W_h(H_1, w)\Delta_w(i_1, i_2) \tag{5}$$

Hierarchy Weight The dependency degree in the rough set theory [18] measures the degree of the dependency between attributes, so it is applied for the hierarchy weight.

When calculating the hierarchy distance weight, we can consider a decision system $S = (I^D, A_n^{H_2}, A_n^{H_1})$, since we do not take the first level of a hierarchy into account, $A_n^{H_1} = A^{H_1} \setminus (\{id\} \cup Weak^{H_1}[id])$, $A_n^{H_2} = A^{H_2} \setminus (\{id\} \cup Weak^{H_2}[id])$. The second hierarchy level parameters $p_2^{H_1}, p_2^{H_2}$ determine all the other hierarchy attributes in $A_n^{H_1}$ and $A_n^{H_2}$, we can reduce the attribute sets of $A_n^{H_1}$ and $A_n^{H_2}$ to the sets containing only the values of the second hierarchy level parameter $p_2^{H_1}, p_2^{H_2}$. According to [18], the degree k to which H_1 depends on H_2, denoted $H_2 \Rightarrow_k H_1$ is thus defined as:

$$k = \gamma(A_n^{H_2}, A_n^{H_1}) = \gamma(p_2^{H_2}, p_2^{H_1}) = \frac{card(POS_{p_2^{H_2}}(p_2^{H_1}))}{card(I^D)} \tag{6}$$

where $POS_{p_2^{H_2}}(p_2^{H_1}) = \bigcup_{X \in I^D/p_2^{H_1}} p_{2*}^{H_2}(X)$ and $card(X)$ is the cardinality of an non-empty set X, the missing second level parameter values are not taken into account. For H_1 itself, we have $\gamma(A_n^{H_1}, A_n^{H_1}) = 1$.

The hierarchy distance weight of H_2 with respect to H_1 is the ratio of their dependency degree with respect to the sum of the dependency degrees of the all hierarchies and first level weak attributes in D with respect to H_1 as (7).

$$W_h(H_1, H_2) = \frac{\gamma(A_n^{H_2}, A_n^{H_1})}{\sum\limits_{H_3 \in H^D} \gamma(A_n^{H_3}, A_n^{H_1}) + \sum\limits_{w \in Weak^{H_1}[id]} \gamma(w, A_n^{H_1})} \tag{7}$$

Example 5. In our example, we have $card(I^D) = 5$, $card(POS_{p_2^{H_2}}(p_2^{H_1})) = 2$, so $\gamma(A_n^{H_2}, A_n^{H_1}) = 2/5 = 0.4$. In the same way, we can get $\gamma(w_1, A_n^{H_1}) = 2/5 = 0.4$, we also have $\gamma(A_n^{H_1}, A_n^{H_1}) = 1$. We can thus get the hierarchy weights: $W_h(H_1, H_2) = 0.4/(0.4+0.4+1) = 0.22$, $W_h(H_1, H_1) = 1/(0.4+0.4+1) = 0.56$ and $W_h(w_1, H_2) = 0.4/(0.4+0.4+1) = 0.22$. We can finally obtain the dimension instance distance $\Delta(i_1, i_2) = 0.22 \times 0.46 + 0.22 \times 0.8 + 0.56 \times 0 = 0.28$

5 H-OLAPKNN Imputation

5.1 H-OLAPKNN Overview

The H-OLAPKNN imputation is shown in Algo. 1. It is composed of three steps where the first is the hierarchical imputation and the next two steps concern the OLAPKNN imputation.

1. The hierarchical imputation is based on the functional dependencies of the hierarchy attributes. It searches for an instance having the same value on a lower-granularity level parameter of the missing value and whose attribute of the missing value is not empty, we can then replace the missing value with this non-empty value ($line_1$).
2. The weak attributes' values are determined by their parameters' values, so we complete the parameters before completing their weak attributes. Thus then, for missing data of each hierarchy ($line_2$), we create candidate lists of the instances containing possible replaced values and select the k nearest neighbors in the candidate lists to complete the missing data ($line_3$).
3. There are weak attributes which can be completed together with their parameter. Finally for the remaining missing weak attribute data, they are completed in the similar way ($line_4$).

Next, we explain in detail the OLAPKNN imputation algorithm. A weak attribute of a hierarchy can be regarded as a "highest level parameter" of a part of the hierarchy whose imputation is similar to the parameter imputation. So we only explain the parameter imputation in this paper.

5.2 Imputation for Parameters by OLAPKNN

Parameter Imputation Order We first introduce the continuous missing parameter group in order to explain the imputation order for parameters.

Definition 3 (Continuous missing parameter group). *For an instance $i_r \in I^D$ in the dimension D containing missing values on parameters of a hierarchy H, all these parameters are in a set $Pm_r^H = \{p_v^H \in Param^H : i_{r,p_v^H}$ is empty$\}$. For the parameters in Pm_r^H, they can be divided into one or several continuous missing parameter groups. A **continuous missing parameter group (CG)** contains one or several parameters which are neighbors on H and are maximal neighbors in Pm_r^H. By neighbors on H, we mean that for the parameter p_{lowest} having the lowest-granularity level in the CG on H and the one $p_{highest}$ having the highest-granularity level, if there exists any parameter $p_{middle} \in Param^H$, such that $p_{lowest} \preceq_H p_{middle} \preceq_H p_{highest}$, then $p_{middle} \in Pm_r^H$; By maximal neighbors in Pm_r^H, we mean that if there exists any parameter $p_{low_2} \in Param^H$, such that $p_{low_2} \preceq_H p_{lowest}$, then $p_{low_2} \notin Pm_r^H$, if there exists any parameter $p_{high_2} \in Param^H$, such that $p_{highest} \preceq_H p_{high_2}$, then $p_{high_2} \notin Pm_r^H$. We call all CGs of a hierarchy H containing a same number of parameters a n-CGs of H, where n denotes the number of parameters.*

Algo. 2 shows the imputation of the parameters. For a given hierarchy H on a dimension D, we carry out the imputation for parameters in the n-CGs by the ascending order of n ($line_1$). We can thus make sure that all the $(n-1)$-CGs instances are completed so that we can carry out the imputation for the n-CGs based on the existing data. Then for each n-CGs, we look at all possible CG combinations ($line_{2-3}$). Next we verify if there are instances containing missing values for each possible CG ($line_{4-9}$). According to $Definition$ 3, the instances

of a CG on H have missing values on all parameters of the group. If there is a neighboring lower-granularity or higher-granularity parameter of the group, the instances do not have missing value on them ($line_9$).

Algorithm 1: $H - OLAPKNN(D)$

```
1 hierarchicalImputation(D);
2 for H ∈ H^D do
3 |   imputeParam(D, H) ;
4 |   imputeWeak(D, H) ;
```

Algorithm 2: $imputeParam(D, H)$

```
1  for ncontinuous ← 1 to |Param^H| − 1 do
2  |   for i ← 1 to |Param^H| − ncontinuous do
3  |   |   P_CG ← Param^H[i : i + ncontinuous − 1] ;
4  |   |   p_low, p_high ← ∅ ;
5  |   |   if i > 1 then
6  |   |   |   p_low ← Param[i − 1] ;
7  |   |   if i < |Param^H| − ncontinuous then
8  |   |   |   p_high ← Param[i + ncontinuous] ;
9  |   |   I_missing = {i_r ∈ I^D : (∀p_cg ∈ P_CG, i_r,p_cg =
   |   |   |   null) ∧ (∃p_low ⟹ i_r,p_low ≠
   |   |   |   null) ∧ (∃p_high ⟹ i_r,p_high ≠ null)} ;
10 |   |   lowMap ← Map ;
11 |   |   for i_m ∈ I_missing do
12 |   |   |   I_candidate ← getCandidateList(D, P_CG,
   |   |   |   |   p_high, i_m, 1) ;
13 |   |   |   vWeightMap ← getVWeightMap(D, i_m,
   |   |   |   |   I_candidate, k, P_CG) ;
14 |   |   |   lowMap ← replaceNoPlow(D, H,
   |   |   |   |   lowMap, vWeightMap, i_m, P_CG, p_low) ;
15 |   |   if ∃p_low then
16 |   |   |   replacePlow(lowMap, P_CG, H, D, p_low) ;
```

Candidate List Since some missing data are already completed by the hierarchical imputation, for the remaining missing data, they can no longer be completed with the aid of their lower-granularity parameters. For a value of one parent parameter, there may be several possible values on a child parameter of its. So for a missing data instance of a CG, we can find all possible replaced values based on their neighboring higher parameter and create a candidate list (Algo. 2 $line_{11}$). The candidate list contains not only the candidate replaced values of CG attributes but also the values of all other attributes of the dimension because we need all attribute's value for the calculation of the distances.

Algo. 3 shows the candidate list creation for an instance of a CG. If the neighboring higher-granularity parameter p_{high} of the CG exists, we search for all the instances having the same values on p_{high} as the CG instance, and containing non-missing values on the CG parameters. Then these instances can be added into the candidate list ($line_{1-3}$). If there does not exist a neighboring higher parameter for a CG, we add all the instances of the dimension which contain non-missing values on the CG parameters into the candidate list ($line_{4-5}$).

Creation of Replaced Value Weight Map For the CG instance, we can get a map for each possible replaced values in the nearest neighbors with their distance-based weight for the selection of the final replaced value as described in Algo. 4. We first create a map of each instance in the candidate list with its distance with respect to the missing instance ($line_{1-3}$). Then we can select the k nearest candidate instances to create a candidate list if the candidate list contains more than k instances, if not, we can keep all candidate instances ($line_{4-5}$). The selected candidate instances may contain same replaced values,

so we create a map of each replaced values with their weight ($line_6$). According to [7], for an instance i_m of a CG, for a selected candidate list containing k instances, the distance weight of the n nearest instance i_{cn} can be calculated as (8), where i_{ck} denotes the kth nearest instance and i_{c1} denotes the nearest instance. It is to be noted that $W_d(i_m, i_c) = 1$ when $\Delta(i_m, i_{ck}) = \Delta(i_m, i_{c1})$.

$$W_d(i_m, i_c) = \frac{\Delta(i_m, i_{ck}) - \Delta(i_m, i_{cn})}{\Delta(i_m, i_{ck}) - \Delta(i_m, i_{c1})} \tag{8}$$

Thus the weight of a candidate of replaced values is the sum of the weight of the instances which contain them ($line_{4-5}$).

Algorithm 3: $getCanList(D, P_{CG}, p_{high}, i_m, parameter)$

1 **if** $parameter = 1$ **then**
2 **if** $\exists p_{high}$ **then**
3 $I_{candidate} \leftarrow \{i_r \in I^D : (\exists p_{cg} \in P_{CG}, i_{r,p_{cg}} \neq null) \wedge (i_{r,p_{high}} = i_{r_{missing},p_{high}})\}$;
4 **else**
5 $I_{candidate} \leftarrow \{i_r \in I^D : (\exists p_{cg} \in P_{CG}, i_{r,p_{cg}} \neq null)\}$;
6 **else**
7 $I_{candidate} \leftarrow \{i_r \in I^D : (i_{r,weak} \neq null)\}$;
8 **return** $I_{candidate}$

Algorithm 4: $getVWeightMap(D, i_m, I_{candidate}, k, P_{CG})$

1 $iDistanceMap \leftarrow Map$;
2 **for** $i_c \in I_{candidate}$ **do**
3 $iDistanceMap[i_{c,id}] \leftarrow \Delta(i_m, i_c)$;
4 **if** $|I_{candidate}| > k$ **then**
5 $iDistanceMap \leftarrow iDistanceMap.top(k)$;
6 $vWeightMap \leftarrow Map$;
7 **for** $i_{c,id} \in iDistanceMap.keys()$ **do**
  /* addMap(Map, key, value): Create the map if it does not exist. Add the value to the existing value if the key exists, assign the value to the key if not. */
8 $addMap(vWeightMap, \{i_{c,p_{cg}} : i_{c,p_{cg}} \in i_{c,P_{CG}}\}, Wv(i_m, i_c))$;
9 **return** $vWeightMap$

Replacement of Values To fill in the missing values of CG, we have two cases: the first case (Algo. 2 $line_{13}$) is that there is no lower non-id parameter of the missing parameter group, the second case (Algo. 2 $line_{14-15}$) is that there is such parameter. The difference is that for the second case, we have to take the strictness of hierarchy into account by making sure that a lower parameter value of the CG has only one higher-granularity level parameter after the imputation.

The replacement of the values of the first case is described in Algo. 5. We can take the values having the highest weight in the weight map ($line_1$) to fill in the missing values of the CG ($line_{2-3}$).

The replacement of the values for the second case is described in Algo. 5 and Algo. 6. We create a map $lowMap$ for each neighboring lower-granularity parameter value which corresponds to another map of the each possible replaced value and its total weight (Algo. 2 $line_{10}$). For each instance of the CG, we get the replaced values with the highest value weight (Algo. 5 $line_1$). For the value of its neighboring lower-granularity parameter, we update the replaced values and the weight (Algo. 5 $line_{8-10}$). When all the missing instances of a CG are treated, we get a final $lowMap$. For each value of the neighboring lower-granularity level parameter in $lowMap$, we can take the replaced values with the highest weight to fill in the missing values (Algo. 6 $line_{1-5}$).

Algorithm 5: $replaceNoPlow(D, H, lowMap, vWeightMap, i_m, P_{CG}, p_{low})$

```
i_{replace,P_CG} ← vWeightMap.top(1).key() ;
if ∄p_low then
    i_{m,P_CG} ← i_{replace,P_CG} ;
    for p_cg ∈ P_CG do
        for w^{p_cg} ∈ Weak^H[p_cg] do
            if i_{m,w^{p_cg}} = ∅ then
                i_{m,w^{p_cg}} ← {i_{r,w^{p_cg}} ∈ I^D : i_{r,p_cg} =
                  i_{m,p_cg}}.getOne() ;
else
    addMap(lowMap[i_{m,p_low}], i_{replace,P_CG},
      vWeightMap[i_{replace,P_CG}]) ;
    addMap(lowMap, i_{m,p_low}, lowMap[i_{m,p_low}]) ;
return lowMap
```

Algorithm 6: $replacePlow(lowMap, P_{CG}, H, D, p_{low})$

```
for i_{m,p_low} ∈ lowMap.keys() do
    vWeightMap ← lowMap[i_{m,p_low}].top(1) ;
    i_{replace,p_low} ← vWeightMap.key() ;
    for i_m ∈ {i_r ∈ I^D : i_{r,p_low} = i_{m,p_low}} do
        i_{m,P_CG} ← i_{replace,P_CG} ;
        for p_cg ∈ P_CG do
            for w^{p_cg} ∈ Weak^H[p_cg] do
                if i_{m,w^{p_cg}} = ∅ then
                    i_{m,w^{p_cg}} ← {i_{r,w^{p_cg}} ∈ I^D : i_{r,p_cg} =
                      i_{m,p_cg}}.getOne() ;
```

6 Experimental assessment

6.1 Technical environment and datasets

To evaluate the effectiveness and efficiency of H-OLAPKNN, we implement our algorithms and conduct experiments with different datasets and compare it to other imputation methods. Our code is developed in Python 3.7 and is executed on a Intel(R) Core(TM) i5-10210U 1.60 GHz CPU with a 16 GB RAM. Data are integrated in R-OLAP format with Oracle 11g. The distance metrics that we use are like described in section 4.1, for the word embedding based distance, we use Google's pre-trained word2vec model [3].

We employ 3 real world datasets. The first dataset is a regional sale dataset (**RegionalSales**) storing sales data for a company across US regions. The second (**IBRD**) and the third (**MIGA**) ones are data of world bank which are respectively the International Bank for Reconstruction and Development balance sheet data and the Multilateral Investment Guarantee Agency issued projects data. For each one of the datasets, we create a DW for our experiment. The link of the dataset source and more information can be found in our github[4].

6.2 Experimental methodology

We apply different missing rates (1%, 5%, 10%, 20%, 30%, 40%) for each attribute except for the primary keys. To generate missing data for an attribute, we sort randomly all the instances and remove attribute values of the first certain percentage of instances. For the effectiveness, since we focus on the qualitative data, we apply the accuracy (number of correctly replaced values divided by number of missing values) as the metric instead of the root mean square error (RMSE) [16] which is widely used but is only suitable for quantitative data imputation. For the efficiency, we get the run time of each algorithm. We carry out 20 tests for each dataset and get the average accuracy and run time.

[3] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?resourcekey=0-wjGZdNAUop6WykTtMip30g

[4] https://github.com/Implementation111/H-OLAPKNN/

We compare our proposed method with some other methods in the literature as baseline. **H-OLAPKNN(MI)**: Since the mutual information is widely used in the KNN based data imputation [10,17]. In this baseline, we apply our proposed OLAPKNN algorithm by using the mutual information instead of the dependency degree as the hierarchy distance weight. **KNN** [6]: This method use the basic KNN algorithm to generate the replaced values for missing data **Mode**: The Mode method simply replace the missing data with the most frequent non-empty value of the attribute in the table.

6.3 Results and analysis

For each dataset, the optimal k of KNN is different. So we test with different k values between 1 and 20 and choose the best one for the experiment of each data set which are respectively 5, 4 and 8. For the W_l, we choose the weight with a better result as the weight for each dataset which are respectively W_l^i, W_l^c and W_l^i. Then we compare the accuracy and the run time of each algorithm.

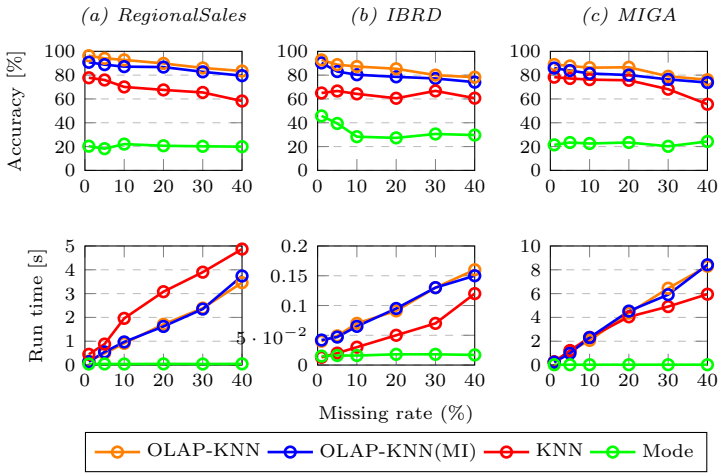


Fig. 3: Results of experiments

Accuracy The accuracy result is shown as Fig. 3. We can see that the proposed H-OLAPKNN algorithm outperforms all the other baseline algorithms for each dataset. The Mode method has the worst result since it is too simple and it takes nothing into account. Compared to the mutual information, we observe that using the dependency degree as the hierarchy distance weight can help us get a more accurate result as it considers the ordered dependency instead of the inter-dependency. Compared to the basic KNN method, the H-OLAPKNN returns a better accuracy results since it considers the structure of the DW

and take the dependencies among the attributes into account. The accuracy of H-OLAPKNN, H-OLAPKNN(MI) and KNN decreases with the increase of the missing rate because when there are more missing data, 1) we have less complete data for getting more precise distance scores and 2) it is more likely that the proper replaced data do not exist in the table.

Run time The run time result is shown as Fig 3. The Mode algorithm costs less time since it is the simplest method. The run time of the other three algorithms changes linearly with respect to the missing rate. There is no big difference between H-OLAPKNN and H-OLAPKNN(MI) since they are only different in terms of the calculation of hierarchy distance weight. The OLAPKNN costs less time than KNN for dataset **RegionalSales**, but more time for the other two datasets. This is because the hierarchical imputation complete most of the data of **RegionalSales** so that it takes less time for OLAPKNN to create the candidate list and compare the similarities.

7 Conclusion and future work

In this paper, we propose a DW dimensional data imputation method by combining hierarchical imputation with a novel KNN-based algorithm. We first define a 4-level distance calculation method for dimension instances by taking advantage of the DW dimension structure. Then by applying the proposed distances, we define the KNN-based algorithm. The advantage of the proposed algorithm is that it takes the dimension structure complexity into account and is able to make replaced values conform to the dependency constraints of the hierarchies. Our proposal is validated by a series of experiments and is proved to outperform other baselines in the literature. It increases the dimension data imputation accuracy by up to 25.2% with respect to the basic KNN imputation.

In the future, we will extend our method for the imputation of numerical data in the dimensions and facts. We also intend to generalize the method for non-DW data by proposing an approach automatically modelling them in OLAP.

References

1. Preface. In: Han, J., Kamber, M., Pei, J. (eds.) Data Mining. third edn. (2012)
2. Nearest neighbor imputation using spatial–temporal correlations in wireless sensor networks. Information Fusion **15**, 64–79 (2014)
3. Beretta, L., Santaniello, A.: Nearest neighbor imputation algorithms: a critical evaluation. BMC medical informatics and decision making **16**(3), 197–208 (2016)
4. Bimonte, S., Ren, L., Koueya, N.: A linear programming-based framework for handling missing data in multi-granular data warehouses. Data & Knowledge Engineering **128** (2020)
5. Breve, B., Caruccio, L., Deufemia, V., Polese, G.: Renuver: A missing value imputation algorithm based on relaxed functional dependencies. In: EDBT. pp. 1–52 (2022)

6. Domeniconi, C., Yan, B.: Nearest neighbor ensemble. In: ICPR. vol. 1 (2004)
7. Dudani, S.A.: The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics (4), 325–327 (1976)
8. Fan, W., Jianzhong, L., Shuai, M., Nan, T., Wenyuan, Y.: Towards certain fixes with editing rules and master data. In: The VLDB Journal. pp. 173–184 (2010)
9. Farhangfar, A., Kurgan, L.A., Pedrycz, W.: A novel framework for imputation of missing values in databases. IEEE SMC **37**(5), 692–709 (2007)
10. García-Laencina, P.J., Sancho-Gómez, J.L., Figueiras-Vidal, A.R., Verleysen, M.: K nearest neighbours with mutual information for simultaneous classification and missing data imputation. Neurocomputing **72**(7), 1483–1493 (2009)
11. Jatnika, D., Bijaksana, M.A., Suryani, A.A.: Word2vec model analysis for semantic similarities in english words. Procedia Computer Science **157**, 160–167 (2019)
12. Li, D., Deogun, J., Spaulding, W., Shuart, B.: Towards missing data imputation: A study of fuzzy k-means clustering method. In: IJCRS. pp. 573–579 (2004)
13. Little, R., Rubin, D.: Statistical analysis with missing data. Wiley (2002)
14. Lofi, C., El Maarry, K., Balke, W.T.: Skyline queries over incomplete data-error models for focused crowd-sourcing. In: Conceptual Modeling. pp. 298–312 (2013)
15. Malinowski, E., Zimányi, E.: Olap hierarchies: A conceptual perspective. In: Advanced Information Systems Engineering. pp. 477–491 (2004)
16. Miao, X., Gao, Y., Guo, S., Liu, W.: Incomplete data management: a survey. Frontiers of Computer Science **12**, 4–25 (2018)
17. Pan, R., Yang, T., Cao, J., Lu, K., Zhang, Z.: Missing data imputation by K nearest neighbours based on grey relational structure and mutual information. Applied Intelligence **43**(3), 614–632 (2015)
18. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences **177**(1), 3–27 (2007)
19. Pedro J., G.L., José-Luis, S.G., Aníbal R., F.V.: Pattern classification with missing data: a review. Neural Computing and Applications **19**(2), 263–282 (2010)
20. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for olap manipulations. Inter. J. of Data Warehousing and Mining **4**, 17–46 (2008)
21. de S. Ribeiro, L., Goldschmidt, R.R., Cavalcanti, M.C.: Complementing data in the etl process. In: Cuzzocrea, A., Dayal, U. (eds.) DaWak. pp. 112–123 (2011)
22. Song, S., Zhang, A., Chen, L., Wang, J.: Enriching data imputation with extensive similarity neighbors **8**(11), 1286–1297 (2015)
23. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for DNA microarrays. Bioinformatics **17**(6), 520–525 (2001)
24. Trujillo, J., Palomar, M., Gomez, J., Song, I.Y.: Designing data warehouses with oo conceptual models. Computer **34**(12), 66–75 (2001)
25. Wu, X., Barbará, D.: Modeling and imputation of large incomplete multidimensional datasets. In: DaWak. pp. 286–295 (2002)
26. Yakout, M., Ganjam, K., Chakrabarti, K., Chaudhuri, S.: Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In: ACM SIGMOD. p. 97–108 (2012)
27. Yang, Y., Abdelhédi, F., Darmont, J., Ravat, F., Teste, O.: Internal data imputation in data warehouse dimensions. In: DEXA. pp. 237–244 (2021)
28. Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(6), 1091–1095 (2007)
29. Zhixu, L., Sharaf, M.A., Sitbon, L., Sadiq, S., Indulska, M., Zhou, X.: A web based approach to data imputation. World Wide Web **17**(5), 873–897 (2014)