Keys as Features for Graph Entity Matching

Ting Deng[1,2] Lei Hou[1,2] Ziyan Han[1,2]
[1]*Beijing Advanced Institution on Big Data and Brain Computing, Beihang University, Beijing, China*
[2]*SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China*
{dengting, houlei, hanzy}@act.buaa.edu.cn

***Abstract*—Keys for graphs aim to uniquely identify entities represented by vertices in a graph, using the combination of topological constraints and value equality constraints. This paper proposes graph matching keys, referred to as GMKs, an extension of graph keys with similarity predicates on values, supporting approximation entity matching. We treat entity matching as a classification problem, and propose GMKSLEM, a supervised learning method for graph entity matching. In GMKSLEM, a feature extraction method is provided to discover candidate GMKs (CGMKs) to construct features for vector representation, and then high-quality features and representations are generated by feature selection. Moreover, GMKSLEM provides support to explain the classification results. Using real-life data, we experimentally verify the effectiveness of GMKSLEM, as well as its interpretability.**

I. Introduction

Keys are fundamental integrity constraints in data management to define attributes for identifying entities. They have also been extensively studied for XML, and recently further defined for graphs.

Keys for graph (GKeys) [1] are designed as a combination of topological constraints (by using graph patterns) and value constraints to identify entities, with node identity to reflect the recursive property in collective entity matching [2]. They are further extended to ontological graph keys (OGKs) [3], graph entity dependencies (GEDs) [4] and graph differential dependencies (GDDs), by supporting ontological similarity on labels (*i.e.,* node types), the unified expression of functional dependencies, and the similarity on attribute values and properties, respectively. These constraints have been applied to entity matching [1], [3], [5] and error fixing combining data repairing and object identification [6], on graphs, where a set of high-quality constraints need to be designed beforehand by domain experts or discovered automatically. Relying on domain experts is often unrealistic, which motivates a lot of study on automatic discovery algorithms for dependencies, while it is also not an easy job (see [7] for a review). Automatic discovery of keys is even harder since it needs to find both meaningful graph patterns and value constraints [8], [9]. Moreover, for keys with attribute similarity, it is challenging to define similarity thresholds [5].

The other line of studies on entity matching is machine learning (ML)-based methods, which provide state-of-the-art results for EM on web data [10]–[12]. However, one of the problems of ML-based methods is the lack of explanation of why two entities match.

In this paper, we aim to combine the advantages of the ML-based and constraint-based methods for graph entity matching. We want to develop a ML-based method to get high accuracy on identifying entities, and provide interpretability using keys.

We propose graph matching keys (GMKs), which extend GKeys [1], [4] with similarity predicates on attributes. A GMK is a combination of a graph pattern as a topological constraint and an attribute dependency with similarity predicates, supporting collective entity matching using node identities. We interpret GMKs with a single homomorphic match of pattern in [4], instead of three isomorphic mappings in [1]. This makes GMKs have clearer semantics than GKeys.

We present GMKSLEM, a supervised learning method for graph entity matching, in which entity matching is modeled as a classification problem. GMKSLEM uses GMKs to construct features, for generating vector representation for node pairs in a labeled dataset. High-quality features are achieved by a two-step hybrid feature selection method, including feature ranking and feature filtering, discarding those with low contributions in classification. To the best of our knowledge, GMKSLEM is a first attempt to combine supervised learning method with graph dependencies in entity matching. By using the classifiers generated during the process of feature filtering, we provide an approach to explaining why two entities match using the features that make contributions when they are identified for the first time.

Using real-life graphs, we experimentally verify the effectiveness of GMKSLEM. We also provide a study case to show how GMKs work for explaining the matching results.

II. Graph Matching Keys

In this section, we define the graph matching keys (GMKs). We first review the notions of graphs and graph patterns. Assume two sets Γ and Θ of labels and attributes, respectively.

Graphs and graph patterns. A *graph* G is specified as (V, E, L, F), where V is a finite set of nodes, $E \subseteq V \times V$ is the set of directed edges, L is a labeling function, assigning each node v and edge e with $L(v)$ and $L(e)$ in Γ, respectively, and for each node $v \in V$, $F(v)$ is a tuple $(A_1 = a_1, \ldots, A_n = a_n)$ where A_i is an attribute of v and a_i is a constant, written as $v.A_i = a_i$, and $A_i \neq A_j$ if $i \neq j$. In particular, each v has a special attribute id denoting its *node identity*.

A *graph pattern* is a graph $Q[\bar{x}] = (V_Q, E_Q, L_Q)$, where V_Q (resp. E_Q) is a set of pattern nodes (resp. edges), L_Q is a function that assigns a label $L_Q(u)$ (resp. $L_Q(e)$) to each node $u \in V_Q$ (resp. edge $e \in E_Q$), and $\bar{x}$ is a list of variables,

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00217

one for each node in V_Q. We allow wildcard '_' as a special label in Q.

Pattern matching. We say that a label ι matches ι', denoted by $\iota \asymp \iota'$, if either (a) ι and ι' are in Γ and $\iota = \iota'$, or (b) $\iota' \in \Gamma$ and ι is '_' , *i.e.,* wildcard matches any label in Γ.

A *match* of pattern Q in graph G is a *homomorphism* h from Q to G such that (a) for each node $u \in V_Q$, $L_Q(u) \asymp L(h(u))$ and (b) for each edge $e = (u, u')$ in Q, $e' = (h(u), h(u'))$ is an edge in G where $L_Q(e) \asymp L(e')$. We also denote the match as a vector $h(\bar{x})$ if it is clear from the context, consisting of $h(x)$ for all variables $x \in \bar{x}$, in the same order as $\bar{x}$, and refer to $Q(G)$ as the set of all matches of Q in G.

We use $Q(u, u', G)$ to denote the set of all matches of pair (u, u') in $Q(G)$ for $u, u' \in Q$, *i.e., $Q(u, u', G)$* consists of pairs of nodes (v, v') in G such that there exists a match h of Q in G under which $h(u) = v$ and $h(u') = v'$.

Graph Matching Keys. GMKs extend GKeys in [4] with similarity predicates on attributes, which adopt the semantics of a single homomorphic match instead of three isomorphic mappings in [1].

Syntax. A GMK ψ is a pair $Q[\bar{z}](X_\psi \to x_0.\mathsf{id} = y_0.\mathsf{id})$, where

(1) $Q[\bar{z}] = Q_1[\bar{x}] \cup Q_2[\bar{y}]$ is composed of two patterns $Q_1[\bar{x}] = (V_{Q_1}, E_{Q_1}, L_{Q_1})$ and $Q_2[\bar{y}] = (V_{Q_2}, E_{Q_2}, L_{Q_2})$, and $Q_2[\bar{y}]$ is a *copy* of $Q_1[\bar{x}]$ via a bijection $f : \bar{x} \mapsto \bar{y}$, such that (a) $\bar{x}$ and $\bar{y}$ are disjoint, and (b) f is an isomorphism from Q_1 to Q_2, *i.e.,* for each $x \in \bar{x}$, $L_{Q_1}(x) = L_{Q_2}(f(x))$; and $e = (x_1, x_2)$ is in E_{Q_1}, labeled with $L_{Q_1}(e)$ if and only if $e' = (f(x_1), f(x_2))$ is in E_{Q_2}, labeled with $L_{Q_2}(e') = L_{Q_1}(e)$; intuitively, $Q_2[\bar{y}]$ is $Q_1[\bar{x}]$ with variables renamed by f.

(2) $\bar{z}$ consists of $\bar{x}$ followed by $\bar{y}$, and $x_0 \in \bar{x}$ and $y_0 \in \bar{y}$ are designated nodes in ψ; and

(3) X_ψ is a set of literals of $\bar{z}$, where each literal is either (a) an id literal $x.\mathsf{id} = y.\mathsf{id}$ where $y = f(x)$, or (b) a variable literal $x.A \approx y.B$ where $y = f(x)$ and A and B are attributes in Θ that are not id. Here we assume that $\approx$ is a domain specific similarity predicate that can be defined in terms of any similarity metric used in entity matching, *e.g.,* q-grams, Jaro distance or edit distance, such that $a \approx b$ is true if a and b are "close" enough with respect to a predefined threshold δ. In particular, the equality relation $=$ is also a similarity predicate.

Semantics. To interpret GMKs, we use the following notations. Consider a match $h(\bar{z})$ of Q in a graph G, and a literal l of $\bar{z}$. We say that $h(\bar{z})$ satisfies l, if (a) when l is $x.A \approx y.B$, then attributes A and B exist at $v = h(x)$ and $v' = h(y)$, respectively, and $v.A \approx v'.B$; and (b) when l is $x.\mathsf{id} = y.\mathsf{id}$, then $h(x)$ and $h(y)$ refer to the same node.

We denote by $h(\bar{z}) \models X_\psi$ if $h(\bar{z})$ satisfies all the literals in X_ψ; We write $h(\bar{z}) \models X_\psi \to x_0.\mathsf{id} = y_0.\mathsf{id}$ if $h(\bar{z}) \models X_\psi$ implies that $h(x_0)$ and $h(y_0)$ refer to the same node in G.

A graph G satisfies GMK ψ, denoted by $G \models \psi$, if for all matches $h(\bar{z})$ of Q in G, $h(\bar{z}) \models X_\psi \to x_0.\mathsf{id} = y_0.\mathsf{id}$. A graph G satisfies a set Σ of GMKs if for all $\psi \in \Sigma$, $G \models \psi$, *i.e.,* G satisfies each GMK in Σ.

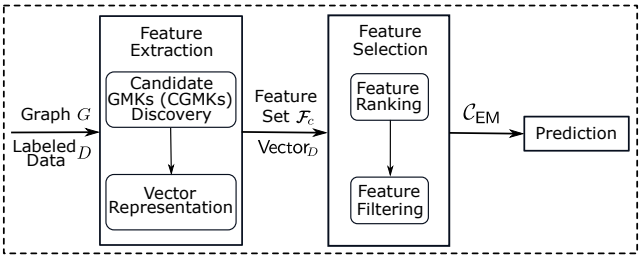

Fig. 1. GMKSLEM overview

III. Overview of GMKSLEM

We treat graph entity matching as a classification problem and present GMKSLEM, a supervised learning method for it. As depicted in Fig. 1, given a graph G and a labeled data D consisting of a finite number of triples (v, v', r), where v and v' are nodes of G and $r \in \{0, 1\}$. Here $r = 1$ if v and v' are identified as the same entity and $r = 0$ otherwise. GMKSLEM extracts a set Σ_c of GMKs, referred to as candidate GMKs (CGMKs), to generate a feature set $\mathcal{F}_c$, and then represents each node pair in D as a vector. We denote by Vector_D the set of all such vectors. Then it performs a process of feature selection to get a high-quality feature set $\mathcal{F}$ from $\mathcal{F}_c$. Finally, a classifier $\mathcal{C}_{\mathsf{EM}}$ is generated to predict if a pair of nodes refers to the same entity.

More specifically, GMKSLEM works as follows.

(1) GMKs feature extraction. The feature extraction consists of two steps: CGMKs discovery and vector representation.

CGMKs discovery. This procedure first finds all frequent graph patterns $Q[\bar{z}] = Q_1[\bar{x}] \cup Q_2[\bar{y}]$ over D such that the support $\mathsf{supp}(Q, D)$ in D is no less than a given threshold δ, where

$$\mathsf{supp}(Q, D) = \frac{|Q(x_0, y_0, G) \cap D_{\mathsf{pair}}|}{|D_{\mathsf{pair}}|}.$$

Here D_{pair} is the set of all pairs (v, v') such that $(v, v', r) \in D$ for some $r \in \{0, 1\}$. Intuitively, $\mathsf{supp}(Q, D)$ is the fraction of the node pairs in D that have matches in $Q(G)$.

More specifically, for each label a appearing in some node in D, we initially generate a single-node pattern Q_1 with a designated node x_0 labeled with a. Then Q_1 is expanded with edges, adding one edge at a time, when $\mathsf{supp}(Q, D) \geq \delta$, where $Q = Q_1 \cup Q_2$ and Q_2 is a copy of Q_1. We implement the expansion by using the tree structure in [8].

For each frequent graph pattern $Q[\bar{z}] = Q_1[\bar{x}] \cup Q_2[\bar{y}]$ with two designated nodes x_0 and y_0 and bijection f between $Q_1[\bar{x}]$ and $Q_2[\bar{y}]$, a CGMK $\psi : Q[\bar{z}](X_\psi \to x_0.\mathsf{id} = y_0.\mathsf{id})$ is constructed, where X_ψ consists of all literals $x.A_i \approx_j f(x).B_i$ and $x.\mathsf{id} = f(x).\mathsf{id}$, for every $x \in \bar{x}$, all labels A_i and B_i appearing in ψ_i, and all optional similarity functions $\approx_i$.

Vector representation. This procedure generates a feature vector $\mathsf{v}_{(v,v')}$ for every node pair (v, v') in D_{pair}, where each feature is a pair (ψ, l) for a CGMK ψ in Σ_c and a literal l in ψ. We denote by $\mathcal{F}_c$ the feature set consisting of all such features. The value of (ψ, l) in $\mathsf{v}_{(v,v')}$ is set to be the confidence $\mathsf{conf}_\psi(l, v, v')$, quantifying *"the greatest possibility"*

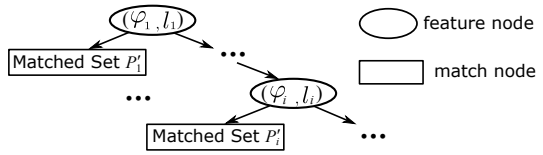

Fig. 2. Tree T_P for explaining matching results

of matches $h(\bar{z})$ in $Q(G)$ satisfying l, when $h(x_0) = v$ and $h(y_0) = v'$. Here Q is the pattern in ψ, and x_0 and y_0 are the designated nodes in ψ. Correspondingly, $\mathsf{conf}_\psi(l, v, v') = 0$ when $(v, v') \notin Q(x_0, y_0, G)$. This makes it possible to remove useless literals by the process of feature selection.

When $(v, v') \in Q(x_0, y_0, G)$, we define $\mathsf{conf}_\psi(l, v, v')$ as the maximum of similarities $\mathsf{s}(l, v, v', h)$ for all matches h of Q in G such that $h(x_0) = v$ and $h(y_0) = v'$, *i.e.,*

$$\mathsf{conf}_\psi(l, v, v') = \max_{h \in Q(G)} \{\mathsf{s}(l, v, v', h) | h(x_0, y_0) = (v, v')\};$$

otherwise, $\mathsf{conf}_\psi(l, v, v') = 0$.

Here $\mathsf{s}(l, v, v', h)$ is defined as follows: (a) if l is $x_i.A_i \approx y_i.B_i$, then $\mathsf{s}(l, v, v', h)$ is set to be the similarity between $h(x_i).A_i$ and $h(y_i).B_i$ in terms of the similarity metric used in $\approx$; and (b) if l is $x_i.\mathsf{id} = y_i.\mathsf{id}$, then $\mathsf{s}(l, v, v', h) = 1$ when $h(x_i) = h(y_i)$, *i.e.,* $h(x_i)$ and $h(y_i)$ are the same node in graph G, and $\mathsf{s}(l, v, v', h) = 0$ otherwise.

(2) Feature selection. Given set $\mathcal{F}_c$, feature selection is to identify the subset $\mathcal{F} \subseteq \mathcal{F}_c$ of high quality features for the classification. To do it, we use a two-step hybrid selection method including feature ranking and feature filtering based on an incremental wrapper selection over that ranking [13].

Feature ranking. We measure the importance of each feature θ in F_c by using its Gini importance score, which is known to provide a relative ranking of the features and is usually generated by training a random forest [14]. Intuitively, the higher Gini importance score a feature has, the more important it is for the classification. Then all the features in $\mathcal{F}_c$ are sorted in descending order on their Gini importance scores, becoming a ranked list, denoted by $L_{\mathcal{F}_c}$.

Feature filtering. This step evaluates the contribution of each θ in $L_{\mathcal{F}_c}$, and gets a subset $\mathcal{F}$ of $\mathcal{F}_c$ after discarding those features without contributions. It starts with $\mathcal{F} = \emptyset$ and runs over the ranked list $L_{\mathcal{F}_c}$ to add features to $\mathcal{F}$, one at a time, as follows. We train a classifier using features in $\mathcal{F} \cup \{\theta\}$ and validate their contributions using a validation set. A feature θ is added to $\mathcal{F}$ if the performance of this classifier is better than the classifier learned using $\mathcal{F}$; otherwise it is discarded. After the feature filtering, all the reserved features constitute the final high-quality feature set $\mathcal{F}$. Meanwhile, we get a new vector representation $\mathsf{Vector}'_{\mathsf{D}}$ for nodes pairs in D and a final classifier $\mathcal{C}_{\mathsf{EM}}$ to predict if an arbitrary pair of nodes match.

(3) Interpretability. We provide an approach to explaining why two entities match, by identifying the features (ψ, l) that make contributions. Let $\mathcal{C}_i (i \in [1, |\mathcal{F}|])$ be the classifiers trained by the first i features added to $\mathcal{F}$ during the process of feature filtering, and P be the set of node pairs that are to be classified. We denote by P_i $(i \in [1, |\mathcal{F}|])$ the set of node pairs (v, v') in P that are predicted to be matched by $\mathcal{F}_i$.

A tree T_P is built to help understand the predictions (See Fig. 2). There are two kinds of nodes in T_P: *feature nodes* and *match nodes*, storing features in $\mathcal{F}$ and sets of matched nodes in P, respectively. The root of T_P is a feature node that stores the first feature added in $\mathcal{F}$, denoted by (ψ_1, l_1). The right node at level $i \in [2, |\mathcal{F}|]$ is a feature node, storing the ith feature added to $\mathcal{F}$, denoted by (ψ_i, l_i), and the left node is a match node, carrying the set P'_i of newly matched node pairs, *i.e.,* $P'_i = P_i \setminus \bigcup_{j \in [1, i-1]} P_j$. Obviously, $P_1 = P'_1$.

Intuitively, P'_i consists of all node pairs in P that are predicted matched for the first time once feature (ψ_i, l_i) is used for training. That is, all the first i features $(\psi_1, l_1), \ldots, (\psi_i, l_i)$ make contributions for predicting matched node pairs in P'_i, which help users understand why those node pairs match.

TABLE I
GRAPHS CHARACTERISTICS

Dataset	$\|V\|$	$\|E\|$	#Pos.
BeRa [11]	7,416	7,345	68
iTAm [11]	71,253	125,660	132
LaMy [15]	1.0M	8.2M	1,381
FlLa [15]	0.3M	10.8M	510
DBLP [1]	6.2M	11.5M	38,731
Amazon [2]	2.4M	5.3M	31,025

TABLE II
ACCURACY COMPARISONS

Graph	DeepMatcher			Magellan			GMKSLEM			$\Delta\mathsf{F}_1$
	P	R	F_1	P	R	F_1	P	R	F_1	
BeRa	63.2	85.7	**72.7**	68.4	92.9	**78.8**	92.9	92.9	**92.9**	14.1
iTAm	92.0	85.2	**88.5**	86.7	96.3	**91.2**	90.0	100.0	**94.7**	3.5
LaMy	86.9	73.2	**79.4**	83.4	80.9	**82.4**	88.0	89.0	**88.5**	6.1
FlLa	100	81.8	**90.0**	96.1	73.7	**83.4**	93.9	93.9	**93.9**	3.9
DBLP	85.8	87.7	**86.7**	87.0	86.7	**86.8**	85.6	92.2	**88.8**	2.0
Amazon	95.1	95.9	**95.5**	97.4	94.6	**96.0**	94.9	97.2	**96.1**	0.1

IV. EXPERIMENTAL STUDY

Using real-life datasets, we experimentally evaluated GMKSLEM for the effectiveness and the interpretability.

Experimental setting. We used six real-life datasets and their details are shown in Table I, including the numbers of nodes ($|\boldsymbol{V}|$), edges ($|\boldsymbol{E}|$), and positive duplicate node pairs **#Pos.**.

Methods compared. We compared our method with two supervised learning-based entity matching methods: DeepMatcher [11] and Magellan [10]. We ran all neural network components in DeepMatcher including SIF, RNN, Attention and Hybrid, for five training epochs; and three ML modules *i.e.,* kNN, SVM and random forest (RF) for Magellan and GMKSLEM.

Implementation. We split node pairs including positive and negative samples into three parts with the ratio of 3:1:1, for training, validation and testing, respectively. For GMKSLEM, we use five string similarity functions as candidates in similarity predicates, including Jaro similarity (Jr), Jaccard similarity

[1] https://dblp.uni-trier.de/xml/
[2] http://snap.stanford.edu/data/amazon/

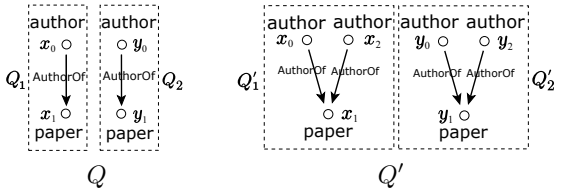

Fig. 3. Example of frequent graph patterns in DBLP

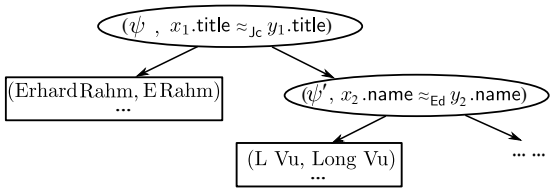

Fig. 4. Tree T_P for explaining the matching results for DBLP

(Jc), edit distance (Ed), Needleman-Wunsch (NW) algorithm and Smith-Waterman (SW) algorithm. We set the support threshold $s = 0.2$ for Amazon and 0.5 for the rest datasets. We deployed all the three methods on a Linux machine with 2.2GHz CPU and 96GB of memory. Each experiment was run 5 times and the average is reported here.

Experimental results. We next report our findings.

Effectiveness. The effectiveness is evaluated by precision (P), recall (R) and F_1 value defined as $2P \cdot R/(P + R)$. Table II shows the best P, R and F_1 scores of DeepMatcher, Magellan and GMKSLEM, as well as ΔF_1, the relative increase in F_1 of GMKSLEM over the best of DeepMatcher and Magellan. We use red font to highlight the highest F_1 score.

Compared with other methods, on average GMKSLEM achieves at least 4.95% improvements on both F_1 and recall. In detail, for BeRa and iTAm, on average, GMKSLEM outperforms DeepMatcher and Magellan by 8.8% on F_1. For LaMy, our method outperforms other methods by at least 1.1% and 8.1% on precision and recall, respectively. For FlLa, the precision of GMKSLEM is lower than the others; however, its recall is at least 12.1% higher than them. For DBLP and Amazon, the improvements are also reflected by the higher recall (more than 4.5% and 1.3% higher, respectively).

Interpretability. Figure 3 shows two frequent graph patterns Q and Q' discovered from DBLP, where $Q[x_0, x_1, y_0, y_1]$ consists of a pattern $Q_1[x_0, x_1]$ with variables x_0 and x_1, specifying a relationship between an author entity x_0 and a paper entity x_1, and a copy $Q_2[y_0, y_1]$ of $Q_1[x_0, x_1]$ with variables renamed; similarly for $Q'[x_0, x_1, x_2, y_0, y_1, y_2]$. Two CGMKs ψ, ψ' are generated as follows:

$$\psi : Q[x_0, x_1, y_0, y_1](X \to x_0.\mathsf{id} = y_0.\mathsf{id}),$$
$$\psi' : Q'[x_0, x_1, x_2, y_0, y_1, y_2](X' \to x_0.\mathsf{id} = y_0.\mathsf{id}),$$

where X and X' consist of all candidate literals of ψ and ψ', respectively (Details of X and X' are not given here).

The first two features added in the feature set $\mathcal{F}$ for DBLP is $l : x_1.\mathsf{title} \approx_{\mathsf{Jc}} y_1.\mathsf{title}$ and $l' : x_2.\mathsf{name} \approx_{\mathsf{Ed}} y_2.\mathsf{name}$. Figure 4 shows the first three levels of the tree T_P for DBLP. It tell us that (a) "Erhard Rahm" and "E Rahm" can be identified as the same author because the papers they published have similar titles by Jc metric; and (b) however, it is not enough to identify "L Vu" and "Long Vu", unless we consider the similarity of their co-authors' names evaluated by Ed.

V. Conclusion

We have proposed a class of GMKs with similarity predicates for entity matching on graphs. We have developed a supervised learning method GMKSLEM, which supports to explain the matching results. We have presented a feature extraction method to discover GMKs to generate features for vector representation, and a feature selection method to discard those features with low contributions in entity matching. We have empirically verified the effectiveness of our method and provided a case of interpretability.

VI. Acknowledgment

This work is supported in part by The National Key Research and Development Program of China (2016YFB1000103) and NSFC (61602023, 61421003).

References

[1] W. Fan, Z. Fan, C. Tian, and X. L. Dong, "Keys for graphs," *PVLDB*, vol. 8, no. 12, pp. 1590–1601, Aug. 2015.
[2] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *ACM TKDD*, vol. 1, no. 1, 2007.
[3] H. Ma, M. Alipourlangouri, Y. Wu, F. Chiang, and J. Pi, "Ontology-based entity matching in attributed graphs," *PVLDB*, vol. 12, no. 10, pp. 1195–1207, Jun. 2019.
[4] W. Fan and P. Lu, "Dependencies for graphs," *ACM TODS*, vol. 44, no. 2, pp. 5:1–5:40, 2019.
[5] S. Kwashie, L. Liu, J. Liu, M. Stumptner, J. Li, and L. Yang, "Certus: an effective entity resolution approach with graph differential dependencies (GDDs)," *PVLDB*, vol. 12, no. 6, pp. 653–666, 2019.
[6] W. Fan, P. Lu, C. Tian, and J. Zhou, "Deducing certain fixes to graphs," *PVLDB*, vol. 12, no. 7, pp. 752–765, 2019.
[7] J. Liu, J. Li, C. Liu, and Y. Chen, "Discover dependencies from data—a review," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 251–264, 2012.
[8] W. Fan, C. Hu, X. Liu, and P. Lu, "Discovering graph functional dependencies," in *SIGMOD*, 2018, pp. 427–439.
[9] M. Alipourlangouri and F. Chiang, "Keyminer: Discovering keys for graphs," in *VLDB workshop*, 2018, pp. 1–7.
[10] P. Konda, S. Das, P. Suganthan G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra, "Magellan: Toward building entity matching management systems," *PVLDB*, vol. 9, no. 12, pp. 1197–1208, 2016.
[11] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *SIGMOD*, 2018, pp. 19–34.
[12] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang, "Distributed representations of tuples for entity resolution," *PVLDB*, vol. 11, no. 11, pp. 1454–1467, 2018.
[13] P. Bermejo, L. de la Ossa, J. A. Gámez, and J. M. Puerta, "Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 35–44, 2012.
[14] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. A. Hamprecht, "A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data," *BMC bioinformatics*, vol. 10, no. 1, p. 213, 2009.
[15] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "Cosnet: Connecting heterogeneous social networks with local and global consistency," in *KDD*, 2015, pp. 1485–1494.