Ontology Reshaping for Knowledge Graph Construction: Applied on Bosch Welding Case

Dongzhuoran Zhou[1,2], Baifan Zhou[2(✉)], Zhuoxun Zheng[1,3], Ahmet Soylu[3], Gong Cheng[4], Ernesto Jimenez-Ruiz[2,5], Egor V. Kostylev[2], and Evgeny Kharlamov[1,2]

[1] Bosch Center for Artificial Intelligence, Renningen, Germany
dongzhuoran.zhou@de.bosch.com
[2] Department of Informatics, Univeristy of Oslo, Oslo, Norway
baifanz@ifi.uio.no
[3] Department of Computer Science, Oslo Metropolitan University, Oslo, Norway
[4] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[5] Department of Computer Science, City University of London, London, UK

Abstract. Automatic knowledge graph (KG) construction is widely used in industry for data integration and access, and there are several approaches to enable (semi-)automatic construction of knowledge graphs. One important approach is to map the raw data to a given knowledge graph schema, often a domain ontology, and construct the entities and properties according to the ontology. However, the existing approaches to construct knowledge graphs are not always efficient enough and the resulting knowledge graphs are not sufficiently application-oriented and user-friendly. The challenge arises from the trade-off: the domain ontology should be knowledge-oriented, to reflect the general domain knowledge rather than data particularities; while a knowledge graph schema should be data-oriented, to cover all data features. If the former is directly used as the knowledge graph schema, this can cause issues like blank nodes created due to classes unmapped to data and deep knowledge graph structures. To this end, we propose a system for ontology reshaping, which generates knowledge graph schemata that fully cover the data while also covers domain knowledge well. We evaluated our approach extensively with a user study and three real manufacturing datasets from Bosch against four baselines, showing promising results.

Keywords: Semantic data integration · Knowledge graph · Ontology reshaping · Graph algorithm · Automatic knowledge graph construction

1 Introduction

Knowledge graphs (KG) allow to structure information in terms of nodes and edges [17]. The nodes represent entities of interests. The edges that connect entities represent relationships between them. The edges that connect entities to their data values, represent the data properties of the entities. In the context of Industry 4.0 [26] and

D. Zhou and B. Zhou—Contributed equally to this work as first authors.

U. Sattler et al. (Eds.): ISWC 2022, LNCS 13489, pp. 770–790, 2022.
https://doi.org/10.1007/978-3-031-19433-7_44

Internet of Things [20], knowledge graphs have been successfully used in a wide range of applications and industrial sectors [18,37,38,41,52,59].

Due to the complexity and variety of industrial data (the typical example is relational tables [54]), it is very desired to facilitate automation of knowledge graph construction [39]. A common approach on knowledge graph construction is to construct entities and properties by relying on a given knowledge graph schema, often a domain ontology (Fig. 1a). This approach matches the attributes names in raw data to entities and properties in knowledge graph, then organise them in the same pattern as the schema [9,22,28]. However, the existing approaches to construct knowledge graphs are not always efficient enough and the resulting knowledge graphs are not sufficiently application-oriented and user-friendly. The challenge arises from the trade-off between the knowledge-orientation and data-orientation: A classical domain ontology is a formal specification of shared conceptualisation of knowledge [14,40]. It should be *knowledge-oriented*, to reflect the experts knowledge on upper level concepts, specific domains, or applications, rather than data particularities of arbitrary datasets [27]; while a knowledge graph schema should be *data-oriented*, to cover all the features (columns in tables) and have limited number of blank nodes. If a knowledge-oriented domain ontology is directly used as the knowledge graph schema, this can cause a series of issues, e.g., the data integrated with the help of domain ontologies suffers from a high load of blank nodes in knowledge graphs that result from data integration, e.g., up to 90% of information in the knowledge graph are blank nodes [16].

Indeed, sparse knowledge graphs are hard to digest for end-users: browsing them is a bad experience, users will have to go through hordes of blank nodes. Then, blank nodes affect application development. The applications should adapt to the structure of the knowledge graph, e.g., by reflecting this structure in SPARQL queries, thus the queries will have to handle and skip many bank nodes. Then, the bigger a knowledge graph gets the mode difficult is to process or search in it. Thus, it is desired to reduce the number of spurious blank nodes and to make knowledge graphs more compact.

Considering an example in Fig. 1c-d, where classes and data properties in the domain ontology ($\mathcal{G}^{do}$) are mapped to tables and attributes in the relational schema (R). There exist many discrepancies between $\mathcal{G}^{do}$ and R. If $\mathcal{G}^{do}$ is directly used as the schema to construct knowledge graphs, a number of issues will arise: many classes in $\mathcal{G}^{do}$ that are not mapped to any tables or attributes in R will lead to blank nodes (or dummy nodes); the attribute DP2 will be connected to a dummy class C6, instead of C1, which it should be connected to, etc.

Past works like ontology modularisation, summarisation did not address the challenge, because they still use the domain ontology to construct knowledge graph. Our previous work [60] could convert the domain ontology to data-oriented ontologies as knowledge graph schemata, but did not provide interoperability between these knowledge graphs and also did not fully exploit the knowledge in the domain ontology. A better solution is to have data-oriented knowledge graph schemata while still preserve knowledge in the domain ontology.

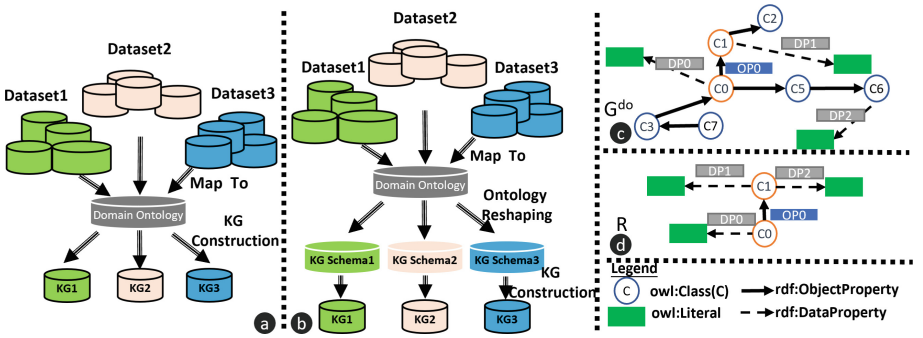

Fig. 1. (a) Ontology-based knowledge graph construction without ontology reshaping generates sparse knowledge graphs with many dummy nodes, which are generated based on classes in the knowledge graph schema that do not have correspondence in the raw data; (b) knowledge graph-construction with ontology reshaping that converts the general domain ontology to data-specific knowledge graph schemata, which makes the knowledge graph more user-friendly. (c) Domain ontology reflects the domain knowledge; (d) The knowledge graph schema needs to reflect raw relational data schema specificities and usability. orange and red circles: classes that can be mapped to attributes in the relational data schema; blue circles: classes that cannot be found in the relational data schema. (Color figure online)

To this end, we propose our knowledge graph construction system that relies on the OntoReshape$^+$ algorithm to "reshape" a given domain ontology to data-oriented knowledge graph schemata (Fig. 1b), better incorporates knowledge in the domain ontology, and provide interoperability between the knowledge graphs based on the reshaped knowledge graph schemata. Our contributions are as follows:

- We introduce a use case of knowledge graph generation for welding quality monitoring which shows the challenge of sparse knowledge graphs constructed from raw data based on the domain ontology as the schema.
- We derive the four requirements: data coverage, knowledge coverage, user-friendliness and efficiency, from the use case perspective, and mathematically abstract them.
- We propose an algorithm, OntoReshape$^+$, which can fully satisfy data coverage while better incorporates knowledge from the domain ontology, compared to the baselines.
- We implemented the algorithms in system of knowledge graph construction enhanced by ontology reshaping, which can automatically reshape the domain ontology to data-oriented ontologies that serve as knowledge graph schemata, and construct the knowledge graph without dummy nodes.
- We evaluated our approach extensively with a user study and three real manufacturing dataset from Bosch against four benchmarks, showing promising results.

This paper is organised as follows. Section 2 introduces Bosch manufacturing welding use case. Section 3 introduces some preliminary knowledge. Section 4 presents our method. Section 5 evaluates the method. Section 6 discusses related work. Section 7 concludes the paper.

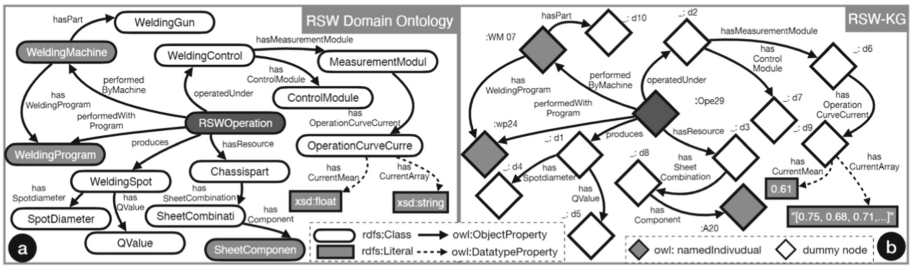

Fig. 2. Schematic illustration of the (a) domain ontology (partial) and (b) an excerpt of knowledge graph constructed by directly using the domain ontology as the knowledge graph schema, which has many dummy nodes due to classes in (a) that are unmapped to the data.

2 The Bosch Welding Use Case

Resistance Spot Welding and Quality Monitoring. Resistance spot welding is a type of automated welding process that accounts for millions of car production globally. During the welding, the electrode presses the worksheets (car bodies) and passes a high current through the electrodes and the worksheets [53,55]. The material in the small area between the electrodes will melt due to the heat generated by electricity and then congeal after cooling down, forming a welding spot that connects the worksheets by controlling robot arm positioning [3,34]. Multiple quality indicators, e.g. the spot diameter, are monitored to ensure the welding quality. The quality monitoring of resistance spot welding is essential and involves large amounts of data collected from welding process.

Bosch Welding Data with High Variety. Bosch welding data come from multiple sources [44,57], e.g. welding production plants, welding laboratories, analytical or numerical simulation models in Bosch's research centres. Just taking the production data as example, whose sources are hundreds of Bosch plants worldwide and many Bosch's renowned customers [47]. These data are highly diversified because they are collected with various sensors settings, formats, databases, software versions, etc. that are tailored to individual customer needs and factory specifications [51,58,61].

Data Integration, Domain Ontology and Knowledge Graph. Due to the many discrepancies of data semantics and formats, data integration is essential for building user-friendly, sustainable and efficient industrial solutions [45,56]. Bosch adopts semantic data integration that relies on domain ontologies to transform various data into uniform data formats, one typical example of which is knowledge graph for it provides an efficient foundation for many applications. The welding domain ontology is usually generated by semantic experts or domain experts, and should reflect the general resistance welding knowledge across different scenarios of production, laboratory and simulation (Fig. 2). It is modelled in OWL 2 language and has a large number of axioms. One of such example has 1181 axioms that describe 210 classes, 203 object properties, and 191 datatype properties. In contrast, the various welding datasets may have a much smaller scope. For example, one production dataset only contains data generated by the welding

control of a particular welding setting or a specific software version, and miss large data that are measured in other settings, software versions, or in laboratory or simulation. On the other hand, laboratory and simulation data enjoy the flexibility of sensor installation that would be otherwise extremely costly to realise in the real production. Traditional approaches that use a common domain ontology as the knowledge graph schema for integrating various data will cause a series of issues, discussed in next section.

```
SELECT ?ca
WHERE {
    ?op rdf:type rsw:RSWOperation
    ?op rsw_kg:operatedUnder ?wc .
    ?wc rsw_kg:hasMeasurementModule ?mm .
    ?mm rsw_kg:hasOperationCurveCurent ?occ
    ?occ rsw_kg:CurrentArray ?ca .
}
```

a

```
SELECT ?ca
WHERE {
    ?op rdf:type rsw:RSWOperation
    ?op rsw_kg:CurrentArray ?ca .
}
```

b

Fig. 3. (a) An example query to retrieve the current sensor measurement array, over knowledge graph constructed based on the domain ontology. (b) The query that retrieves the same results over knowledge graph constructed based on the reshaped ontology, which is much user-friendly than that in (a).

Cumbersome KGs and Long Queries due to KG Schema. The knowledge graphs integrated from various data sources with the same domain ontology as the knowledge graph schema enjoys the data interoperability, namely uniform data access across all datasets. However, it also has serious drawbacks. Considering the example knowledge graph (Fig. 5b) generated with the schema in Fig. 5a, where the black blocks with white background are dummy nodes, generated because classes in the domain ontology is not mapped to anything in the data. The number of such dummy nodes are very high, up to 63.6%. The dummy nodes cause the knowledge graph to be unnecessarily cumbersome, consuming much computational power in generation and storage resource in the database. In addition, they also lead to superfluously long queries (Fig. 3a) that need to traverse many dummy nodes during data accessing, which is neither technologically-friendly nor user-friendly. Moreover, our users also complain that some knowledge graphs based on domain ontologies have disconnected sub-graphs that cannot be reached with queries starting from the welding operation, which is the most important node in the knowledge graphs that they usually start in the queries. They prefer connected knowledge graphs schemata.

Requirements for the Ontology Reshaping System. Both from the system and user view, it is highly desired to simplify the knowledge graph schemata to avoid the dummy nodes while still cover all the data and reflect the domain knowledge, so that the knowledge graphs become much more efficient and queries become simpler (Fig. 3b). We thus derive the following requirements for the new knowledge graph schemata and for the algorithm and system that generates the knowledge graph schemata and facilitates knowledge graph construction:

– *R1 Data Coverage.* The knowledge graph schemata generated by system should still cover all the data, e.g. including table names and attribute names for relational tables.

- *R2 Knowledge Coverage.* The knowledge graph schemata should still possibly preserve the knowledge encoded in the domain ontology. It should be similar to the domain ontology, either judged by the users or with some metrics.
- *R3 User-friendliness.* The user-friendliness involves at least 3 aspects: R3.1, the knowledge graphs constructed based on the new knowledge graph schemata should possible have very few dummy nodes, ideally zero (we call this the *succinctness* of the knowledge graph schemata or the knowledge graph); R3.2, the knowledge graphs schemata should be connected, namely no disconnected sub-graphs, so that the users can reach all nodes relevant to when they write queries (connectivity); R3.3, users prefer simpler and shorter queries than long queries when they can retrieve the same information. Thus, the constructed knowledge graphs should possibly have shallower structure (simplicity). Apart from that, the system for generating knowledge graph schemata and constructing knowledge graphs should also be user-friendly. This is commonly known as system usability [19] in terms of human machine interaction. It is evaluated by effectiveness, user efficiency (note this is the efficiency of users using the system, different from the R4 system efficiency), and user satisfaction of the system.
- *R4 System Efficiency.* The system efficiency measures two aspects: time efficiency, namely the overall time for generating the knowledge graph schemata and constructing the knowledge graphs, and the space efficiency, the storage space needed for the knowledge graphs to store the same information.

3 Preliminaries

Concepts and Problem Formulation. We formulate the problem of *Ontology Reshaping* as problem of computing from a given ontology and some context, a new ontology that *fully* satisfies the requirement R1 (Sect. 2) and achieves possibly good performance in terms of R2-R4. In particular, in this work we focus on specific type of contexts that can be formulated as follows:

$$\text{Ontology Reshaping} : (\mathcal{G}^{do}, R, M^{do}, U) \rightarrow \mathcal{G}^{ro}, M^{ro} \tag{1}$$

where $\mathcal{G}^{do}$ is a given domain ontology, R is a relational schema of relational tables, M^{do} is a mapping between R and $\mathcal{G}^{do}$, U is optional user information, and $\mathcal{G}^{ro}$ is the "reshaped" ontology, M^{ro} is a mapping between $\mathcal{G}^{ro}$ and R– defined as follows:

An *Ontology* in the context of our work is a directed labelled multigraph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, e.g., projected[1] from a set of OWL 2 axioms (e.g., the domain ontology $\mathcal{G}^{do}$ and reshaped ontology $\mathcal{G}^{ro}$) as follows: The classes are projected to class nodes $\mathcal{N}^C$, the datatypes to datatype nodes $\mathcal{N}^D$, the object properties to object property edges $\mathcal{E}^O$, and the datatype properties to datatype property edges $\mathcal{E}^D$.

A *Relational Schema* (R) is a finite set relational tables $R = \{\mathtt{T}_1(\mathtt{A}), ..., \mathtt{T}_n(\mathtt{A})\}$, where $\mathtt{T}_i$ is a table name while $\mathtt{A}$ is a finite set of attributes $\mathtt{A} = \{a_1, ..., a_k\}$ represented by

[1] Ontology projections typically do not preserve all information captured by ontologies, but they are sufficient for our purpose of ontology reshaping.

their attribute names a_j. Among the attributes, there exist attributes called the primary key $\mathtt{A}_p$ (each table only one) that uniquely identifies the rows, (optionally) foreign key attributes $\mathtt{A}^f$ refer to the primary keys of other tables, and normal attributes $\mathtt{A}^n$ that contain normal data.

A *Mapping* (M) is a bidirectional function that maps the elements in R to elements in $\mathcal{G}$. The Raw-to-DO Mapping (raw data to domain ontology mapping) M^{do} maps the table names $\mathtt{T}$ in R to class nodes $\mathcal{N}^C$ in $\mathcal{G}^{do}$, normal attributes $\mathtt{A}^n$ to datatype property edges $\mathcal{E}^D$, and foreign keys $\mathtt{A}^f$ to object property edges $\mathcal{E}^O$, and vice versa. Similarly, the generated Raw-to-RO Mapping (raw data to reshaped ontology mapping) M^{ro} maps the $\mathtt{T}$, $\mathtt{A}^n$, $\mathtt{A}^f$ to $\mathcal{N}^C$, $\mathcal{E}^D$, $\mathcal{E}^O$ in $\mathcal{G}^{ro}$. In this work, we assume the mapping M^{do} is one-to-one mapping that maps all elements in R to elements in $\mathcal{G}^{do}$.[2] Similarly, the generated M^{ro} is also one-to-one mapping.

$$M : \{\mathtt{T} \leftrightarrow \mathcal{N}^C, \mathtt{A}^n \leftrightarrow \mathcal{E}^D, \mathtt{A}^f \leftrightarrow \mathcal{E}^O \mid \mathtt{T}, \mathtt{A}^n, \mathtt{A}^f \in R,\ \mathcal{N}^C, \mathcal{E}^D, \mathcal{E}^O \in \mathcal{G}\}.$$

The *User Information* (U) can be understood as (1) a mandatory label that labels a node in $\mathcal{G}^{do}$ as the most important node for the users, named as the *main node*, n_m; (2) an extra set of mappings that map some normal attributes $\mathtt{A}^n$ in R to class nodes $\mathcal{N}^C$ in $\mathcal{G}^{do}$: $U : \{\mathtt{A}^n \leftrightarrow \mathcal{N}^C \mid \mathtt{A}^n \in R, \mathcal{N}^C \in \mathcal{G}^{do}\}$.

The *Dummy Nodes* $\mathcal{N}^{dummy}$ are the nodes in the knowledge graph schema $\mathcal{G}$ (and the knowledge graph constructed based on $\mathcal{G}$) that cannot be mapped to any elements in R.

Mathematical Abstraction of Requirements. Following the requirements for the system in Sect. 2, we derive their mathematical abstraction. The R1-R3 are designed in a way that they range from 0 to 1. The closer to 1 they are, the better performance the ontology reshaping algorithm has

- *R1 Data Coverage*, this is measured by the number of elements in R mapped to $\mathcal{G}^{ro}$:
- *R2 Knowledge Coverage*, $\mathcal{G}^{ro}$ should preserve possible many nodes and edges in $\mathcal{G}^{do}$, measured by the number of elements in $\mathcal{G}^{do}$ kept in $\mathcal{G}^{ro}$. We use the formula to transform this metric to a range between(0,1]: (|{n}| + |{e}|) / ($|\mathcal{N}^{do}| + |\mathcal{E}^{do}|$), where $\exists\, n^{do} \in \mathcal{N}^{do}, n \leftrightarrow n^{do}, \exists\, e^{do} \in \mathcal{E}^{do}, e \leftrightarrow e^{do}$, n,e$\in \mathcal{G}^{ro}$.
- *R3 User-friendliness*, calculated in 3 aspects:
  - *R3.1 Succinctness*, measured by the percentage of non-dummy nodes divided by the total number of nodes: $|\mathcal{N}^{dummy}|/|\mathcal{N}^{ro}|, \mathcal{N}^{dummy} \subset \mathcal{N}^{ro}$.
  - *R3.2 Connectivity*, determined by the number of required extra edges e needed to connect $\mathcal{G}^{do}$. We use the formula to transform this metric to a range between (0,1]: 1/(1 + #e).
  - *R3.3 Simplicity*, determined by the graph diameter d of $\mathcal{G}^{ro}$. We use the formula to transform this metric to a range between (0,1]: $1/d$.
- *R4 Efficiency*. The time efficiency is measured by the total time of ontology reshaping and knowledge graph construction based on knowledge graph schema. The space efficiency is measured by the storage space needed for the constructed knowledge graph.

[2] Note it is not the same case for the other way around: there normally exist many nodes or edges in $\mathcal{G}^{do}$ that cannot be mapped to any elements in R.

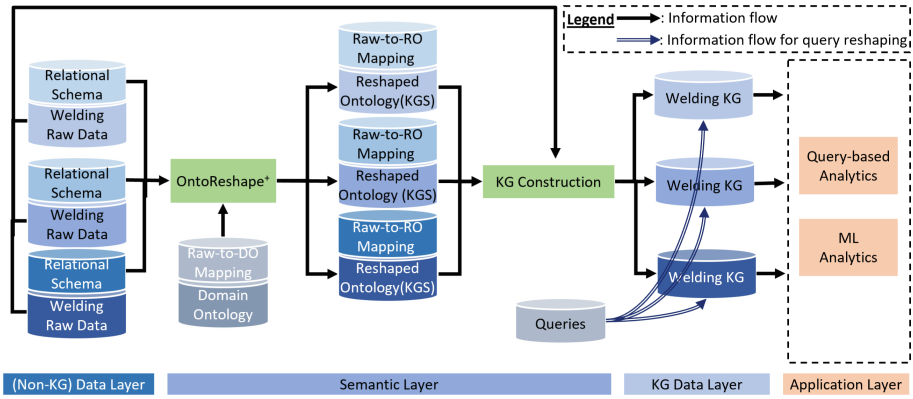

Fig. 4. An architectural overview of our KG solution. KG: knowledge graph. KGS: KG schema.

4 Our Approach

4.1 Architectural Overview

We now walk through the readers through our ontology reshaping system (Fig. 4) The system consists of four layers: *(Non-KG) Data Layer*, *Semantic Layer*, *KG Data Layer*, and *Application Layer*. From the very left, the *(Non-KG) Data Layer* contains the *Welding Raw Data*. The *Welding Raw Data* are in the form of relational tables and also have their corresponding *Relational Schemata*. The *Semantic Layer* contains several semantic artefacts and semantic modules. The *OntoReshape*$^+$ module takes the *Domain Ontology* $\mathcal{G}^{do}$, the *Raw-to-DO Mapping* M^{do} (raw data to domain ontology), and the *Relational Schemata* R (in addition, the user information U) as inputs, and generates a series of *Reshaped Ontology* $\mathcal{G}^{ro}$ (*KG Schemata* at the same time) and their corresponding *Raw-to-RO Mappings* M^{ro}. These *KG Schemata* and *Raw-to-RO Mappings* are then used by the *KG Construction* module to construct the *Welding KGs* from the *Welding Raw Data*. And common *Queries* are selected by the users for welding quality monitoring. The *Welding KGs* in the *KG Data Layer* then can be used for applications like *Query-Based Analytics* and *ML Analytics* [59] in *Application Layer*.

4.2 Semantic Artefacts

Ontologies. The three different type of ontologies are domain ontology, relational schema graph and KG schema.

Domain Ontology $\mathcal{G}^{do}$. The domain ontology models the general knowledge of resistance welding spot manufacturing process (Fig. 2) and should cover all attributes in the common Bosch datasets in our consideration. The domain ontology has the RSWOperation as the most important class, where the RSWOperation is a welding operation that produces an atomic product. The RSWOperation takes sheet components with specified combination in, choose the specific welding machine and outputs the welding sheet combination with welding spots.

Reshaped Ontology $\mathcal{G}^{ro}$. The reshaped ontology is similar to domain ontologies. Our reshaped ontology are reshaped from the Domain Ontology $\mathcal{G}^{do}$ by Algorithm 1. An example is given by Fig. 5d. The reshaped ontologies are the simplified knowledge graph schemata, and keep the necessary parts to cover the specified datasets, which are then used as the schema of Welding knowledge graph.

Mapping. The system has two types of mappings: Raw-to-DO Mapping M^{do} (raw data to domain ontology) and the Raw-to-RO Mapping M^{ro} (raw data to reshaped ontology).

Raw-to-DO Mapping M^{do} is generated manually by users (welding experts). It should map all tables and attributes in the data to the nodes or edges in the domain ontology. Thus, each dataset has its own M^{do}.

Raw-to-RO Mapping M^{ro} is automatically generated by the ontology reshaping algorithm, accompanying the reshaped ontology $\mathcal{G}^{ro}$. It is needed for every $\mathcal{G}^{ro}$ since every $\mathcal{G}^{ro}$ will be used for data integration. M^{ro} reuses most of the M^{do} and should map all tables and attributes in the raw data to the nodes and edges in $\mathcal{G}^{ro}$.

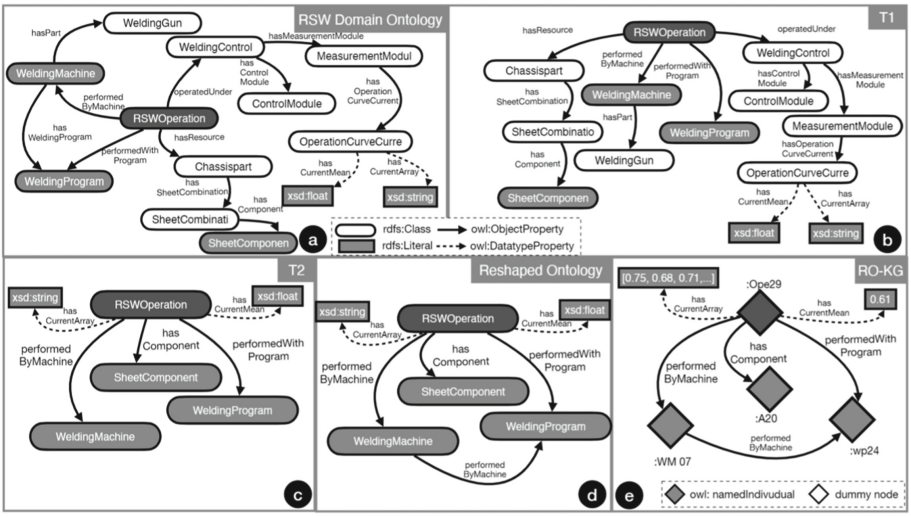

Fig. 5. (a) Schematic illustration of a small excerpt of the domain ontology $\mathcal{G}^{do}$. (b) Intermediate results in OntoReshape^{+}: Tree 1 $\mathcal{T}_1$ and (c) Tree 2 $\mathcal{T}_2$. (d) Reshaped ontology $\mathcal{G}^{ro}$. (e) knowledge graph constructed based on (d).

Queries. The queries in our system are SPARQL queries with the backbone as Basic Graph Pattern (BGP) query.

4.3 The Algorithm OntoReshape^{+}

Intuition. The intuition behind our algorithm OntoReshape^{+} is to select subsets of nodes and edges from a given domain ontology $\mathcal{G}^{do}$, which can be mapped to a relational schema R or included in the user information U, and then connect the selected

Algorithm 1: Schema Reshaping

Input: $\mathcal{G}^{do}$, R, M^{ro}, U
Output: $\mathcal{G}^{ro}$

1 $\mathcal{T}_1,\ \mathcal{E}_1^{deleted} \leftarrow$ Graph2Tree($\mathcal{G}^{do}, U$)
2 $\mathcal{T}_2,\ \mathcal{E}_2^{deleted},\ M^{ro} \leftarrow$ TreeCollapse($\mathcal{T}_1,\ R,\ M^{do},\ U$)
3 $\mathcal{G}^{ro} \leftarrow \mathcal{T}_2 \cup \{e(n_t, n_h) \mid e(n_t, n_h) \in \mathcal{E}_1^{deleted} \cup \mathcal{E}_2^{deleted}, n_t \in \mathcal{T}_2, n_h \in \mathcal{T}_2\}$

subsets with possibly more edges in $\mathcal{G}^{do}$, thus generating the reshaped ontology $\mathcal{G}^{ro}$. More specifically, OntoReshape$^+$ does so in three steps:

- *Step 1*, it transforms $\mathcal{G}^{do}$ (Fig. 5.a) to a tree $\mathcal{T}_1$ (Fig. 5 b) by removing some edges, where the tree has the *main node* n_m given in U as the root;
- *Step 2*, it selects the subsets of nodes and edges of $\mathcal{T}_1$ that are mapped in R by M^{do} or pointed by the users, creating a $\mathcal{T}_2$ (Fig. 5 c);
- *Step 3*, some deleted edges in Step1 and Step 2 are added back to $\mathcal{T}_2$, where these edges have both their head and tail in $\mathcal{T}_2$, resulting $\mathcal{G}^{ro}$ (Fig. 5 d).

Step 1. Graph2Tree. With n_m as the root node, Step 1 (Algorithm 2) expands the tree $\mathcal{T}_1$ with nodes and edges selected from $\mathcal{G}^{do}$ layer by layer, in a way that there exists only one path between any node and n_m. We first clarify several concepts used in the step: $\mathcal{N}^{leaf}$ refer to the set of leaf nodes of $\mathcal{T}_1$, $\mathcal{N}^{ring}$ refers to the set of "ring nodes" (nodes in a outer layer of the leaf nodes) that are potential to be added to $\mathcal{T}_1$, $\mathcal{N}^{visited}$ is the set of visited nodes, and $\mathcal{E}_1^{deleted}$ is a set of the deleted edges. Then we introduce the procedure. First, Algorithm 2 reads the user information to mark the main node n_m, and initialise $\mathcal{T}_1, \mathcal{N}^{leaf}, \mathcal{N}^{visited}$ with n_m, and the set $\mathcal{E}_1^{deleted}$ with the empty set (Line 1). Next, if $\mathcal{N}^{leaf}$ is not empty, Algorithm 2 does the following steps: it initialises an empty set $\mathcal{N}^{ring}$ (Line3), then it enumerates each node n_i in the current $\mathcal{N}^{leaf}$ (Line 4) and create an empty set of ring nodes $\mathcal{N}_i^{ring}$ that belong to n_i. For each leaf node n_i, it enumerates the edges incidental to the node n_i in $\mathcal{G}^{do}$, $e^u(n_i, n_j)$[3], but not in $\mathcal{E}_1^{deleted}$, and exams the other node n_j that this edge is connected to. If n_j is not visited (not in $\mathcal{N}^{visited}$), then the node n_j and the edge $e^u(n_i, n_j)$ are added to $\mathcal{T}_1$ (Line 8), n_j is added to $\mathcal{N}^{visited}$ and a new ring set $\mathcal{N}_i^{ring}$ that belongs to n_i (Line 9–10), and . If n_j is already visited, the edge $e^u(n_i, n_j)$ is added to $\mathcal{E}_1^{deleted}$ (Line 12). After all $e^u(n_i, n_j)$ are enumerated, all elements in $\mathcal{N}_i^{ring}$ are added to $\mathcal{N}^{ring}$ (Line 13). After all n_i are numerated, the $\mathcal{N}^{ring}$ becomes the new $\mathcal{N}^{leaf}$ (Line 14).

Step 3. Tree collapse. Step 3 (in Algorithm 3) selects nodes in $\mathcal{G}^{rs}$, by user or rule, and save them in $\mathcal{N}^{selected}$, then deletes the nodes not in $\mathcal{N}^{selected}$ from $\mathcal{T}_2$ which is copied by $\mathcal{T}_1$, at the same time keeps the connectivity of $\mathcal{T}_2$. It takes 4 inputs: the tree $\mathcal{T}_1$, the relational schema R, raw data to domain ontology mapping M^{do}, and user information U. The algorithm firstly inisialised the ring node set $\mathcal{N}^{ring}$ with main node n_m, $\mathcal{T}_2$ with $\mathcal{T}_1$, and deleted edge set $\mathcal{E}_2^{deleted}$ for$\mathcal{T}_2$ with empty set (Line 1). Then the algorithm selects the nodes in relational schema graph $\mathcal{G}^{rs}$, or with datatype property

[3] Here we use $e^u(n_i, n_j)$ to represent both the edge $e(n_i, n_j)$ and $e(n_i, n_j)$.

Algorithm 2: Graph2Tree

Input: $\mathcal{G}^{do}, U$
Output: $\mathcal{T}_1,\ \mathcal{E}_1^{deleted}$

1 **Initialisation:** $n_m \leftarrow$*ReadUserInfo(U);* $\mathcal{T}_1, \mathcal{N}^{leaf}, \mathcal{N}^{visited} \leftarrow \{n_m\};\ \mathcal{E}_1^{deleted} \leftarrow \{\}$
2 **while** $\mathcal{N}^{leaf} \neq \varnothing$ **do**
3 $\mathcal{N}^{ring} \leftarrow \{\}$
4 **foreach** $n_i \in \mathcal{N}^{leaf}$ **do**
5 $\mathcal{N}_i^{ring} \leftarrow \{\}$
6 **foreach** $e^u(n_i, n_j) \in \mathcal{G}^{do} \setminus \mathcal{E}_1^{deleted}, n_j \in \mathcal{G}^{do}$ **do**
7 **if** $n_j \notin \mathcal{N}^{visited}$ **then**
8 $\mathcal{T}_1 := \mathcal{T}_1 \cup \{n_j, e^u(n_i, n_j)\}$
9 $\mathcal{N}^{visited} := \mathcal{N}^{visited} \cup \{n_j\}$
10 $\mathcal{N}_i^{ring} := \mathcal{N}_i^{ring} \cup \{n_j\}$
11 **else**
12 $\mathcal{E}_1^{deleted} := \mathcal{E}_1^{deleted} \cup \{e^u(n_i, n_j)\}$
13 $\mathcal{N}^{ring} := \mathcal{N}^{ring} \cup \mathcal{N}_i^{ring}$
14 $\mathcal{N}^{leaf} \leftarrow \mathcal{N}^{ring}$

having"ID" or "Name", or by user choices. These nodes are added into $\mathcal{N}^{selected}$ (Line 2). If $\mathcal{N}^{ring}$ is not empty, the Algorithm 3 does the following steps: it inisialise an empty set $\mathcal{N}_{next}^{ring}$, then it enumerate each node n_i in the current $\mathcal{N}^{leaf}$ (Line 4). For each leaf node n_i, it enumerates the edges incidental to the node n_i in $\mathcal{T}_1$, $e^u(n_i, n_j)$. If n_j is not selected (not in $\mathcal{N}^{selected}$), then the node n_j and edge $e^u(n_i, n_j)$ are deleted from $\mathcal{T}_2$ and $e^u(n_i, n_j)$ is added to $\mathcal{E}_2^{deleted}$. If the edge $e^u(n_j, n_k)$ is in $\mathcal{T}_1$,then $e^u(n_j, n_k)$ is deleted from $\mathcal{T}_2$, and a new edge $e^u(n_i, n_j)$ with same label of $e^u(n_j, n_k)$ is added to $\mathcal{T}_2$. The $e^u(n_j, n_k)$ is added to $\mathcal{N}^{selected}$ and the n_i is added to $\mathcal{N}_{next}^{ring}$. If n_j is in $\mathcal{N}^{selected}$, n_j is added to $\mathcal{N}_{next}^{ring}$. After all n_i are enumerated, The $\mathcal{N}^{ring}$ is added to $\mathcal{N}_{visited}$, the $\mathcal{N}_{next}^{ring}$ becomes the new $\mathcal{N}^{ring}$. After $\mathcal{N}^{ring}$ is empty, items in M^{do}, of which exist in $\mathcal{T}_2$, are added in M^{ro}.

Step 4. Add edges back. The algorithm adds the edge back into $\mathcal{T}_2$, which is in $\mathcal{E}_1^{deleted}$ or $\mathcal{E}_2^{deleted}$, and the endpoints are both in $\mathcal{T}_2$. The final tree is the reshaped ontology $\mathcal{G}^{ro}$.

4.4 Knowledge Graph Construction

The *KG Construction* module takes the reshaped ontology $\mathcal{G}^{ro}$, the *Raw-to-RO Mapping* M^{ro} and the *Welding Raw Data* as inputs, and generates a series corresponding *Welding KG*. We enumerate all class nodes in $\mathcal{G}^{ro}$. For each node and its datatype property edges, we find the primary keys for node and attributes for the edge respectively in the mapped tables and attributes in *Welding Raw Data* via M^{ro}, and create an entity for each key, and create datatype properties for each such edge. Next, we enumerate all object property edges in $\mathcal{G}^{ro}$, find the mapped foreign keys in the *Welding Raw Data* via M^{ro}, and create links (object properties) between the entity represented by the primary key and the entity represented by the foreign key. An small excerpt is shown in Fig. 5e,

which shows the knowledge graphs constructed based on $\mathcal{G}^{ro}$ as the schema has zero dummy nodes.

5 Evaluation

This section includes a preliminary user study and a system evaluation that evaluate our system from the user view and system view, respectively.

5.1 Preliminary User Study

Participants. We deployed our system with tasks and questionnaires on a Bosch environment and received a number of results. The participants (Table 1) include Bosch welding experts, engineers, welding, and production, and additionally software engineers and data scientists. They need to input their age, occupation, education and skills for semantic web, query, and welding, ranging from 0 (no knowledge), to 5 (experts).

Algorithm 3: TreeCollapse

Input: $\mathcal{T}_1,\ R,\ M^{do},\ U$
Output: $\mathcal{T}_2,\ \mathcal{E}_2^{deleted},\ M^{ro}$

Initialisation: $\mathcal{N}^{ring} \leftarrow \{n_m\},\ \mathcal{T}_2 \leftarrow \mathcal{T}_1,\ \mathcal{E}_2^{deleted} \leftarrow \{\}$
$\mathcal{N}^{selected} \leftarrow$ GetNodes$(R,\ M^{do}) \cup$ ReadUserInfo$(U) \cup$ IdentifyID$(\mathcal{G}^{rs})$
while $\mathcal{N}^{ring} \neq \varnothing$ **do**
  $\mathcal{N}_{next}^{ring} \leftarrow \{\}$
  foreach $n_i \in \mathcal{N}^{ring}$ **do**
    foreach $e^u(n_i, n_j) \in \mathcal{T}_1$ **do**
      if $n_j \notin \mathcal{N}^{selected}$ **then**
        $\mathcal{T}_2 := \mathcal{T}_2 \setminus \{n_j, e^u(n_i, n_j)\}$
        $\mathcal{E}_2^{deleted} := \mathcal{E}_2^{deleted} \cup \{e^u(n_i, n_j)\}$
        if $e^u(n_j, n_k) \in \mathcal{T}_1$ **then**
          $\mathcal{T}_2 := \mathcal{T}_2 \setminus \{e^u(n_j, n_k)\}$
          $\mathcal{T}_2 := \mathcal{T}_2 \cup \{e^u(n_i, n_k)\}$, where $e^u(n_i, n_k)$ adopts the label of $e^u(n_j, n_k)$
          $\mathcal{E}^{deleted} := \mathcal{E}^{deleted} \cup \{e^u(n_j, n_k)\}$
          $\mathcal{N}_{next}^{ring} := \mathcal{N}_{next}^{ring} \cup \{n_i\}$
      else
        $\mathcal{N}_{next}^{ring} := \mathcal{N}_{next}^{ring} \cup \{n_j\}$
  $\mathcal{N}^{ring} \leftarrow \mathcal{N}_{next}^{ring}$
$M^{ro} \leftarrow MappingGeneration(T2, M^{do})$

Tasks. We selected 7 tasks (Table 2) that should reach a balance between testing the system and maintaining a controllable scope. The tasks include two types: Type 1, to input user information for ontology reshaping and Type 2, to select one query from four options (only one option is correct) to perform data inspection or diagnostics in

Table 1. User profiles in the user study

#	**Age**	Occupation	Education	Sem. Web	Query	Welding skills
P1	28	R&D Engineer	MSc	2	2	3
P2	29	R&D Engineer	MSc	2	1	3
P3	29	Welding Engineer	MSc	1	0	3
P4	41	Senior Welding Expert	MSc	0	0	5
P5	45	Welding Engineer	MSc	0	0	4
P6	25	Welding Engineer	BSc	0	0	4
P7	42	Software Engineer	BSc	3	2	2
P8	39	Production Engineer	BSc	0	0	3
P9	23	Data Scientist	MSc	2	2	2
P10	44	Data Scientist	PhD	2	1	2

Table 2. Tasks and type in the user study

#	Tasks	Type
T1	Select "RSWOperation" as the main node	Type 1
T2	Mark "SheetComponent1" as a table node	Type 1
T3	Create a new table node "SheetCombination"	Type 1
T4	Inspect operation curves on KG^{ro}	Type 2
T5	Inspect operation curves on KG^{do}	Type 2
T6	Detect abnormal welding operations KG^{ro}	Type 2
T7	Detect abnormal welding operations KG^{do}	Type 2

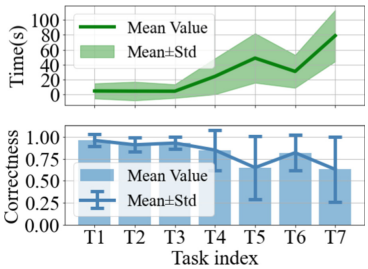

Fig. 6. Time/correctness for tasks

the knowledge graph (KG^{ro}) with the reshaped knowledge graph schema and in the knowledge graph (KG^{do}) with the domain ontology as the schema. Type 1 measure the usability of using our ontology reshaping system, and Type 2 compares users' perception of querying knowledge graphs with and without the ontology reshaping. Specifically, Type 1 has three tasks: T1, select the main node; T2, mark an attribute to table node in R; T3, create a new table node in R. Type 2 has four tasks: T4, select a query to inspect operation curves in KG^{ro}; T5, do the same on T4 in the KG^{do}; T6, select a query to detect abnormal welding operations (exceeding tolerance limit) in the KG^{ro}; T7, do the same on T6 in the KG^{do}.

Workflow of the User Study. For the user study, we first give the participants a short introduction with background knowledge, including basics of semantic technology like ontology, knowledge graph construction, and SPARQL query. Then, we explain them some relevant concepts of welding and the welding data (some users are not welding experts), present them visualisation of resistance welding domain ontology (Fig. 5). Then, we introduce them our tasks and how to use our GUI system. This introduction text is shown later constantly during the tasks. After that, the participants use the GUI system to perform the tasks. We record the time they use for each task, and the results of their actions stored in json. At the end, they answer a questionnaire (Table 3) with 12 questions that represent dimensions of their satisfaction about the system.

Results and Discussion. The results reflect the system usability (R3) [19] in efficiency (time used for tasks), effectiveness (correctness of user actions), and satisfaction. The recorded time (Fig. 6) show that the users need very limited time (average 28.0s) to perform the tasks, and thus the system is efficient. We compared the user results with a list of recommended results (we designed the tasks in a way so that the comparison is possible) and calculate the correctness. The results show (Fig. 6) that the correctness is always very high (average 82.1%) for the ontology reshaping tasks (Type 1) and for the query on the KG^{ro} (Type 2). The results also show that the correctness of selecting queries on KG^{ro} is higher than that on KG^{do} (T4>T5, T6>T7), which demonstrates the benefit of our ontology reshaping system.

The questionnaires (Table 3) subjectively evaluate the users' satisfaction about our system in four requirements (Sect. 2). From the aggregated scores, it can be seen that the users unanimously agree that our ontology reshaping system has good data coverage (R1); The knowledge coverage (R2) is scored 3.8, relatively good but has improvement room; The user-friendliness (R3) that covers connectivity, succinctness, simplicity and usability is also evaluated relatively high; The users are also quite satisfied with the system efficiency in terms of saving time and space (R4).

5.2 System Evaluation with Bosch Welding Dataset

We evaluated our system with OntoReshape$^+$ on 3 industrial datasets. In addition to baseline of using $\mathcal{G}^{do}$ as knowledge graph schema, we also compare with other 3 baselines.

Data Description. We now describe the datasets, including 3 industrial datasets D for knowledge graph construction and four inputs for ontology reshaping: 1 domain ontology $\mathcal{G}^{do}$, 3 relational schema R, 3 data to domain mappings M, and user information U.

Table 3. Questionnaires and scores for subjective evaluation. The scores range from 1 (disagree), 2 (fairly disagree), 3 (neutral), 4 (fairly agree), to 5 (agree). The column *Score* is aggregated by reversing the scores of negative questions (Q2, 4, 6, 8, 10, 12) and then computing the average (avg.) and standard deviation (std.) (`avg.±std.`)

#	Questions	Dimension	Score
Q1	I'm in general satisfied that KG^{ro} cover the data that I need.	Data coverage	4.31 ± 0.87
Q2	I found KG^{ro} miss some welding parameters that I need.		
Q3	I felt the knowledge represented by KG^{ro} is reasonable.	Knowledge coverage	4.63 ± 0.32
Q4	I thought KG^{ro} differs much from my understanding of welding.		
Q5	I like that in KG^{ro} all data can be reached from the main node.	User-friendliness	4.23 ± 0.71
Q6	I do not think that the queries over KG^{ro} become simpler.		
Q7	I found that it is great that KG^{ro} contains no dummy nodes.		
Q8	I hardly found KG^{ro} became simpler compared to KG^{do}.		
Q9	I found very confident using the system		
Q10	I needed to learn many things before I could use the system.		
Q11	I like that KG^{ro} saves storage space.	System efficiency	4.46 ± 0.33
Q12	I find it unnecessary the small amount of time saved by KG^{ro}.		

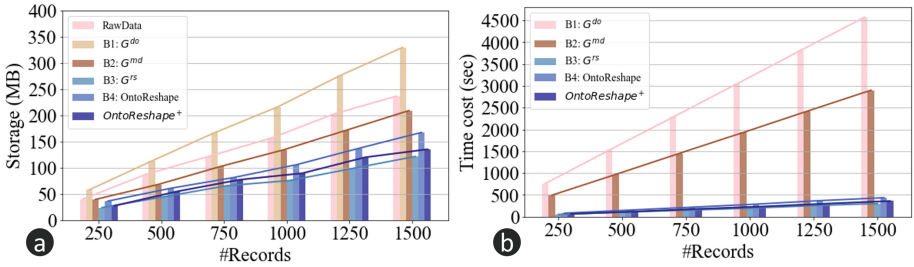

Fig. 7. Evaluation of space efficiency with storage taken by the constructed knowledge graph (a) and consumed time (b). The figure exemplifies the results obtained on D_1 since the results obtained on D_2 and D_3 are very similar.

Industrial Datasets D. Two production datasets D_1 and D_2 are collected from production lines in a factory of resistance spot welding in Germany. The third dataset D_3 is collected from a laboratory for welding research in Germany. After some processing they are transformed into relational tables. D_1 and D_2 contain 4 types of tables: they are the welding operation table, welding setting table, operation curve tables and reference curve tables. D_1 has 121 attributes and D_2 has 147 attributes. D_3 contains 5 types of tables: similar 4 types of tables as in D_1 and D_2 and an extra table of control parameter setting. D_3 has 160 attributes. For the evaluation purpose and a fair comparison, we select 1000 welding operations from each dataset.

Domain Ontology $\mathcal{G}^{rsw}$. The domain ontology models general knowledge of resistance spot welding. It is projected to a graph $\mathcal{G}^{rsw}$ with 210 class nodes and 191 datatype nodes, and 203 edges for object properties and 191 edges for datatype properties.

Relational Schema Rand Mappings M. The 3 relational schemata are information of table names and attribute names stored in csv. They are extracted from the three datasets D_1, D_2, and D_3. The 3 mappings map the table names and attribute names in the relational schemata to the domain ontology $\mathcal{G}^{rsw}$. These two help to generate the relational graphs $\mathcal{G}^{rs}$.

Table 4. The data coverage of all methods is 100%, and thus not displayed in the table. B: baseline.

Dataset	Evaluation metrics		Baseline methods/Ontology reshape methods				
			B1: $\mathcal{G}^{do}$	B2: $\mathcal{G}^{md}$	B3: $\mathcal{G}^{rs}$	B4: OntoReshape	OntoReshape$^+$
Production1 (D_1)	Knowledge coverage		1.00	0.36	0.21	0.42	0.74
	User-friendliness	Succinctness	0.38	0.46	1.00	1.00	1.00
		Connectivity	1.00	0.50	1.00	1.00	1.00
		Simplicity	0.13	0.17	0.33	0.33	0.33
Production2 (D_2)	Knowledge coverage		1.00	0.42	0.25	0.42	0.61
	User-friendliness	Succinctness	0.45	0.57	1.00	1.00	1.00
		Connectivity	1.00	0.50	1.00	1.00	1.00
		Simplicity	0.13	0.14	0.33	0.33	0.33
Lab data (D_3)	Knowledge coverage		1.00	0.45	0.27	0.42	0.81
	User-friendliness	Succinctness	0.51	0.59	1.00	1.00	1.00
		Connectivity	1.00	0.50	0.60	1.00	1.00
		Simplicity	0.13	0.17	0.33	0.33	0.33

Baselines. We compare the OntoReshape$^+$ algorithm with the traditional approach (Baseline 1, B1) that directly uses the domain ontology $\mathcal{G}^{do}$ as the schema for knowledge graph construction, in terms of the four requirements and 7 performance metrics (Sect. 3). In addition, we also compare with three other state-of-the-art baselines: Baseline 2 (B2) adopts an established ontology modularisation method [13,23] and uses the graph $\mathcal{G}^{md}$ projected from the modular ontology as the knowledge graph schema, which is computed with a signature of all table and attribute names in R; Baseline 3 (B3) uses the relational graph $\mathcal{G}^{rs}$ as the knowledge graph schema, which is trivially transformed from the relational schema R and the mapping M^{do}; Baseline 4 (B4) is a previous work of ontology reshaping [60].

Results and Discussion. We now discuss the performance of OntoReshape$^+$ in terms of the 4 requirements. We show the results evaluated in Table 4 and Fig. 7. We first look at D_1. It can be seen from Fig. 7a that our OntoReshape$^+$ outperforms the RawData, B1, B2 significantly in terms of the storage space (system efficiency R4), fairly better than B4, and slightly worse but comparable to B3. In terms of time efficiency (Fig. 7b), OntoReshape$^+$ significantly outperforms B1 and B2, while achieving comparable performance with respect to B3, B4.

All approaches have 100% data coverage (R1). Thus it is not displayed in the table. In terms of knowledge coverage (R2), it can be seem that OntoReshape$^+$ outperforms B2-B4 significantly, which means OntoReshape$^+$ keep the most knowledge of the domain ontology. It of course cannot beat B1 because B1 directly uses $\mathcal{G}^{do}$ as the knowledge graph schema, but B1 suffers substantially in terms of the later two metrics. The user-friendliness (R3) is decomposed to three metrics. OntoReshape$^+$ outperforms B1 and B2, and is equally good as B3 and B4 concerning succinctness. In respect to connectivity B3 is the worst and the others are equally good. As to simplicity, OntoReshape$^+$ outperforms B1, B2 and B4 and is equally good as B3. Thus, OntoReshape$^+$ either beats the baselines or is equally good as some. Regarding efficiency (R4), OntoReshape$^+$

saves time and space for knowledge graph generation when compared to B1, B2, and is comparable to B3 and B4. When looking at D_2 and D_3, it can be seem that the results are quite consistent across the datasets.

In summary, baselines B1, B2, B3 all are too focused either on knowledge coverage or data coverage. B4 and OntoReshape$^+$ are a balance between them, but OntoReshape$^+$ outperforms B4 in knowledge coverage and is comparable in other requirements.

6 Related Work

Knowledge graphs provide semantically structured information that can be interpreted by computing machines [49,62] and are widely used in industries [11,18,37,50]. The methods for knowledge graph construction have also been studied in many works [12,21,33], with focus on the rule-based approach [15], the combination of rule-based and similarity-based approach [29], the connection of data silos methods [18]. RDF lifting and lowering [2]. Commercial tools like OpenRefine [46] and OntoRefine [10] can transfer XML or tabular data to knowledge graphs or generate RML [1] and SPARQL [32]. Yet, they do not provide docking interface to our ML Mapping Reasoner/Annotator that reasons over domain ontologies, mappings and ML ontology.

The problem of transforming a bigger ontology to a smaller ontology of the same domain is often referred to as ontology modularisation [4–7,31] and ontology summarisation [35,36,48]. Most of them focus on the problem of selecting a subset of the ontology that is interesting for the users [30], but they still cannot avoid dummy entities. Works on ontology reengineering [42,43] also talked about reuse/adjustment of ontologies, they do not focus on the challenge of creating an ontology that reflect data specificities.

Previous work on ontology evolution [13] did not focus on the data coverage requirement. Our previous work on ontology reshaping [60] insufficiently address the knowledge coverage. Works on ontology bootstrapping [8,24,25] attempt to align ontologies with relational data schemata by automatically computing mappings between the ontologies and the data schemata, but the ontologies in these work only serve as a vocabulary for computing the mapping and new ontologies. Not much information from the original ontologies are retained.

In summary, past works insufficiently addressed the requirements R1-R4. Thus, we propose our work that can better address them overall.

7 Conclusion and Outlook

This work addresses the challenge of sparse knowledge graphs with many dummy nodes when domain ontologies that reflect general knowledge are directly used as the knowledge graph schemata. To this end, we proposed the ontology reshaping system and the algorithm OntoReshape$^+$. We evaluated the approach with a user study and a system evaluation in terms of four requirements, which shows promising results.

Our system is currently deployed in our Bosch evaluation environment, and we are considering to push it further into a more advanced and strict evaluation phase of

production that runs in real-time. To show the benefits, we also plan to demonstrate our knowledge graph solution with more users and more use cases. In the future, we plan to study the compatibility between domain ontologies and knowledge graph schemata, i.e. to ensure that the semantics of the domain is respected in the smaller ontology.

Acknowledgements. The work was partially supported by the H2020 projects Dome 4.0 (Grant Agreement No. 953163), OntoCommons (Grant Agreement No. 958371), and DataCloud (Grant Agreement No. 101016835) and the SIRIUS Centre, Norwegian Research Council project number 237898.

References

1. Arenas-Guerrero, J., et al.: Knowledge graph construction with R2RML and RML: an ETL system-based overview (2021)
2. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. J. Data Semant. **1**(3), 147–185 (2012)
3. Celik, O., Zhou, D., Li, G., Becker, P., Neumann, G.: Specializing versatile skill libraries using local mixture of experts. In: Conference on Robot Learning, pp. 1423–1433. PMLR (2022)
4. Chen, J., Ludwig, M., Ma, Y., Walther, D.: Zooming in on ontologies: minimal modules and best excerpts. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 173–189. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_11
5. Chen, J., Ludwig, M., Ma, Y., Walther, D.: Computing minimal projection modules for $\mathcal{ELH}^r$-terminologies. In: Calimeri, F., Leone, N., Manna, M. (eds.) JELIA 2019. LNCS (LNAI), vol. 11468, pp. 355–370. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19570-0_23
6. Chen, J., Ludwig, M., Walther, D.: On computing minimal $\mathcal{EL}$-subsumption modules. In: Proceedings of WOMoCoE 2016. CEUR-WS.org (2016)
7. Chen, J., Ludwig, M., Walther, D.: Computing minimal subsumption modules of ontologies. In: Proceedings of GCAI 2018, pp. 41–53. EasyChair (2018)
8. Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with APFEL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 186–200. Springer, Heidelberg (2005). https://doi.org/10.1007/11574620_16
9. Fan, M., Zhou, Q., Chang, E., Zheng, F.: Transition-based knowledge graph embedding with relational mapping properties. In: Proceedings of the 28th Pacific Asia Conference on Language, Information And Computing, pp. 328–337 (2014)
10. Fiorelli, M., Stellato, A.: Lifting tabular data to RDF: a survey. Metadata Semant. Res. **1355**, 85 (2021)
11. Garofalo, M., Pellegrino, M.A., Altabba, A., Cochez, M.: Leveraging knowledge graph embedding techniques for industry 4.0 use cases. In: Cyber Defence in Industry 4.0 Systems and Related Logistics and IT Infrastructures, pp. 10–26. IOS Press (2018)
12. Goodwin, T., Harabagiu, S.M.: Automatic generation of a qualified medical knowledge graph and its usage for retrieving patient cohorts from electronic medical records. In: 2013 IEEE Seventh International Conference on Semantic Computing, pp. 363–370. IEEE (2013)
13. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: theory and practice. J. Artif. Intell. Res. **31**, 273–318 (2008)
14. Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 1–17. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3_0

15. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Rule-driven inconsistency resolution for knowledge graph generation rules. Semant. Web **10**(6), 1071–1086 (2019)
16. Hogan, A., Arenas, M., Mallea, A., Polleres, A.: Everything you always wanted to know about blank nodes. J. Web Semant. **27**, 42–69 (2014)
17. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. (CSUR) **54**(4), 1–37 (2021)
18. Hubauer, T., Lamparter, S., Haase, P., Herzig, D.M.: Use cases of the industrial knowledge graph at siemens. In: International Semantic Web Conference (P&D/Industry/BlueSky) (2018)
19. ISO, C.: 9241–11.3. Part II: Guidance on specifying and measuring usability. ISO 9241 Ergonomic Requirements for Office Work With Visual Display Terminals (VDTs) (1993)
20. ITU: Recommendation ITU - T Y.2060: Overview of the Internet of Things. Technical report, International Telecommunication Union
21. Jain, N.: Domain-specific knowledge graph construction for semantic analysis. In: Harth, A., Presutti, V., Troncy, R., Acosta, M., Polleres, A., Fernández, J.D., Xavier Parreira, J., Hartig, O., Hose, K., Cochez, M. (eds.) ESWC 2020. LNCS, vol. 12124, pp. 250–260. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62327-2_40
22. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 687–696 (2015)
23. Jiménez-Ruiz, E., Grau, B.C., Sattler, U., Schneider, T., Berlanga, R.: Safe and economic re-use of ontologies: a logic-based methodology and tool support. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 185–199. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68234-9_16
24. Jiménez-Ruiz, E., et al.: BootOX: bootstrapping OWL 2 ontologies and R2RML mappings from relational databases. In: International Semantic Web Conference (Posters & Demos) (2015)
25. Jiménez-Ruiz, E., et al.: BootOX: practical mapping of RDBs to OWL 2. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 113–132. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_7
26. Kagermann, H.: Change through digitization—value creation in the age of industry 4.0. In: Albach, H., Meffert, H., Pinkwart, A., Reichwald, R. (eds.) Management of Permanent Change, pp. 23–45. Springer, Wiesbaden (2015). https://doi.org/10.1007/978-3-658-05014-6_2
27. Kaiya, H., Saeki, M.: Using domain ontology as domain knowledge for requirements elicitation. In: 14th IEEE International Requirements Engineering Conference (RE 2006), pp. 189–198. IEEE (2006)
28. Kartsaklis, D., Pilehvar, M.T., Collier, N.: Mapping text to knowledge graph entities using multi-sense LSTMs. arXiv preprint arXiv:1808.07724 (2018)
29. Kertkeidkachorn, N., Ichise, R.: An automatic knowledge graph creation framework from natural language text. IEICE Trans. Inf. Syst. **101**(1), 90–98 (2018)
30. Konev, B., Lutz, C., Walther, D., Wolter, F.: Model-theoretic inseparability and modularity of description logic ontologies. Artif. Intell. **203**, 66–103 (2013)
31. Koopmann, P., Chen, J.: Deductive module extraction for expressive description logics. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-2020, pp. 1636–1643. International Joint Conferences on Artificial Intelligence Organization, July 2020
32. Lefrançois, M., Zimmermann, A., Bakerally, N.: A SPARQL extension for generating RDF from heterogeneous formats. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 35–50. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58068-5_3

33. Liebig, T., Maisenbacher, A., Opitz, M., Seyler, J.R., Sudra, G., Wissmann, J.: Building a Knowledge Graph for Products and Solutions in the Automation Industry (2019)
34. Naab, C., Zheng, Z.: Application of the unscented Kalman filter in position estimation a case study on a robot for precise positioning. Robot. Auton. Syst. **147**, 103904 (2022)
35. Ozacar, T., Ozturk, O.: Karyon: a scalable and easy to integrate ontology summarisation framework. J. Inf. Sci. **47**(2), 255–268 (2021)
36. Pouriyeh, S., et al.: Ontology summarization: graph-based methods and beyond. Int. J. Semant. Comput. **13**(2), 259–283 (2019). https://doi.org/10.1142/S1793351X19300012
37. Ringsquandl, M., et al.: On event-driven knowledge graph completion in digital factories. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 1676–1681. IEEE (2017)
38. Roman, D.: The euBusiness graph ontology: a lightweight ontology for harmonizing basic company information. Semant. Web **13**(1), 41–68 (2022)
39. Ryen, V., Soylu, A., Roman, D.: Building semantic knowledge graphs from (semi-)structured data: a review. Future Internet **14**(5), 129 (2022)
40. Smith, B.: Ontology. In: The furniture of the world, pp. 47–68. Brill (2012)
41. Soylu, A., et al.: TheyBuyForYou platform and knowledge graph: expanding horizons in public procurement with open linked data. Semant. Web **13**(2), 265–291 (2022)
42. Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The NeOn methodology for ontology engineering. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 9–34. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24794-1_2
43. Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A.: Introduction: ontology engineering in a networked world. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 1–6. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24794-1_1
44. Svetashova, Y., et al.: Ontology-enhanced machine learning: a Bosch use case of welding quality monitoring. In: ISWC (2020)
45. Svetashova, Y., Zhou, B., Schmid, S., Pychynski, T., Kharlamov, E.: SemML: reusable ML for condition monitoring in discrete manufacturing. In: ISWC (Demos/Industry), vol. 2721, pp. 213–218 (2020)
46. Verborgh, R., De Wilde, M.: Using OpenRefine. Packt Publishing Ltd. (2013)
47. Yahya, M., et al.: Towards generalized welding ontology in line with ISO and knowledge graph construction. In: Paul, G., et al. (eds.) The Semantic Web: ESWC 2022 Satellite Events. ESWC 2022. LNCS, vol. 13384, pp. 83–88. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_16
48. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on RDF sentence graph. In: WWW, pp. 707–716. ACM (2007)
49. Zhao, Z., Han, S.K., So, I.M.: Architecture of knowledge graph construction techniques. Int. J. Pure Appl. Math. **118**(19), 1869–1883 (2018)
50. Zheng, P., Xia, L., Li, C., Li, X., Liu, B.: Towards self-x cognitive manufacturing network: an industrial knowledge graph-based multi-agent reinforcement learning approach. J. Manuf. Syst. **61**, 16–26 (2021)
51. Zheng, Z., et al.: Query-based industrial analytics over knowledge graphs with ontology reshaping. In: Paul, G. et al. (eds.) The Semantic Web: ESWC 2022 Satellite Events. ESWC 2022. LNCS, vol. 13384, pp. 123–128. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_23
52. Zheng, Z., et al.: Executable knowledge graph for machine learning: a Bosch case for welding monitoring. In: ISWC (2022)
53. Zhou, B.: Machine Learning Methods for Product Quality Monitoring in Electric Resistance Welding. Ph.D. thesis, Karlsruhe Institute of Technology, Germany (2021)

54. Zhou, B., Pychynski, T., Reischl, M., Kharlamov, E., Mikut, R.: Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding. J. Intell. Manuf. **33**(4), 1139–1163 (2022)
55. Zhou, B., Svetashova, Y., Byeon, S., Pychynski, T., Mikut, R., Kharlamov, E.: Predicting quality of automated welding with machine learning and semantics: a Bosch case study. In: CIKM (2020)
56. Zhou, B., et al.: SemML: facilitating development of ML models for condition monitoring with semantics. J. Web Semant. **71**, 100664 (2021)
57. Zhou, B., Svetashova, Y., Pychynski, T., Kharlamov, E.: Semantic ML for manufacturing monitoring at Bosch. In: ISWC (Demos/Industry), vol. 2721, p. 398 (2020)
58. Zhou, B., et al.: The data value quest: a holistic semantic approach at Bosch. In: Paul, et al. (eds.) The Semantic Web: ESWC 2022 Satellite Events. ESWC 2022. LNCS, vol. 13384, pp. 287–290. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_42
59. Zhou, B., Zhou, D., Chen, J., Svetashova, Y., Cheng, G., Kharlamov, E.: Scaling usability of ML analytics with knowledge graphs: exemplified with a Bosch welding case. In: IJCKG (2021)
60. Zhou, D., Zhou, B., Chen, J., Cheng, G., Kostylev, E.V., Kharlamov, E.: Towards ontology reshaping for kg generation with user-in-the-loop: applied to Bosch welding. In: IJCKG (2021)
61. Zhou, D., et al.: Enhancing knowledge graph generation with ontology reshaping-Bosch case. In: Paul, et al. (eds.) The Semantic Web: ESWC 2022 Satellite Events. ESWC 2022. LNCS, vol. 13384, pp. 299–302. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_42
62. Zou, X.: A survey on application of knowledge graph. In: Journal of Physics: Conference Series, vol. 1487, p. 012016. IOP Publishing (2020)