Ontology-Compliant Knowledge Graphs

Zhangcheng Qiang[0000−0001−5977−6506]

School of Computing, Australian National University, Canberra, Australia
Category: Early Stage PhD, Email: qzc438@gmail.com

Abstract. Ontologies can act as a schema for constructing knowledge graphs (KGs), offering explainability, interoperability, and reusability. We explore *ontology-compliant* KGs, aiming to build both internal and external ontology compliance. We discuss key tasks in ontology compliance and introduce our novel term-matching algorithms. We also propose a *pattern-based compliance* approach and novel compliance metrics. The building sector is a case study to test the validity of ontology-compliant KGs. We recommend using ontology-compliant KGs to pursue automatic matching, alignment, and harmonisation of heterogeneous KGs.

Keywords: Ontology · Knowledge Graphs · Matching and Alignment

1 Introduction and Motivation

An ontology is typically used as the backbone for constructing a KG, building so-called *ontology-based KGs*. In this setting, the ontology and the KG are often treated as independent functional components. An ontology provides a knowledge-oriented graph schema (i.e., TBox), whereas a KG represents the corresponding data-driven instances (i.e., ABox). With the proliferation of KGs in real-world applications, problems arise when data in the KG is generated for different user requirements. The ontology is likely to be incompatible with the data in the KG because ABox assertions may extend or be incomplete with respect to the ontology. While ABox contents can be adapted to suit a TBox, for interoperability amongst independent TBoxes, an ABox that is compatible with a number of TBoxes may be needed. Such overarching TBoxes should support conversion and exchange for cross-KG harvesting and federated searches.

Fig. 1 illustrates three types of non-compliance between KG and its ontology. (1) The ABox in the KG only covers a small amount of TBox terminologies, and its ontology has many unused classes and properties. (2) The ABox in the KG contains more information than the TBox terminologies, and many terms in the KG cannot find appropriate classes and properties in its ontology. (3) In a combination of (1) and (2), the ABox in the KG and the TBox in the ontology are mismatched and overlapped on both sides. Many application tasks, for example, KG embedding and ontology learning, are hampered by non-compliance between KG and its ontology. When using KG embedding for ontology-based KGs, unused classes and properties in the ontology are noisy data. This results in inaccurate embeddings for KG terms. Ontology learning is the task of using KG to infer

ontology classes and properties. KG data is diverse in nature; thus, the new ontology classes and properties learnt from KG can be different according to KG instances. These task-specific ontology classes and properties may violate the FAIR (i.e., Findability, Accessibility, Interoperability, and Reusability) principle and can be challenging to map and integrate into the original ontology.

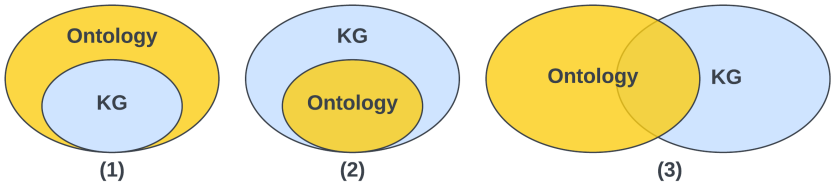


Fig. 1. Non-compliance between KG and its ontology.

Current work mainly focuses on either schema matching (e.g., TBox-TBox compliance) or instance matching (e.g., ABox-ABox compliance). TBox-ABox compliance is underexplored. While it is questionable whether the TBox is always compliant with the ABox, ontology-based KGs assume they are compliant by nature (excluding or ignoring the three types of non-compliance). For this reason, there is rarely a compliance check in popular KG and ontology modelling libraries or editors (e.g., RDFLib [2], Protégé [11], and TopBraid Composer [16]). Even within the Ontology Alignment Evaluation Initiative (OAEI) [12], to the best of our knowledge, we cannot find tools available to track these mismatches and overlaps between ABox in the KG and TBox in its corresponding ontology.

2 State of the Art

Ontology-Based KGs describe the traditional design for using ontologies with KGs, whereby the ontology serves as the schema for the KGs. KGs are generated using the classes and properties pre-defined in the ontology. In this setting, ontology-based KGs assume the ontology has established well-defined concepts, taxonomies, relationships, and domain axioms. Compared with ontology-less KGs, ontology-based KGs provide more formal representations for data understanding, organisation, and integration. They also enable improved logical reasoning, empowered reuse, and enhanced interoperability between different downstream applications. However, a complete ontology is almost impossible. Ontology is built on the Open World Assumption (OWA). We cannot assume an ontology has captured all domain concepts because the absence of concepts is not non-existence (i.e., these concepts may exist in other ontologies). A "well-defined" ontology also requires solid verification and validation. There is no gold standard for dealing with individual differences among opposing viewpoints.

Ontology-Aware KGs follow a reverse way of using ontologies with KGs. Conceptual components learnt from KGs are used to build or evolve the original ontology. The paradigm of ontology-aware KGs assumes the KG data is noiseless. There are two directions for constructing ontology-aware KGs. (1) Ontology reshaping is applied to data in the KG only covers part of the concepts in the ontology. The goal of ontology reshaping is to create a data-oriented local schemata that preserves the domain ontology knowledge while removing unused

nodes [18,19]. (2) Ontology enrichment is used where data exists in the KG that is not covered by the ontology. In this case, the new concepts and relationships learnt from the KG are registered as new classes and properties in the ontology [9,17]. While ontology-aware KGs achieve partial compliance between the KG and its ontology, they still have some limitations. Concepts that have been locally reshaped and redefined are task-specific, with limited sharing and reusing capabilities. Moreover, ontology-aware KGs cannot track the poly-ontological representation of KGs as they only match one KG to its corresponding ontology.

3 Problem Statement and Contributions

In the real world, KGs and ontologies are mostly incomplete. Neither ontology-based KGs nor ontology-aware KGs could fully handle the compliance issue between KGs and ontologies. We plan to propose **Ontology-Compliant KGs** to fill this gap. "Compliant" here has two aspects: (1) The terms used in KG are in line with the definition provided by the ontology. Mismatched terms in the KG are replaced with the most relevant classes and properties defined in the original ontology. (2) The size of the ontology complies with the information coverage of the KG. There are no unused classes or properties. In this work, we also extend this definition to be ontology compliant across KGs. Joint learning, vector embedding methods, and pattern-based engineering concepts are employed to achieve the goal of both internal and external compliance between KGs and ontologies. Fig. 2 shows the difference between ontology-compliant KGs and the other two types of ontology-related KGs. While ontology-based and ontology-aware KGs only consider a one-way connection, in ontology-compliant KGs, the link between ontology and KG is bidirectional and can be bridged by their patterns (details are described in Section 6).

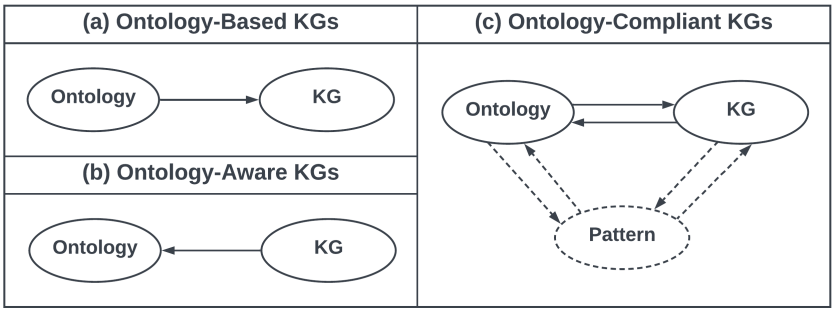


Fig. 2. The difference between ontology-compliant KGs and the other two types.

Hypothesis Ontology-compliant KGs have the following unique features:

H1 Given an ontology and a baseline KG, ontology-compliant KGs can eliminate the unused classes and properties in the ontology and reduce misdefined terms in the KG (interpreted as Ontology Compliance *within* KG).

H2 Given a set of ontologies and a baseline KG, ontology-compliant KGs allow automatic transmission from one schema to another (interpreted as Ontology Compliance *over* KGs).

H3 Given a set of ontologies and a baseline KG, ontology-compliant KGs allow different ontology fragment representations via a pattern-based approach. These ontology fragments are provided with multiple criteria for integration, evaluation, and selection (interpreted as *Pattern-based* Compliance).

Research Questions We formulate the related research questions:

RQ1 (wrt H1) How to reconstruct ontology-based KGs into ontology-compliant KGs, while retaining critical information and primary inference capability but eliminating unused nodes and reducing misdefined nodes?

RQ2 (wrt H2) How to enable schema-free KGs that can be compliant with multiple ontologies, using the ontology-compliant KGs to automate and optimise the ontology alignment and matching process?

RQ3 (wrt H3) How to select the most compliant set of ontology fragments for KGs? How to capture the different ontology fragment representations using a pattern-based approach, and evaluate them according to sound criteria from different useful perspectives?

4 Research Methodology and Approach

Details of preliminary results based on the research methodology and approach are described in Section 6. This PhD aims to define a generalised approach to constructing ontology-compliant KGs. We propose to classify three stages of compliance in ontology-compliant KGs, namely (1) Ontology Compliance *within* KG, (2) Ontology Compliance *over* KGs, and (3) *Pattern-Based* Compliance. In each stage, we intend to address the hypothesis and its related research question. The "building domain" is selected as a case study. We design, implement, and evaluate our matching algorithms, and analyse their matching performance in terms of different building use cases and various application-level tasks.

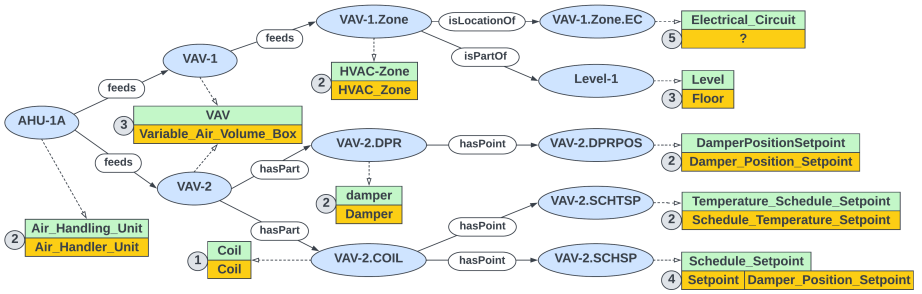
5 Evaluation Plan: A Case Study in the Building Sector

In the context of Industry 5.0 and the Internet of Things (IoT), digitisation and automation are becoming emerging research areas in the building sector. While a number of building and building-related ontologies have been developed, data interoperability issues have become more apparent. Different building ontologies are developed and maintained by different institutions. These ontologies are modelled at multiple levels of abstraction for various purposes, and their definitions are frequently competing and overlapping. Proposed ontology-compliant KGs would potentially help with the unified vision of building ontologies, where the data in this domain has complexity and variety in concepts and relations.

6 Preliminary Results

6.1 Ontology Compliance within KG A KG and its ontology share all terms and topology. However, the concepts and properties defined in KG and ontology can be mismatched due to human errors, design choices, or changes in newer versions. Fig. 3 shows different types of node matching in a snippet of an air handling unit (AHU) system represented by a KG and its ontology Brick Schema [1]

(abbr. "Brick"). The concepts with green colours are KG classes, while the concepts with yellow colours are ontology classes. Different matching types and their examples are shown in the table below. These also applied to the property matching between KG and its ontology. We design Algorithm 1 for building ontology compliance within KG. It has two phases: (1) Entity Alignment and (2) Ontology Reconstruction. We first find non-compliant terms in the KG and replace them with the most relevant classes and properties in the original ontology, assigning a confidence score for each replacement. Then, we find all the related triples (including constraints and axioms) and restore the ontology hierarchies.



Legend
- KG Instance
- KG Concept
- Ontology Class
- Relation
- Type Link
- Matching Type

ID	Matching Type	Matching Sub-type	Example
1	Purely Matching	N/A	Coil vs. Coil
2	Heuristic Matching	Upper/Lower Case	damper vs. Damper
		Separator Used	HVAC-Zone vs. HVAC_Zone
		Camel/Snake Case	DamperPositionSetpoint vs. Damper_Position_Setpoint
		Word Order	Temperature_Schedule_Setpoint vs. Schedule_Temperature_Setpoint
		Stemming/Lemmatisation	Air_Handling_Unit vs. Air_Handler_Unit
3	Semantic Matching	Acronym	VAV vs. Variable_Air_Volume_Box
		Synonym	Level vs. Floor
4	Topological Matching	Ontology Subsumption	Schedule_Setpoint vs. Setpoint
		Centrality Measure	Schedule_Setpoint vs. Damper_Position_Setpoint (Incorrect in this case)
5	Purely Mismatching	N/A	Electrical_Circuit does not exist in Brick classes

Fig. 3. An example of ontology compliance within KG.

Evaluation The preliminary experiment uses the sample example from the Brick Schema official website. We synthesise a number of mismatched classes and properties with different types. The results in Table 1 show that reshaped ontologies can significantly reduce the original ontology size and increase the number of used and matched classes. We can also observe a trade-off between the confidence score and the level of matching applied. The confidence score slightly decreases when the level of matching increases. A potential reason is that Level 3 and Level 4 use learning-based approaches. While they are more powerful at discovering more pairs of matches, the confidence level of the matching accuracy highly depends on the models and methods used.

Table 1. Evaluation of algorithm for building ontology compliance within KG.

Type of Ontology & Matching Level	Used Entity	Matching Rate	Confidence
Original Ontology & Matching Lv. 1	0.52%	46.15%	100.00%
Reshaped Ontology & Matching Lv. 1	30.00%	46.15%	100.00%
Reshaped Ontology & Matching Lv. 2	38.46%	76.92%	97.50%
Reshaped Ontology & Matching Lv. 3	42.31%	84.62%	88.00%
Reshaped Ontology & Matching Lv. 4	46.15%	92.31%	78.33%

Algorithm 1 Building Ontology Compliance within KG

Input: Ontology-based KG, Ontology $Onto$
Output: Reshaped Onto $Onto_{RE}$, Confidence μ

```
/* Phase 1: Entity Alignment */
/* Find all terms in KG */
Concept Set Con_S, Relation Set Rel_S ← ∅
Con_S, Rel_S ← findConceptAndRelation(KG)
/* Find all classes and properties in Onto */
Class Set Cls_S, Property Set Pro_S ← ∅
Cls_S, Pro_S ← findClassAndProperty(Onto)
/* Divide each naming into a keyword set */
Con_S, Rel_S ← findKwd(Con_S, Rel_S)
Cls_S, Pro_S ← findKwd(Cls_S, Pro_S)
/* Match classes between KG and Onto */
for i ∈ Con_S do
  for j ∈ Cls_S do
    if purelyMatching(i, j) ≠ ∅ then
      μ_i ← 1
    else if heuristicMatching(i, j) ≠ ∅ then
      μ_i ← LevenshteinDistance
      Con_S → i ∧ Con_S ← j
    else if semanticMatching(i, j) ≠ ∅ then
      μ_i ← Similarity
      Con_S → i ∧ Con_S ← j
    else if topologicalMatching(i, j) ≠ ∅ then
      μ_i ← Accuracy
      Con_S → i ∧ Con_S ← j
    end if
  end for
end for
/* Match properties between KG and Onto */
for k ∈ Rel_S do
  for l ∈ Pro_S do
    /* Corresponding procedure applies to
    Rel_S and Pro_S */
  end for
end for
/* Calculate total confidence */
Total Confidence μ ← ∅
μ ← Avarage(μ_1, ..., μ_i, μ_1, ..., μ_k)
/* Phase 2: Ontology Reconstruction */
/* Find super-classes */
Onto_RE Class Set RE_Cls_S ← ∅
RE_Cls_S ← findSuperClasses(Con_S)
/* Find super-properties */
Onto_RE Property Set RE_Pro_S ← ∅
RE_Pro_S ← findSuperProperties(Rel_S)
/* Restore reshaped ontology */
Reshaped Ontology Onto_RE ← ∅
Onto_RE ← findTriples(RE_Cls_S, RE_Pro_S)

return Onto_RE, μ
```

6.2 Ontology Compliance over KGs It is often the case that a KG may be restructured to comply with one of several different ontologies, while preserving the KG's intended information content, as illustrated in Fig. 4.

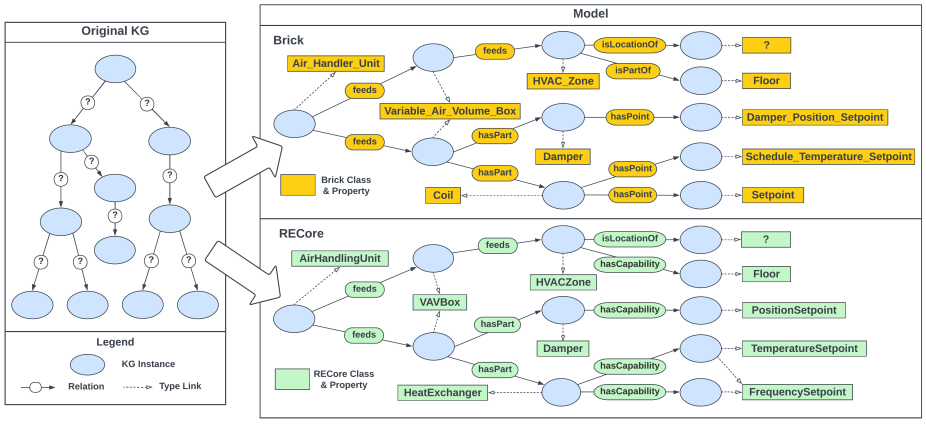


Fig. 4. An example of ontology compliance over KGs.

The key task is ontology alignment and matching. Embedding-based methods are prevalent for exploring the potential matching in graphs due to their conceptual simplicity and computational operability. However, there are several challenges when applying the embedding methods to ontology alignment and matching. Firstly, not all the classes and properties from the original ontology are useful for KG instance embedding. The unused classes and properties could be noise for matching. Secondly, the respective KGs have no connection with

each other. Embedding methods are based on random sampling, meaning the vector only represents the relative position of the nodes, and the vector number sets can be different. If two graphs are not tightly connected, the embedding results are most likely to be incorrect. Thirdly, the majority of the embedding methods are targeting graphs without schema. They focus more on topological matching rather than lexicographic matching. We employ our findings from ontology-compliant within KG to address the first challenge, and use their compliance as an intermediate link to connect two graphs - the second challenge, and facilitate lexicographical order - the third challenge. We design Algorithm 2 for building ontology compliance over KGs. It has three phases: (1) Build ontology compliance within each KG, (2) Match terms across ontologies, and (3) Match overlapping terms. Phases 1 and 2 also reuse the Algorithm 1.

Algorithm 2 Building Ontology Compliance over KGs

Input: KG_1 and related ontology $Onto_1$, KG_2 and related ontology $Onto_2$
Output: Matching Set $Match\_S(Onto_1, Onto_2)$, Confidence Set $\mu(\mu_{within}, \mu_{over})$
/* Phase 1: Compliance within KG */
$Onto_{RE_1}, \mu_1 = withinComp(KG_1, Onto_1)$
$Onto_{RE_2}, \mu_2 = withinComp(KG_2, Onto_2)$
$\mu_{within} \leftarrow \mu_1, \mu_2$
/* Phase 2: Match terms across Onto */
/* Create matching set */
Matching Set $Match\_S(Onto_1, Onto_2) \leftarrow \varnothing$
/* Follow same procedure in Algorithm 1 */
for $i \in Con\_Onto_{RE_1}, k \in Rel\_KG_1$ **do**
  for $j \in Con\_Onto_{RE_2}, l \in Rel\_KG_2$ **do**
    $Match\_S(Onto_1, Onto_2), \mu_{cmatch} \leftarrow$ Matched Con & Rel, $\mu_{over} \leftarrow \mu_{cmatch}$
  end for
end for
/* Phase 3: Match overlapping terms */
/* Set a vector space */
Vector space $VecSpace \leftarrow \varnothing$
/* Put KGs and Onto_RE */
for $i \in KG_1, Onto_{RE_1}, KG_2, Onto_{RE_2}$ **do**
  $VecSpace \leftarrow i$
end for
/* Put matched Onto set to build links */
$VecSpace \leftarrow Match\_S(Onto_1, Onto_2)$
/* Define embedding models */
Embedding Model $Model \leftarrow X2Vec.train()$
/* Define vector set */
Vector Set $Vec\_S \leftarrow Model.getEmbeddings()$
/* Add predict match for overlapping */
Predict Match $P \leftarrow \varnothing$
for Unmatched $U \notin Match\_S(Onto_1, Onto_2)$ **do**
  $P, \mu_{overlap} \leftarrow Vec\_S.getMostSimilar(U)$
  $Match\_S(Onto_1, Onto_2) \leftarrow (U, P)$
  $\mu_{over} \leftarrow \mu_{overlap}$
end for
/* Summarise total confidence */
Confidence Set $\mu \leftarrow \varnothing$
$\mu \leftarrow \mu_{within}$
$\mu \leftarrow \mu_{over}$

return $Match\_S(Onto_1, Onto_2)$, $\mu(\mu_{within}, \mu_{over})$

Evaluation The preliminary experiment is set to predict the similarity of two overlapping properties, brick:hasPoint in Brick Schema [1] (abbr. "Brick") and core:hasCapability in RealEstateCore [8] (abbr. "RECore"). The ground truth is that the meanings of these two properties are very similar. Their different names are due to their different views on how building points are embedded in the building. brick:hasPoint states that the building points are the measurable data points installed in the building, whereas core:hasCapability stands for the building points are the capabilities provided by the building to produce and ingest data. We employ three different vector embedding models to evaluate the top-k searches. Exp.1 uses the traditional KG embedding without ontology compliance, and Exp.2 uses our proposed compliance algorithm. Table 2 shows the results of the comparison in a test run. We can see Exp.2 outperforms Exp.1 in all three sample embedding models, particularly in top-1 and top-3 searches.

6.3 Pattern-Based Compliance Ontology can be decomposed into smaller ontology fragments. For example, the concepts defined in Brick Schema (abbr.

Table 2. Evaluation of algorithm for building ontology compliance over KGs.

@k	DeepWalk [13]		Node2Vec [6]		Struc2Vec [15]	
	Exp.1	Exp.2	Exp.1	Exp.2	Exp.1	Exp.2
1	$0.03 \pm 0.17\%$	$25.86 \pm 4.48\%$	$0.04 \pm 0.20\%$	$15.32 \pm 3.70\%$	$96.88 \pm 1.52\%$	$99.98 \pm 0.14\%$
3	$1.36 \pm 1.07\%$	$57.49 \pm 4.86\%$	$13.4 \pm 3.35\%$	$56.65 \pm 5.28\%$	$100 \pm 0.00\%$	$100 \pm 0.00\%$
5	$7.58 \pm 2.57\%$	$75.51 \pm 4.17\%$	$66.54 \pm 4.85\%$	$84.86 \pm 3.63\%$	$100 \pm 0.00\%$	$100 \pm 0.00\%$

"Brick") can be decomposed into three high-level abstraction fragments: Spaces (i.e., brick:Location), Building Equipment and Systems (i.e., brick:Equipment and brick:System), and Building Points (i.e., brick:Point). For each fragment, they can be replaced with the same concepts defined in other building ontologies, such as RealEstateCore [8] (abbr. "RECore") and Project Haystack [10] (abbr. "Haystack"), or building-related domain ontologies, such as BOT [14] for spatial information, SAREF [4] for equipment and systems, and SSN [3]/SOSA [7] for building points. Fig. 5 demonstrates an example of KG represented by different combinations of building and related domain ontologies. These new ontology fragments can represent the same information as the original ontology, but they can have different numbers of classes, properties, and hierarchies. If we consider ontology fragments, the problem of ontology compliance becomes more complex.

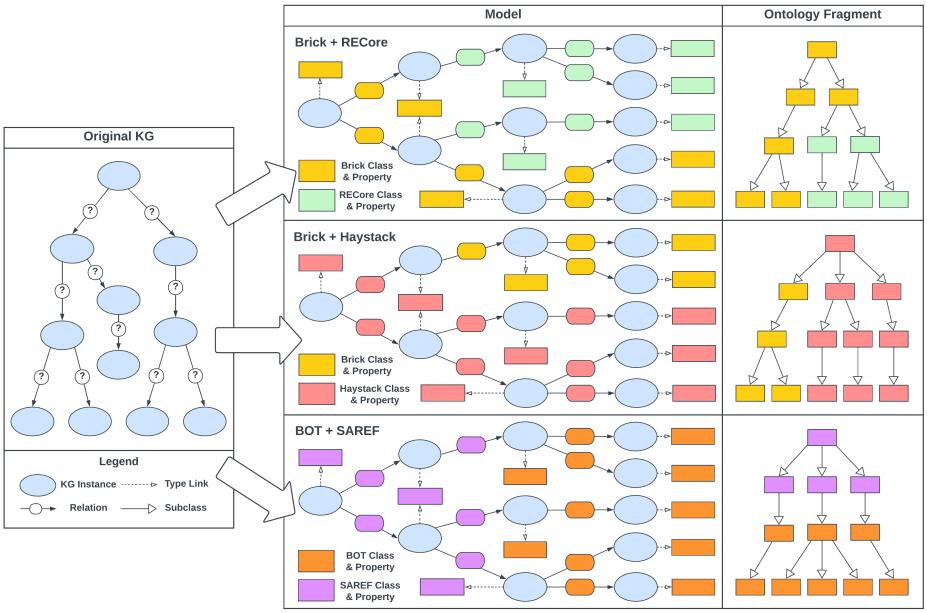


Fig. 5. An example of pattern-based compliance.

Fig. 6 shows the architecture of building pattern-based compliance. It has three main components, namely (1) Pattern Cognition, (2) Pattern Recognition, and (3) Pattern Optimisation. The basic idea is to extract the concepts, relationships, and constraints from the ontology-compliant KG. Each of them goes through a learning and matching process to find their patterns. Then, we inte-

grate and align the same or similar patterns, and use these generic patterns to reconstruct new ontology fragments.

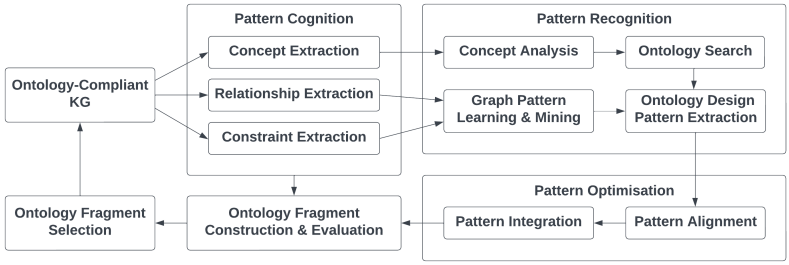


Fig. 6. The architecture of building pattern-based compliance.

Evaluation Ontology fragments may have different numbers of namespaces, levels of abstraction, concept coverage, depth of the class hierarchy, completeness and expressiveness, and performance metrics. Liebig's law [5] is used for ontology fragment construction, evaluation, and selection. The multi-criteria selection depends on the minimum criteria being satisfied. We also introduce a joint-learning approach to evaluate the performance of ontology fragments. An example is shown in Fig. 7. Ontology Fragment 1, 2, and 3 are generated from the same KG. We fit them into the embedding model and perform the classification task according to the original KG. Based on different levels of abstraction, the classification accuracy of the KG embedding decreases at different rates. Fragment 1 and 2 have higher accuracy in Level 1 and Level 2 abstractions, but they have a significant drop in Level 3. By contrast, Fragment 3 decreases gradually at all levels of abstraction.

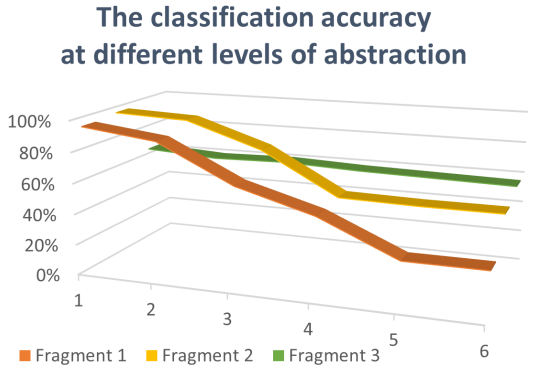


Fig. 7. A joint approach to evaluate the performance of ontology fragments.

Preliminary results may have slight differences across different platforms and library versions. The code implementation is available at `https://github.com/qzc438/ontology-compliant-kgs` (access will be made available on request).

7 Conclusions

In this paper, we present a new concept of ontology-compliant KGs, showing promising results in matching and aligning ontologies within KG and over KGs.

We also illustrate our design for advanced pattern-based compliance. Further work will focus on justifying the results with the capability to allow ontology compliance on large-scale KGs, and implementing pattern-based compliance in a comprehensive framework that enables automatic ontology fragment integration, evaluation, and selection for real-world application-level KGs.

Acknowledgements *This project is supervised by Kerry Taylor, Sergio Rodríguez Méndez, Subbu Sethuvenkatraman, Qing Wang, and Armin Haller. The author also thanks program mentor Maria Maleshkova for providing valuable feedback.*

References

1. Balaji, B., et al.: Brick: Towards a Unified Metadata Schema For Buildings. In: BuildSys '16. p. 41–50 (2016). https://doi.org/10.1145/2993422.2993577
2. Boettiger, C.: RDFLib. https://github.com/RDFLib/rdflib (2018)
3. Compton, M., et al.: The SSN ontology of the W3C semantic sensor network incubator group. Journal of Web Semantics **17**, 25–32 (2012). https://doi.org/10.1016/j.websem.2012.05.003
4. Daniele, L., et al.: Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In: FOMI 2022. pp. 100–112 (2015). https://doi.org/10.1007/978-3-319-21545-7_9
5. de Baar, H.: von Liebig's law of the minimum and plankton ecology (1899–1991). Progress in Oceanography **33**(4), 347–386 (1994). https://doi.org/10.1016/0079-6611(94)90022-1
6. Grover, A., Leskovec, J.: Node2vec: Scalable Feature Learning for Networks. In: KDD '16. p. 855–864 (2016). https://doi.org/10.1145/2939672.2939754
7. Haller, A., et al.: The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web **10**(1), 9–32 (2019). https://doi.org/10.3233/SW-180320
8. Hammar, K., et al.: The RealEstateCore Ontology. In: The Semantic Web – ISWC 2019. pp. 130–145 (2019). https://doi.org/10.1007/978-3-030-30796-7_9
9. Hurlburt, G.F.: The Knowledge Graph as an Ontological Framework. IT Professional **23**(4), 14–18 (2021). https://doi.org/10.1109/MITP.2021.3086918
10. John, J., Gowan, M.: Project Haystack Data Standards. In: Energy and Analytics. pp. 237–243 (2020). https://doi.org/10.1201/9781003151944-16
11. Musen, Mark A: Protégé. https://protege.stanford.edu (2020)
12. Ontology Alignment Evaluation Initiative. http://oaei.ontologymatching.org/
13. Perozzi, B., et al.: DeepWalk: Online Learning of Social Representations. In: KDD '14. p. 701–710 (2014). https://doi.org/10.1145/2623330.2623732
14. Rasmussen, M.H., et al.: BOT: The building topology ontology of the W3C linked building data group. Semantic Web **12**, 143–161 (2021). https://doi.org/10.3233/SW-200385
15. Ribeiro, L.F., et al.: Struc2vec: Learning Node Representations from Structural Identity. In: KDD '17. p. 385–394 (2017). https://doi.org/10.1145/3097983.3098061
16. TopQuadrant, Inc.: TopBraid Composer. https://www.topquadrant.com (2022)
17. Zhao, L., et al.: Learning Ontology Axioms over Knowledge Graphs via Representation Learning. In: ISWC 2019. pp. 57–60 (2019)
18. Zhou, D., et al.: Enhancing Knowledge Graph Generation with Ontology Reshaping – Bosch Case. ESWC (Demos/Industry) (2022)
19. Zhou, D., et al.: Ontology Reshaping for Knowledge Graph Construction: Applied on Bosch Welding Case. In: ISWC 2022. pp. 770–790 (2022)