Discovering alignment relations with Graph Convolutional Networks: A biomedical case study

Pierre Monnin [a,b,*], Chedy Raïssi [a,c], Amedeo Napoli [a] and Adrien Coulet [a,d,e]

[a] *Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France*
E-mails: pierre.monnin@loria.fr, chedy.raissi@inria.fr, amedeo.napoli@loria.fr, adrien.coulet@inria.fr
[b] *Orange, Belfort, France*
[c] *Ubisoft, Singapore*
[d] *Inria Paris, F-75012 Paris, France*
[e] *Centre de Recherche des Cordeliers (UMR1138 Inserm, Université de Paris, Sorbonne Université), F-75006 Paris, France*

Abstract. Knowledge graphs are freely aggregated, published, and edited in the Web of data, and thus may overlap. Hence, a key task resides in aligning (or matching) their content. This task encompasses the identification, within an aggregated knowledge graph, of nodes that are equivalent, more specific, or weakly related. In this article, we propose to match nodes within a knowledge graph by *(i)* learning node embeddings with Graph Convolutional Networks such that similar nodes have low distances in the embedding space, and *(ii)* clustering nodes based on their embeddings, in order to suggest alignment relations between nodes of a same cluster. We conducted experiments with this approach on the real world application of aligning knowledge in the field of pharmacogenomics, which motivated our study. We particularly investigated the interplay between domain knowledge and GCN models with the two following focuses. First, we applied inference rules associated with domain knowledge, independently or combined, before learning node embeddings, and we measured the improvements in matching results. Second, while our GCN model is agnostic to the exact alignment relations (*e.g.*, equivalence, weak similarity), we observed that distances in the embedding space are coherent with the "strength" of these different relations (*e.g.*, smaller distances for equivalences), letting us considering clustering and distances in the embedding space as a means to suggest alignment relations in our case study.

Keywords: Knowledge graph, matching, embedding, Graph Convolutional Network, ontology, clustering

1. Introduction

The Semantic Web [3] offers tools and standards that facilitate the construction of knowledge graphs [17] that may aggregate data and elements of knowledge of various provenances. The combined use of these scattered elements

---

*Corresponding author. E-mail: pierre.monnin@loria.fr.

of knowledge allows access to a larger extent of the available knowledge, which is beneficial to many applications, such as fact-checking or query answering. For this conjoint use to be possible, one crucial task lies in *matching* units across knowledge graphs or within an aggregated knowledge graph, *i.e.*, finding *alignments* or correspondences between nodes, edges, or subgraphs. This task is well-studied in the *Ontology Matching* research field [12] and is challenging since knowledge graphs differ in quality, completeness, vocabularies, and languages. Consequently, different alignment relations may hold between units: some may indicate that two units are equivalent, weakly related, or that one is more specific than the other.

In the present work, we focus on matching specific nodes within an aggregated knowledge graph represented within Semantic Web standards. We view such a knowledge graph as a directed and labeled multigraph in which nodes represent entities of a world – also named individuals – (*e.g.*, places, drugs), literals (*e.g.*, dates, integers), or classes of individuals (*e.g.*, `Person`, `Drug`). It should be noted that we discard litterals from the scope of the present work. Nodes are linked together through edges defined as triples ⟨`subject`, `predicate`, `object`⟩ in the Resource Description Framework (RDF) format language, where the `predicate` qualifies the relationship holding between the `subject` and the `object` (*e.g.*, `has-side-effect`, `has-name`). Entities, classes, and predicates are identified by Uniform Resource Identifiers (URIs). Knowledge graphs can be associated with ontologies, *i.e.*, formal representations of a domain [14], in which classes and predicates are organized in two distinct hierarchies.

We propose to match specific individuals that represent n-ary relationships through an approach that combines graph embedding and clustering, outlined in Fig. 1. Graph embeddings are low-dimensional vectors that represent graph substructures (*e.g.*, nodes, edges, subgraphs) while preserving as much as possible the properties of the graph [5]. More precisely, we learn node embeddings with Graph Convolution Networks (GCNs) [20,29] such that similar nodes have a low distance between their embeddings. We employ graph embeddings since their continuous nature may provide the needed flexibility to cope with the heterogeneous representations of nodes to match [15]. GCNs compute the embedding of a node by considering the embeddings of its neighbors in the graph. Hence,
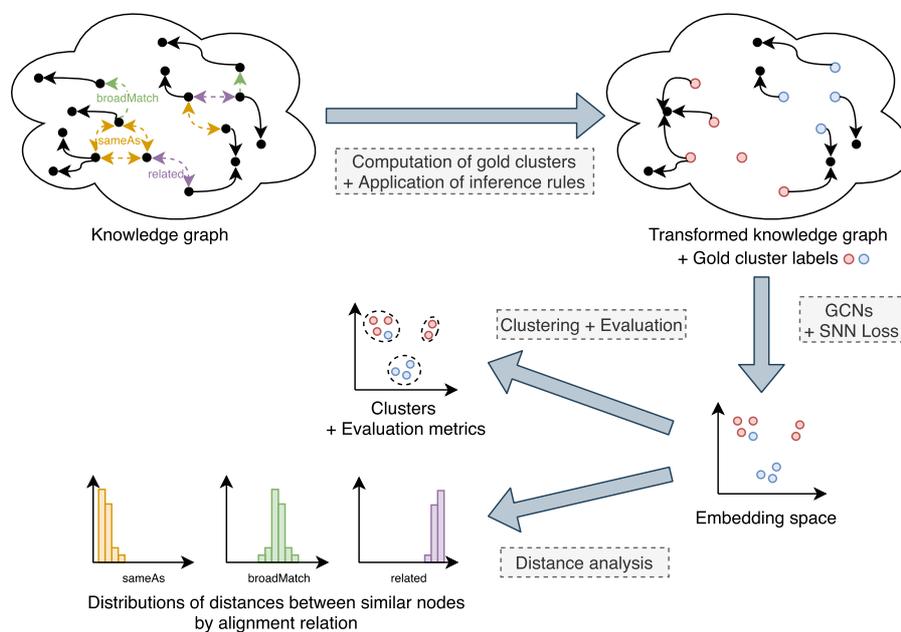
Fig. 1. Outline of our approach. Gold clusters are computed from existing alignments between the nodes to match in the knowledge graph (*e.g.*, `owl:sameAs`, `skos:broadMatch`, `skos:related`, etc.). These alignments are then removed and various inferences rules associated with domain knowledge are applied on the knowledge graph. Embeddings of nodes are learned with Graph Convolutional Networks (GCNs) and the Soft Nearest Neighbor (SNN) loss. Clustering algorithms are then applied on the embedding space and the resulting clusters are evaluated with regard to the gold clusters. A distance analysis is also performed for each alignment relation.

nodes with similar neighborhoods will have similar embeddings, what is well-adapted to a structural and relational matching approach [26,33].

To suggest alignment relations from node embeddings, we apply a clustering algorithm on the embedding space and consider nodes that belong to the same cluster as similar. The resulting clusters are evaluated by comparison with *gold clusters*, *i.e.*, reference clusters that we aim to reproduce. We define these gold clusters as groups of nodes linked directly or indirectly by preexisting alignments we obtained from a rule-based method previously published [21]. These pre-existing alignments use five different alignment relations. For example, nodes may be identical (owl:sameAs links), one may be more specific than the other (skos:broadMatch links), or weakly similar (skos:related links). Hence, our approach is supervised and requires the preexistence of such alignments.

Within our approach, we particularly investigated the interplay between GCNs and domain knowledge through the two following aspects. First, similarly to existing works with different embedding models [18], we applied various inference rules associated with domain knowledge (*e.g.*, class and predicate hierarchies, symmetry and transitivity of predicates), independently or combined, before learning node embeddings, and we measured the improvements or declines in matching results. Second, we explored how embeddings can differentiate between different types of alignment relations. We made our GCN model agnostic to these exact relations during learning. However, we observed that distances between the embeddings of similar nodes are different and coherent with the type and "strength" of each alignment relation (*e.g.*, smaller distances for equivalences, larger distances for weak similarities). Such results allow us to think that distances in the embedding space can be used to suggest alignment relations to connect nodes, in respect with distinct types of similarities. To the best of our knowledge, our approach is the first one to investigate these aspects in a matching task, combining GCNs and clustering.

Our approach based on GCNs was motivated by the need to align pharmacogenomic (PGx) knowledge that we previously aggregated in a knowledge graph named PGxLOD [22]. The biomedical domain of PGx studies the influence of genetic factors on drug response phenotypes. As an example, Fig. 2 depicts the relationship pgr_1, which states that patients treated with warfarin may experience vascular disorders because of variations in the CYP2C9 gene. PGx knowledge originates from distinct sources: reference databases such as PharmGKB [34], biomedical literature, or the mining of Electronic Health Records of hospitals. Consequently, there is an interest in matching these sources to obtain a consolidated view of the PGx knowledge. Such a view would certainly be beneficial to precision medicine, which aims at tailoring drug treatments to patients to reduce adverse effects and maximize drug efficacy [6,9]. Elements of PGx knowledge consist of *n*-ary relationships between drugs, genomic variations, and phenotypes, whereas only binary relations exist in Semantic Web standards. Thus, PGx relationships in PGxLOD are reified as individual nodes whose neighbors are the involved drugs, genetic factors, and phenotypes (see Fig. 2) [25]. In this context, matching PGx relationships reduces to matching the nodes resulting from their reification. By using GCNs, we hope that nodes representing PGx relationships that involve similar drugs, genetic factors, and phenotypes will have similar embeddings since they have similar neighborhoods.

The remainder of this paper is organized as follows. In Section 2, we outline some works related to node matching in knowledge graphs and graph embeddings. We detail the core of our matching approach (node embeddings and clustering) in Section 3, and how inference rules associated with domain knowledge are considered in Section 4. In Section 5, we conduct experiments with this approach on PGxLOD, a large knowledge graph we built that contains 50,435 PGx relationships [22]. Finally, we discuss our results and conclude in Section 6 and 7.
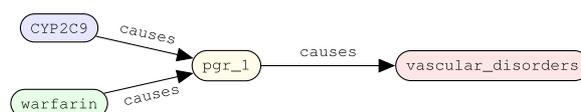

Fig. 2. Representation of a PGx relationship between gene CYP2C9, drug warfarin and phenotype vascular_disorders. This relationship is reified through the individual pgr_1, connecting its components through the causes predicate.

2. Related work

2.1. Matching

Numerous papers exist about knowledge graph matching. The interested reader could refer to the book of Euzenat and Shvaiko [12] for a formalization of the matching task, and a detailed presentation of the main methods. In the following, we focus on graph embedding techniques. Such techniques have been successfully applied on knowledge graphs for various tasks such as node classification, link prediction, or node clustering [5,32]. Interestingly, the task of matching nodes can be alternatively tackled as a link prediction task (*i.e.*, predicting alignments between nodes) or as a node clustering task (*i.e.*, grouping similar nodes into clusters). Here, we choose the node clustering approach.

2.2. Graph embedding

Existing papers about graph embedding differ in the considered type of graphs (*e.g.*, homogeneous graphs, heterogeneous graphs such as knowledge graphs) or in the embedding techniques (*e.g.*, matrix factorization, deep learning with or without random walk). The survey of Cai et al. [5] presents a taxonomies of graph embedding problems and techniques. Hereafter, a few specific examples are detailed but a more thorough overview can be found in the following surveys [5,24,32]. Some approaches are translational. For example, TransE [4] computes for each triple $\langle s, p, o \rangle$ of a knowledge graph, embeddings h_s, h_p, h_o, such that $h_s + h_p \approx h_o$, *i.e.*, the translation vector from the subject to the object of a triple corresponds to the embedding of the predicate. This approach is adapted for link prediction but, according to the authors, it is unclear if it can adequately model relations of distinct arities, such as *1-to-Many*, or *Many-to-Many*. Other approaches use random walks in the knowledge graph. For example, RDF2Vec [28] first extracts, for each node, a set of sequences of graph sub-structures starting from this node. Elements in these sequences can be edges, nodes, or subtrees. Then, sequences feed the word2vec model that compute embeddings for each element in a sequence by either maximizing the probability of an element given the other elements of the sequence (Continuous Bag of World architecture) or maximizing the probability of the other elements given the considered element (Skip-gram architecture).

2.3. Graph Convolutional Networks (GCNs)

The approach adopted in this article is based on Graph Convolutional Networks (GCNs). GCNs have been introduced for semi-supervised classification over graphs [20] and extended for entity classification and link prediction in knowledge graphs [29]. In contrast with TransE and RDF2Vec that work at the triple and sequence levels, GCNs compute the embedding of a node by considering its neighborhood in the graph. Hence, as aforementioned, we believe GCNs are well-suited for our application of matching reified *n*-ary relationships since similar relationships have similar neighborhoods. Other existing works rely on this assumption that similar nodes have similar neighborhoods and use GCNs for their matching. For example, Wang et al. [33] propose to align cross-lingual knowledge graphs by using GCNs to learn node embeddings such that nodes representing the same entity in different languages have close embeddings. Pang et al. [26] use the same approach to align two knowledge graphs, but introduce an iterative aspect. Some newly-aligned entities are selected and used when learning embeddings in the next iteration. To avoid introducing false positive alignments, the newly-aligned entities are selected with a distance-based criteria proposed by the authors. Interestingly, the two previous approaches take into account literals in the embedding process and use the *triplet loss*, also used by TransE. On the contrary, in our work, we discard literals and use the *Soft Nearest Neighbor loss* [13] to consider all positive and negative examples instead of sampling.[1]

---

[1]GCNs and the Soft Nearest Neighbor loss are further detailed in Section 3.2.

2.4. Graph embedding and domain knowledge

However, previous methods do not consider inference rules associated with domain knowledge represented in knowledge graphs on the contrary of recent papers [27]. For example, Iana and Paulheim [18] evaluate the RDF2Vec embedding model when inferred triples associated with subproperties, symmetry, and transitivity of predicates are added to the knowledge graph. Interestingly, the addition of inferred triples seems to degrade the performance of RDV2Vec embeddings in downstream applications (*e.g.*, regression, classification). Instead of materializing inferred triples into the knowledge graph, d'Amato et al. [10] propose to inject domain knowledge in the learning process by defining specific loss functions and scoring functions for triples. Logic Tensor Networks [30] learn groundings of logical terms and logical clauses. The grounding of a logical term consists in a vector of real numbers (*i.e.*, an embedding) and the grounding of a logical clause is a real number in the interval [0, 1] (*i.e.*, the confidence in the truth of the clause). The learning process aims at minimizing the satisfiability error of a set of clauses, while ensuring the logical reasoning. This work can interestingly be compared to graph embeddings if knowledge graphs are considered in their logical form, *i.e.*, considering nodes as logical terms and edges linking two nodes as logical formulae. Alternatively, Wang et al. [31] propose an hybrid attention mechanism named "Logic Attention Network" (LAN) in embedding approaches for link prediction. LAN combines a mechanism based on logical rules and a neural network mechanism. The rule-based mechanism weights neighbors by promoting those linked by a predicate that has been found to strongly imply the predicate of the link to predict. Besides implications between predicates, more complex logical rules can be associated with knowledge graphs through ontologies. That is why Gutiérrez-Basulto and Schockaert [16] investigate how to ensure logical consistency through geometrical constraints on embedding spaces and if classical embedding techniques respect such constraints. Similarly, OWL2Vec⋆ [7] focus on embedding complex logical constructors as well as the graph structure and literals.

These related works and our preliminary results [23] inspired the present work where we investigate how *(i)* inference rules associated with domain knowledge can improve the performances in node matching and *(ii)* the distance in the embedding space is representative of the type and "strength" of alignment relations, and thus can be used to suggest the specific relation to use between matched nodes.

3. Matching nodes with Graph Convolutional Networks and clustering

3.1. Approach outline

Our approach is outlined in Fig. 1.

It takes as input an aggregated knowledge graph $\mathcal{K}$ and a set S of nodes to match, where S is a subset of the nodes of $\mathcal{K}$. This initial selection of the nodes to match is motivated by our biomedical application as we only intend to match nodes that represent reified PGx relationships. We discard literals and edges incident to literals from $\mathcal{K}$ and S. Hence, a node is either an entity or a class. We consider that we have at our disposal *gold clusters*, *i.e.*, sets of nodes from S that are already labeled as similar. These gold clusters can have uneven sizes. We propose to match nodes in S as follows:

1. Learn embeddings for all nodes in $\mathcal{K}$ such that nodes in S labeled as similar (*i.e.*, belonging to the same gold cluster) have smaller distances between their embeddings (Section 3.2).
2. Apply a clustering algorithm only on the embeddings of nodes from S and consider nodes belonging to the same cluster as similar (Section 3.3).

It should be noted that gold clusters can result from another automatic matching method or a manual alignment by an expert. For example, in Section 5, our gold clusters are computed from alignments semi-automatically obtained with rules manually written by experts [21]. These alignments can use different alignment relations (*e.g.*, equivalence, weak similarity). We further detail in Section 5.1 how distinct relations are taken into account in our experiments.

3.2. Learning node embeddings with Graph Convolutional Networks and the Soft Nearest Neighbor loss

To learn embeddings for all nodes in $\mathcal{K}$, we propose to use Graph Convolutional Networks (GCNs) and the Soft Nearest Neighbor loss. In the following, we adopt the model, notations, and definitions of Schlichtkrull et al. [29]. As such, $\mathcal{R}$ denotes the set of predicates in the considered knowledge graph $\mathcal{K}$. Given a node i and a predicate $r \in \mathcal{R}$, we denote by $\mathcal{N}_i^r$ the set of nodes reachable from i by an edge labeled by r.

Graph Convolutional Networks (GCNs) can be seen as a message-passing framework of multiple layers, in which the embedding $h_i^{(l+1)}$ of a node i at layer $(l + 1)$ depends on the embeddings of its neighbors at level (l), as stated in Eq. (1).

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right) \tag{1}$$

This convolution over the neighboring nodes j of i is computed with a specific weight matrix $W_r^{(l)}$ for each predicate $r \in \mathcal{R}$ and each layer (l). The convolution is regularized by a constant $c_{i,r}$, that can be set for each node and each predicate. Similarly to Schlichtkrull et al. [29], we use $c_{i,r} = |\mathcal{N}_i^r|$. The weight matrix $W_0^{(l)}$ enables a self-connection, *i.e.*, the embedding of i at layer $(l + 1)$ also depends on its embedding at layer (l). σ is a non-linear function such as ReLU or tanh.

The number of predicates in $\mathcal{K}$ can lead to a high number of parameters $W_r^{(l)}$ to optimize. To ensure the scalability of our approach and reduce the number of parameters to optimize, we use the basis-decomposition proposed by Schlichtkrull et al. [29]. Hence, each $W_r^{(l)}$ is decomposed as follows:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)} \tag{2}$$

For each level (l), B matrices $V_b^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ and $|\mathcal{R}| \times B$ coefficients $a_{rb}^{(l)} \in \mathbb{R}$ are learned, where $d^{(l)}$ and $d^{(l+1)}$ denote the dimension of embeddings at level (l) and level $(l + 1)$ respectively. Then, each $W_r^{(l)}$ is computed as a linear combination of matrices $V_b^{(l)}$ and coefficients $a_{rb}^{(l)}$. As only these coefficients depend on predicates r, the number of parameters to learn is reduced.

Recall that our objective is to cluster similar nodes, which differs from previous applications of GCNs (*e.g.*, node classification, link prediction [20,29,33]). Hence, we propose to train GCNs from scratch by minimizing the Soft Nearest Neighbor (SNN) loss which, to the best of our knowledge, has never been used with GCN models before. This loss was defined by Frosst et al. [13] and is presented in Eq. (3),

$$\mathcal{L}_{\text{SNN}} = -\frac{1}{|N|} \sum_{i \in N} \log \left(\frac{\sum_{\substack{j \in N \\ j \neq i \\ Y_i = Y_j}} e^{-\frac{\|h_i - h_j\|^2}{T}}}{\sum_{\substack{k \in N \\ k \neq i}} e^{-\frac{\|h_i - h_k\|^2}{T}}} \right) \tag{3}$$

The input of the SNN loss consists of:

 – A set N of nodes belonging to the gold clusters (see Section 5.2).
 – A set Y of labels for nodes in N. These labels corresponds to the assignments of nodes in N to the gold clusters.
 – A temperature T.
 – Embeddings h of nodes. These embeddings are the output of the last layer of the GCN model.

Minimizing the SNN loss corresponds to minimizing intra-cluster distances and maximizing inter-cluster distances for the gold clusters of nodes in N. The temperature T determines how distances influence the loss. Indeed, distances

between widely separated embeddings are taken into account when T is large whereas only distances between close embeddings are taken into account when T is small. To avoid T as an hyperparameter of the model, we adopt the same learning procedure as Frosst et al. [13]: T is initialized to a predefined value and is optimized by learning $\frac{1}{T}$ as a model parameter.

The computation of $\mathcal{L}_{\text{SNN}}$ (Eq. (3)) considers all positive and negative examples from N. Indeed, distances between nodes with the same label are minimized (*i.e.*, positive examples) whereas distances between nodes with different labels are maximized (*i.e.*, negative examples). However, it is noteworthy that $\mathcal{K}$ is based on the *Open World Assumption*. Hence, nodes with different labels are regarded as dissimilar (*i.e.*, negative examples) while their (dis)similarity may only be unknown.

This step enables to learn embeddings for all nodes in $\mathcal{K}$ such that distances between embeddings of identified similar nodes (in N in $\mathcal{L}_{\text{SNN}}$) are low. Note that, in this learning procedure, the semantics of relations in $\mathcal{K}$ is not taken into account. We propose to take into account this semantics with a preprocessing step presented in Section 4.

3.3. Matching nodes by clustering their embeddings

After embeddings of all nodes in the graph have been output by the last layer of the GCN, we perform a clustering on embeddings h_i for all nodes $i \in S$, *i.e.*, all nodes to check for matching. Nodes assigned to one same cluster are considered as similar and connected with an alignment relation. Clusters are evaluated with regard to gold clusters.

We conduct comparative experiments with three distinct clustering algorithms presented in Table 1, in regards with three classical metrics presented in Table 2. Within the large set of existing algorithms, our choice has been guided by the constraints of our task that requires to handle an important number of clusters, potentially large, and with uneven sizes (see Section 5.1 and Fig. 3 for the sizes of gold clusters computed on PGxLOD). We decided to arbitrarily limit ourselves to three algorithms, but decided to opt for algorithms that cover some diversity in the various family of algorithms. Our three algorithms differ in their parameters: in particular they require either the number of clusters to find (Ward and Single) or the minimum size of clusters (OPTICS). We used our set of gold clusters to set these parameters. Considering these distinct algorithms allows us to evaluate the influence of inference rules in different settings (see Section 4).

Table 1

Clustering algorithms applied on the embeddings of nodes in S. Nodes that belong to the same predicted cluster are considered as similar

Algorithm	Parameter	Description
Ward	Number of clusters to find	Hierarchical clustering algorithm that successively merges clusters by minimizing the variance of merged clusters
Single	Number of clusters to find	Hierarchical clustering algorithm that successively merges clusters whose distance between their closest observations is minimal
OPTICS [1]	Minimum size of clusters	Algorithm that finds zones of high density and expand clusters from them

Table 2

Performance metrics used to compare the clusters predicted by the algorithms presented in Table 1 with gold clusters

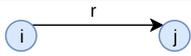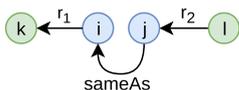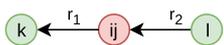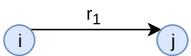| Metric	Abbr.	Domain	Description
Unsupervised Clustering Accuracy	ACC	[0, 1]	Counts nodes whose predicted cluster label is the same as their gold cluster label divided by the total number of nodes. As labels may be permuted between predicted and gold clusters, the mapping with the best ACC is used.
Adjusted Rand Index	ARI	[−1, 1]	Considers all pairs of nodes and counts those whose nodes are assigned to the same or different clusters both in predicted and gold clusters. ARI is equal to 0 for a random labeling, and equal to 1 for a perfect labeling (up to a permutation). ARI is adjusted for chance.
Normalized Mutual Information	NMI	[0, 1]	Measures the mutual information between the predicted and gold clusters, normalized by the entropy of both types of clusters. NMI is equal to 1 for a perfect labeling (up to a permutation).

4. Evaluating the influence of applying inference rules associated with domain knowledge

Semantic Web knowledge graphs are represented within formalims such as Description Logics [2] that are equipped with inference rules. Hence, we propose to evaluate the improvements in the results of our matching approach (detailed in Section 3) when considering such inference rules, independently or combined. Here, we only consider the inference rules associated with the following logic axioms: class and predicate assertions, equivalence axioms between entities or classes, subsumption axioms between classes or predicates, and axioms defining predicate inverses. We consider this limited set of inference rules because they are the only ones actionable in PGxLOD, the knowledge graph that motivated our study. Accordingly, we generate six different graphs ($\mathcal{G}_{0-5}$) by running over $\mathcal{K}$ these inference rules until saturation. This inference and saturation process is implemented in a Python script, without the use of an inference engine in part because at this stage the graph is in the format of the GCN library and in part because this facilitates the independent activation of inference rules that is required by our experiment. Soundness and completeness of this inference and saturation process were carefully checked but scalability was not tested on knowledge graphs larger than PGxLOD. We then test our approach on these six graphs that are summarized in Table 3 and further described below.

$\mathcal{G}_0$ constitutes the baseline in which no inference rules are run and with the systematic addition of abstract inverses. Indeed, Schlichtkrull et al. [29] consider that for every predicate $r \in \mathcal{R}$, there exists an inverse $r_{\text{inv}} \in \mathcal{R}$. Thus, for every $r \in \mathcal{R}$, we add an abstract inverse $r_{\text{inv}} \in \mathcal{R}$ such that its adjacency matrix represents the inverse of r. This addition of abstract inverses is performed in all other graphs, except when explicitly stated otherwise. $\mathcal{G}_1$ results from the contraction of `owl:sameAs` edges. Indeed, in $\mathcal{K}$, several nodes representing the same entity can co-exist. In this case, they may be linked (dire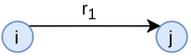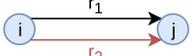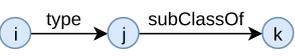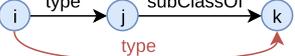ctly or indirectly) by `owl:sameAs` edges and should be considered as one, which is enabled by this contraction. It is noteworthy that the standard transformation actually consists in

Table 3

Visual summary of the transformations of $\mathcal{K}$ to evaluate the influence of the application of inference rules associated with domain knowledge on node matching. $\mathcal{G}_0$ is the baseline that corresponds to no inference rules being run and the systematic addition of abstract inverses

Graph	Before	After
$\mathcal{G}_0$		
$\mathcal{G}_1$		
$\mathcal{G}_2$		
$\mathcal{G}_3$		
$\mathcal{G}_4$		
$\mathcal{G}_5$	All transformations from $\mathcal{G}_1$ to $\mathcal{G}_4$	
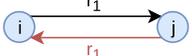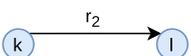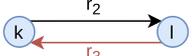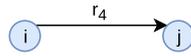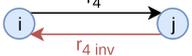
propagating edges to equivalent nodes instead of merging them. However, in our case study, the merging transformation was prefered because it is aligned with the architecture of GCNs that corresponds to the topological structure of the graph. In $\mathcal{G}_2$, we do not always add abstract inverses but consider definitions of inverses and symmetry of predicates instead. That is to say:

(i) For a predicate r_1 defined as symmetric (*i.e.*, $r_1 \equiv r_1^{-1}$), we do not add an abstract inverse $r_{1\,\text{inv}}$ and complete its adjacency matrix to ensure its symmetry.

(ii) For a predicate r_2 that has a defined inverse r_3 (*i.e.*, $r_3 \equiv r_2^{-1}$), we do not add an abstract inverse $r_{2\,\text{inv}}$ and complete their adjacency matrices to ensure they represent inverse predicates.

(iii) Otherwise, for a predicate r_4 that neither is symmetric nor have a defined inverse, we add an abstract inverse $r_{4\,\text{inv}}$ such that its adjacency matrix represents the inverse of r_4.

$\mathcal{G}_3$ takes into account the hierarchy of predicates. Indeed, if a predicate r_1 is a subpredicate of r_2 (*i.e.*, $r_1 \sqsubseteq r_2$) and a triple $\langle i, r_1, j \rangle$ exists, then we make sure the triple $\langle i, r_2, j \rangle$ also exists in the graph. This completion is performed by considering the transitive closure of the subsumption relation $\sqsubseteq$. That is to say, if $r_1 \sqsubseteq r_2$ and $r_2 \sqsubseteq r_3$, we also consider $r_1 \sqsubseteq r_3$. Similarly, $\mathcal{G}_4$ completes `type` edges based on the hierarchy of ontology classes defined by `subClassOf` edges. Hence, if $\langle i, \texttt{type}, j \rangle$ and $\langle j, \texttt{subClassOf}, k \rangle$ exist in the graph, then we ensure that $\langle i, \texttt{type}, k \rangle$ is also in the graph. Here again, `subClassOf` edges are considered by computing their transitive closure. Finally, $\mathcal{G}_5$ is the graph resulting from all transformations from $\mathcal{G}_1$ to $\mathcal{G}_4$.

5. Experiments

We conducted experiments with PGxLOD,[2] a large knowledge graph about pharmacogenomics (PGx) that we previously built and that motivated this study [22]. Our approach is implemented in Python, using PyTorch and the Deep Graph Library for learning embeddings, and scikit-learn for clustering. Our code is available on GitHub.[3]

5.1. Knowledge graph and gold clusters of similar nodes

PGxLOD presents several characteristics needed in the scope of our study. First, PGxLOD contains nodes whose matching is well-adapted to a structure-based approach such as ours. Additionally, alignments are expected to be found between these nodes. Indeed, PGxLOD contains 50,435 PGx relationships resulting from:

– an automatic extraction from the reference database PharmGKB;
– an automatic extraction from the biomedical literature;
– a manual representation of 10 studies made from Electronic Health Records of hospitals.

Alignments are expected to be found between such relationships since, for example, PharmGKB is manually curated by experts after a literature review. Recall that PGx relationships are *n*-ary, and thus they are reified as nodes, as illustrated in Fig. 2 [25]. Hence, nodes representing these relationships form our set *S* of nodes to match. The reification process entails that neighbors of such nodes are the drugs, genetic factors, and phenotypes involved in the relationships. Consequently, similar relationships have similar neighborhoods, which makes a structure-based approach such as ours well-adapted for their matching.

Second, PGxLOD contains `owl:sameAs` edges (or equivalence axioms), which makes possible the transformation represented in $\mathcal{G}_1$. Indeed, PGxLOD integrates several Linked Open Data sets: ClinVar, DrugBank, SIDER, DisGeNET, PharmGKB, and CTD. These LOD sets contain facts describing components of PGx relationships (*i.e.*, drugs, phenotypes, and genetic factors). Several LOD sets may describe the same entities and we know it explicitly, *i.e.*, some nodes belonging to different LOD sets are linked with `owl:sameAs` edges. For example, this could be the case of a drug represented both in PharmGKB and DrugBank. Thus, we can apply the `owl:sameAs` identification.

Table 4

Alignment relations considered in each gold clustering to compute the gold clusters used in our experiments. We indicate whether a relation is transitive (T or ¬ T) and symmetric (S or ¬ S)

	owl:sameAs	skos:closeMatch	skos:relatedMatch	skos:related	skos:broadMatch
	T, S	T, S	T, S	¬ T, S	T, ¬ S
$\mathcal{C}_0$	×	×	×	×	×
$\mathcal{C}_1$	×	×	×	×	
$\mathcal{C}_2$	×				
$\mathcal{C}_3$		×			
$\mathcal{C}_4$			×		
$\mathcal{C}_5$				×	
$\mathcal{C}_6$					×

Third, PGxLOD contains subsumption axioms between classes and between predicates, which makes possible the transformations represented in $\mathcal{G}_3$ and $\mathcal{G}_4$. Indeed, PGxLOD includes the ATC, MeSH, PGxO, and ChEBI ontologies.

Fourth, some PGx relationships in S are already labeled as similar through alignments resulting from the application of five matching rules previously published [21]. These alignments use the five following alignment relations: owl:sameAs, skos:closeMatch, skos:relatedMatch, skos:related, and skos:broadMatch. Alignments using owl:sameAs and skos:closeMatch indicate strong similarities, whereas skos:relatedMatch and skos:related indicate weaker similarities. Alignments using skos:broadMatch indicate that a PGx relationship is more specific than another. These alignments are removed before running inference rules over $\mathcal{K}$, learning embeddings, and clustering. However, they allow to compute *gold clusters*, *i.e.*, sets of nodes that are considered as similar since they are directly or indirectly connected through alignments. We propose the different *gold clusterings* detailed in Table 4 (named $\mathcal{C}_0$ to $\mathcal{C}_6$). They variously consider the five alignment relations when computing gold clusters to evaluate our approach in different settings (*e.g.*, all the different alignment relations in $\mathcal{C}_0$, only symmetric relations in $\mathcal{C}_1$, only equivalences in $\mathcal{C}_2$). For each gold clustering, gold clusters correspond to the connected components computed by only considering the (undirected) alignments of the selected alignment relations between nodes in S. Hence, all alignment relations are regarded as symmetric (undirected links) and transitive (connected components), which is coherent with the majority of alignment relations (see Table 4). Figure 3 presents the sizes of the resulting gold clusters. We notice that many gold clusters have a size lower or equal to 10, and that considering skos:related or skos:broadMatch links increases the maximal size of gold clusters. The availability of all these different alignment relations also allows to perform the distance analysis described in Section 5.4 and indicated in Fig. 1.

5.2. Learning node embeddings

We experimented our approach with different pairs $(\mathcal{C}_i, \mathcal{G}_j)$ that were selected for their experimental interest. All gold clusterings were experimented with graphs $\mathcal{G}_0$ and $\mathcal{G}_5$ to have a global view of the impact on performance of applying inference rules associated with domain knowledge. All graphs were experimented with $\mathcal{C}_0$ to have a finer evaluation of each inference rule on the most heterogeneous gold clustering. For each pair $(\mathcal{C}_i, \mathcal{G}_j)$, a 5-fold cross-validation was performed as follows. For each $\mathcal{C}_i$, S is split into five sets S_k^i ($k \in \{1, 2, 3, 4, 5\}$). All S_k^i contain the same number of nodes for each gold cluster of $\mathcal{C}_i$ larger than 5 nodes. Each set S_k^i is successively used as the test set S_{test}, while set $S_{(k+1)}^i$ is used as the validation set S_{val}.[4] Remaining sets form the train set S_{train}. This corresponds to a random split of nodes to match with 60% train – 20% validation – 20% test.

An architecture formed by 3 GCN layers is used to learn node embeddings. The input layer consists of a featureless approach as in [20,29], *i.e.*, the input is just a one-hot vector for each node of the graph. It should be noted that this one-hot vector encoding can scale to relatively large knowledge graphs because *(i)* we use look-up mechanisms

---

[4] S_1^i is the validation set when $S_{\text{test}} = S_5^i$.

(a) $\mathcal{C}_0$ (max = 17,568)

(b) $\mathcal{C}_1$ (max = 16,961)

(c) $\mathcal{C}_2$ (max = 183)

(d) $\mathcal{C}_3$ (max = 69)

(e) $\mathcal{C}_4$ (max = 892)

(f) $\mathcal{C}_5$ (max = 16,942)

(g) $\mathcal{C}_6$ (max = 2,501)
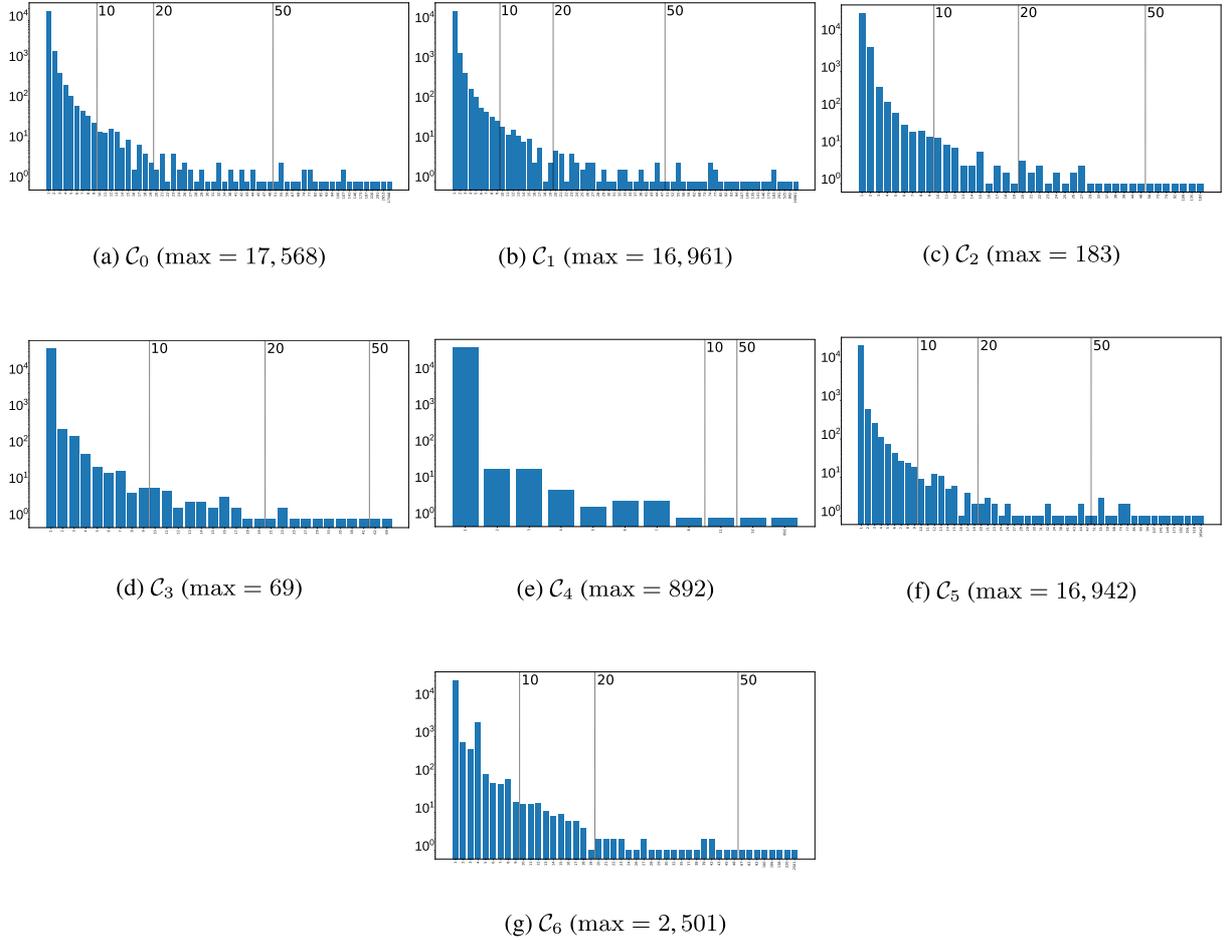
Fig. 3. Number of gold clusters (y-axis) by size (x-axis) for each gold clustering. The max value is the maximum size of gold clusters (in terms of number of nodes). The minimum size is 1 for every gold clustering. Only gold clusters larger than 10, 20, and 50 nodes are later used to compute performance metrics. Gold clusterings are defined in Table 4.

in weight matrices based on node indices, and thus one-hot vectors are actually never stored in memory or used in computations, and *(ii)* we use a basis-decomposition to limit the number of parameters. All three layers of our architecture have an output dimension of 16. Therefore, output embeddings for all nodes in the knowledge graph are in $\mathbb{R}^{16}$. The activation function used on the input and hidden layers is tanh while the output layer uses a linear function. We use a basis-decomposition of 10 bases and set $c_{i,r} = |\mathcal{N}_i^r|$ for all i and all r. Our architecture is similar to the one used in [29] except for the dimension of the hidden layer and the activation function of the output layer. In such a 3-layer architecture, it follows from Eq. (1) that only neighboring nodes up to 3 hops[5] of nodes in S will have an impact on their embeddings, output at layer 3. Thus, to save memory, we reduce graphs to such 3-hop neighborhoods. Statistics about these reduced graphs are available in Table 5.

Only the embeddings of nodes in S (here, the PGx relationships) are considered in our clustering task. Hence, only these embeddings are constrained in the SNN loss. However, in $\mathcal{L}_{SNN}$ (Eq. (3)), each node needs at least one other node assigned to the same gold cluster (*i.e.*, having the same label). Thus, only gold clusters of size greater or equal to 10 are used in the learning process since each S_k^i contains at least 2 nodes of these clusters. This is

---

[5]The 3-hop neighborhood of a node n consists of all the nodes that can be reached with a breadth-first traversal that starts at n and traverses at most 3 edges.

Table 5

Statistics of PGxLOD and its transformations as described in Section 4. Statistics for PGxLOD discard literals and edges incident to literals. As we use a 3-layer architecture, statistics for all $\mathcal{G}_i$ only consider neighboring nodes up to 3 hops of nodes in S (*i.e.*, PGx relationships to match). # denotes "number of"

	# nodes	# edges	# predicates
PGxLOD	11,808,396	43,341,712	416
$\mathcal{G}_0$	3,758,814	39,956,844	689
$\mathcal{G}_1$	3,879,081	46,960,365	733
$\mathcal{G}_2$	3,758,814	22,085,701	347
$\mathcal{G}_3$	3,758,814	41,048,190	697
$\mathcal{G}_4$	3,758,928	42,691,984	701
$\mathcal{G}_5$	3,882,945	27,277,789	375

particularly needed for the validation and test losses but we chose to use the same constraint for the train loss for homogeneity. We use the Adam optimizer [19] with a starting learning rate of 0.01. T is initialized to 1. We learn during 200 epochs with an early-stopping mechanism: if the validation loss does not decrease of 0.0001 after 10 epochs, the learning process is stopped.

5.3. Clustering

Clustering algorithms are only applied on the embeddings of nodes in S_{test} since they are the nodes we aim to match. Recall that the learning process only considers nodes belonging to gold clusters whose size is greater or equal to 10. Accordingly, we apply the three clustering algorithms introduced in Table 1 and evaluate their performance on embeddings of nodes in S_{test} that belong to gold clusters whose size is greater or equal to 50, 20, and 10. These different sizes allow to evaluate the influence of inference rules in the performance of our matching approach when considering only large or all gold clusters.

Results on all gold clusterings and graphs $\mathcal{G}_0$ and $\mathcal{G}_5$ are summarized in Table 6. Detailed results are available Table 8, Table 9 and Table 10 in the Appendix. In these supplementary tables, gray cells indicate the best results among clustering algorithms given a gold clustering, a graph, and a metric. For example, in Table 8, considering $\mathcal{C}_0$ and $\mathcal{G}_0$, the best ACC is obtained with the Single clustering algorithm. Underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$ given a gold clustering and a metric. For example, in Table 8, given $\mathcal{C}_1$, the best NMI for Ward is obtained with $\mathcal{G}_0$ whereas the best ACC is obtained with $\mathcal{G}_5$. We notice in Table 6 that applying all inference rules (*i.e.*, $\mathcal{G}_5$) generally increases performance for $\mathcal{C}_0$ and $\mathcal{C}_1$ which are gold clusterings that mix different alignment relations. Results for the other gold clusterings do not show such an homogeneous and important increase in performance.

Results on $\mathcal{C}_0$ and all graphs are summarized in Table 7. Detailed results are available in Table 11, Table 12, and Table 13 in the Appendix. In these tables, gray cells indicate the best result among clustering algorithms and underlined values indicate the best result between graphs. For example, in Table 11, given $\mathcal{G}_0$, the best ACC is obtained with the Single clustering algorithm. Given the Single algorithm, the best ARI is obtained with $\mathcal{G}_3$ and $\mathcal{G}_5$. Here again, we notice that applying all inference rules (*i.e.*, $\mathcal{G}_5$) leads to the best results. However, computing all instantiations based on the transitive closure of the subsumption (*i.e.*, $\mathcal{G}_4$) seems to degrade clustering performance.

5.4. Distance analysis

During learning and clustering, our model is unaware of the different alignment relations holding between similar nodes. Indeed, the SNN loss only considers labels of gold clusters that do not indicate the alignment relations used to compute these clusters. This is particularly relevant for gold clusterings $\mathcal{C}_0$ and $\mathcal{C}_1$ that mix different alignment relations to compute the gold clusters. However, inspired by our preliminary results [23], we display in Fig. 4 the

Table 6

Summary of the results of clustering nodes for all gold clusterings and graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. For each gold clustering, a cross ($\times$) indicates whether the best results were obtained with $\mathcal{G}_0$ or $\mathcal{G}_5$. Detailed results are available in Tables 8, 9, and 10 in the Appendix

Size of gold clusters	Gold clustering	$\mathcal{G}_0$	$\mathcal{G}_5$
$\geqslant 50$	$\mathcal{C}_0$		$\times$
	$\mathcal{C}_1$		$\times$
	$\mathcal{C}_2$	$\times$	
	$\mathcal{C}_3$	$\times$	$\times$
	$\mathcal{C}_4$	$\times$	$\times$
	$\mathcal{C}_5$	$\times$	
	$\mathcal{C}_6$	$\times$	
$\geqslant 20$	$\mathcal{C}_0$		$\times$
	$\mathcal{C}_1$		$\times$
	$\mathcal{C}_2$		$\times$
	$\mathcal{C}_3$	$\times$	
	$\mathcal{C}_4$		$\times$
	$\mathcal{C}_5$	$\times$	
	$\mathcal{C}_6$	$\times$	$\times$
$\geqslant 10$	$\mathcal{C}_0$		$\times$
	$\mathcal{C}_1$		$\times$
	$\mathcal{C}_2$	$\times$	$\times$
	$\mathcal{C}_3$	$\times$	
	$\mathcal{C}_4$		$\times$
	$\mathcal{C}_5$	$\times$	$\times$
	$\mathcal{C}_6$		$\times$

Table 7

Summary of the results of our clustering for $\mathcal{C}_0$ and all graphs. Detailed results are available in Tables 11, 12, and 13 in the Appendix

Graph	Performance
$\mathcal{G}_0$	Baseline
$\mathcal{G}_1$	Improvements
$\mathcal{G}_2$	Light deterioration
$\mathcal{G}_3$	Improvements
$\mathcal{G}_4$	Consistent deterioration
$\mathcal{G}_5$	Improvements – Best results

distributions of distances between similar nodes in the test set by alignment relation. This analysis is presented for $\mathcal{C}_0$ and graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Interestingly, similarly to our preliminary results [23], such distributions of distances are coherent with the "strength" of the alignment relations. Indeed, for example, nodes that are weakly similar tend to be further apart than equivalent nodes. Only the `skos:broadMatch` relation presents different distance distributions with regard to the distance distributions of the other relations across the different test sets. This could be explained since this is the only non-symmetric relation (see Table 4).
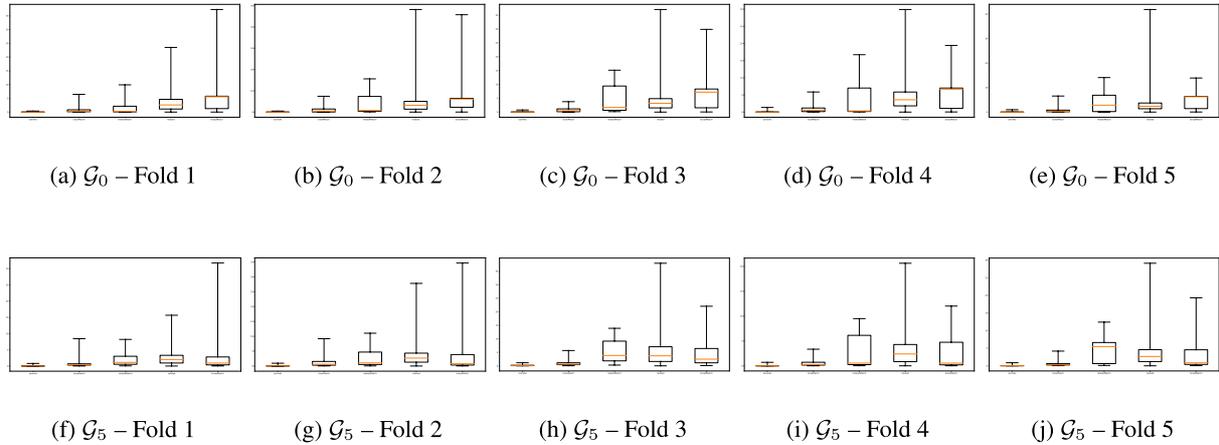
(a) $\mathcal{G}_0$ – Fold 1   (b) $\mathcal{G}_0$ – Fold 2   (c) $\mathcal{G}_0$ – Fold 3   (d) $\mathcal{G}_0$ – Fold 4   (e) $\mathcal{G}_0$ – Fold 5

(f) $\mathcal{G}_5$ – Fold 1   (g) $\mathcal{G}_5$ – Fold 2   (h) $\mathcal{G}_5$ – Fold 3   (i) $\mathcal{G}_5$ – Fold 4   (j) $\mathcal{G}_5$ – Fold 5

Fig. 4. Distributions of distances between similar nodes by alignment relation for each test set, the $\mathcal{C}_0$ gold clustering and the two graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. In each subpicture, links are from left to right: `owl:sameAs`, `skos:closeMatch`, `skos:relatedMatch`, `skos:related`, and `skos:broadMatch`.

6. Discussion

6.1. Impact of inference rules

It appears that $\mathcal{G}_5$ (*i.e.*, all inference rules) consistently increases performance for $\mathcal{C}_0$ and $\mathcal{C}_1$ (see Table 6). Other gold clusterings ($\mathcal{C}_2$–$\mathcal{C}_6$) do not show such an homogeneous and important increase in performance between $\mathcal{G}_0$ and $\mathcal{G}_5$. $\mathcal{C}_0$ and $\mathcal{C}_1$ mix different alignment relations, which leads to a more difficult matching task. This let us think that inference rules associated with domain knowledge provide useful improvements when dealing with heterogeneous similarities and clusters.

In $\mathcal{C}_0$ (clustering with all alignment relations) inference rules seem to improve results with most of the expended graphs, except for $\mathcal{G}_4$ (see Table 7). In $\mathcal{G}_4$, class instantiation is satured. Consequently, "general" classes are directly linked to entities that instantiate them instead of indirectly. Hence, when computing the embeddings of such entities, embeddings of both general and specific classes are directly considered through the same predicate `type`, which makes difficult for the GCN to weight these classes differently. As specific classes are more important than general classes to discriminate similar and dissimilar nodes, their undifferentiated influence in $\mathcal{G}_4$ embeddings may explain the decrease in performance. We notice that $\mathcal{G}_5$ performs best, which advocates for considering all inference rules together. However, based on the degraded performance of $\mathcal{G}_4$ with regard to $\mathcal{G}_0$, one may want to test the progressive addition of inference from $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$. We leave such additional experiment for future works.

6.2. Impact of clustering set-up

Clustering performances are generally better for gold clusterings $\mathcal{C}_2$ to $\mathcal{C}_6$ than $\mathcal{C}_0$ and $\mathcal{C}_1$ (see Table 8, Table 9, and Table 10 in the Appendix). These two gold clusterings mix different alignment relations when computing gold clusters, and thus their matching task is expected to be more difficult. We also notice that performances tend to decrease when considering additional gold clusters (*i.e.*, when decreasing their minimum size). Here again, such a task is more difficult. Indeed, clustering algorithms need to find more clusters (for Ward and Single), or clusters with a reduced minimum size (for OPTICS). However, this is not the case for $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$. This can be explained because, for such gold clusterings, only few gold clusters have a size greater or equal to 50 or 20 (see Fig. 3), and thus only few training examples are available. Hence, reducing the minimum size leads to consider more training examples, and, despite the task being more difficult, improves performance.

Among the considered clustering algorithms, Single generally performs better than the others. For $\mathcal{C}_0$ and $\mathcal{C}_1$, OPTICS is the second best algorithm. For the other gold clusterings, Single and Ward give the best performance. In particular, we notice that OPTICS tends to have a decent ACC but reduced ARI and NMI. As this algorithm is

unaware of the number of clusters to find and only knows their minimum size, low ARI and NMI may indicate a different clustering output in terms of both number and size of clusters. Indeed, ARI counts the pairs of nodes that have similar of different assignments both in predicted and gold clusters and NMI measures the mutual information between two different clusterings. On the contrary, ACC counts the number of nodes correctly assigned. Hence, big gold clusters (partially) correctly assigned may increase the ACC value even between different clusterings. Such a situation arises here since some of our gold clusterings lead to gold clusters with numerous nodes. For example, Fig. 3 shows that a gold cluster in $\mathcal{C}_0$ contains 17,568 nodes. We noticed that in some cases Ward have poor performances. The Ward algorithm minimizes the sum of squared differences within all clusters, whereas Single minimizes the distance between the closest observations of merged clusters. This let us think that the merging criterion of Single is better adapted to the loss function we use (*i.e.*, the Soft Nearest Neighbor loss).

6.3. Distance analysis

Regarding the distance analysis of node embeddings, Fig. 4 shows that distances between similar nodes are different depending on the alignment relation holding between them. Recall that our GCN model is agnostic to these alignment relations when computing the SNN loss. Interestingly, distances reflect the "strength" of the alignment relations: strong similarities (*i.e.*, `owl:sameAs` and `skos:closeMatch` links) have smaller distances than weaker ones (*i.e.*, `skos:relatedMatch` and `skos:related` links). The `skos:broadMatch` relation appears more difficult to position with regard to others. This can be explained as it is the only alignment relation that is not symmetric. Such coherent distributions of distances seem to indicate the "rediscovery" of alignment relations by GCNs and encourages to consider the distance between embeddings of nodes in a "semantic" way, *i.e.*, smaller distances indicate stronger similarities. Hence, an interesting perspective lies in predicting the exact alignment relation holding between similar nodes (*i.e.*, in the same cluster) based on the distance between their embeddings, and evaluating this prediction. Additionally, such different distances also seem to confirm that the neighborhood aggregation of embeddings in GCNs makes them well-suited to a structural and relational matching.

6.4. Towards a further integration of domain knowledge in GCNs

Our results highlight the interest of considering domain knowledge associated with knowledge graphs in embedding approaches and seem to advocate for a further integration of domain knowledge within embedding models. Future works may investigate the same targets with additional inferences rules (*e.g.*, from OWL 2 RL semantics) or different embedding techniques, whether based on graph neural networks [11] or others (*e.g.*, translational approaches such as TransE). Additionally, we did not use attention mechanisms, which could also consider domain knowledge as in Logic Attention Network [31]. Here, inference rules associated with domain knowledge are used to transform the knowledge graph as a pre-processing operation. However, we could envision to consider such mechanisms directly in the model (*e.g.*, weight sharing between predicates and their super-predicates). Literals could also be taken into account [33]. In a larger perspective, one major future work lies in investigating if and how other semantics than types of alignments can emerge in the output embedding space.

6.5. Generalization to other knowledge graphs

Despite our approach being motivated by the matching of individuals within an aggregated knowledge graph, its transposition to distinct graphs could be explored. Such a perspective could allow to assess the generalization of our approach and its results. In particular, we could consider knowledge graphs that are not completely independent such as LOD datasets that are connected through major LOD hubs. Recall that our approach is supervised since gold clusters are computed from preexisting alignments. Hence, testing our approach on different knowledge graphs of the LOD would require such preexisting alignments or using ontology alignment systems in a distant supervision process [8]. In this setting, merging the different graphs into one and learning a "global" embedding, as we did, may provide positive results but may pose additional scalability issues.

7. Conclusion

In this paper, we proposed to match entities of a knowledge graph by learning node embeddings with Graph Convolutional Networks (GCNs) and clustering nodes based on their embeddings. We particularly investigated the interplay between formal semantics associated with knowledge graphs and GCN models. Our results showed that considering inference rules associated with domain knowledge tends to improve performance. Additionally, even if our GCN model was agnostic to the exact alignment relations holding between entities (*e.g.*, equivalence, weak similarity), distances in the embedding space are coherent with the "strength" of the alignment relations. These results seem to advocate for a further integration of formal semantics within embedding models.

Acknowledgements

Appendix. Detailed results of clustering experiments

Detailed results of clustering experiments are available Table 8, Table 9 and Table 10.

Table 8

Results of clustering nodes that belong to gold clusters whose size is greater or equal to 50 for graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Given a gold clustering, gray cells indicate the best results among clustering algorithms and underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$

		$\mathcal{G}_0$			$\mathcal{G}_5$		
		ACC	ARI	NMI	ACC	ARI	NMI
$\mathcal{C}_0$	Ward	0.24 ± 0.02	0.07 ± 0.01	0.37 ± 0.01	0.25 ± 0.02	0.07 ± 0.01	0.35 ± 0.01
	Single	0.84 ± 0.08	0.66 ± 0.13	0.59 ± 0.05	0.90 ± 0.00	0.75 ± 0.01	0.64 ± 0.02
	OPTICS	0.61 ± 0.05	0.21 ± 0.08	0.25 ± 0.04	0.68 ± 0.02	0.27 ± 0.05	0.27 ± 0.02
$\mathcal{C}_1$	Ward	0.19 ± 0.02	0.05 ± 0.00	0.33 ± 0.01	0.20 ± 0.03	0.05 ± 0.01	0.31 ± 0.01
	Single	0.85 ± 0.01	0.55 ± 0.04	0.51 ± 0.03	0.85 ± 0.01	0.57 ± 0.03	0.51 ± 0.03
	OPTICS	0.64 ± 0.03	0.19 ± 0.03	0.28 ± 0.01	0.71 ± 0.04	0.26 ± 0.06	0.30 ± 0.02
$\mathcal{C}_2$	Ward	0.88 ± 0.03	0.84 ± 0.03	0.94 ± 0.01	0.88 ± 0.03	0.84 ± 0.03	0.94 ± 0.01
	Single	0.88 ± 0.03	0.84 ± 0.03	0.94 ± 0.01	0.86 ± 0.04	0.81 ± 0.07	0.93 ± 0.02
	OPTICS	0.94 ± 0.06	0.92 ± 0.07	0.97 ± 0.03	0.91 ± 0.06	0.88 ± 0.08	0.95 ± 0.04
$\mathcal{C}_3$	Ward	0.52 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.53 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
	Single	0.53 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.53 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
	OPTICS	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
$\mathcal{C}_4$	Ward	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Single	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	OPTICS	0.45 ± 0.14	-0.02 ± 0.06	0.08 ± 0.08	0.38 ± 0.11	0.01 ± 0.05	0.11 ± 0.08
$\mathcal{C}_5$	Ward	0.20 ± 0.02	0.04 ± 0.00	0.24 ± 0.01	0.15 ± 0.02	0.03 ± 0.00	0.20 ± 0.01
	Single	0.88 ± 0.00	0.31 ± 0.03	0.29 ± 0.03	0.89 ± 0.00	0.30 ± 0.03	0.27 ± 0.02
	OPTICS	0.71 ± 0.03	0.18 ± 0.07	0.18 ± 0.04	0.68 ± 0.06	0.07 ± 0.07	0.11 ± 0.04
$\mathcal{C}_6$	Ward	0.69 ± 0.02	0.48 ± 0.03	0.64 ± 0.03	0.76 ± 0.12	0.58 ± 0.22	0.67 ± 0.12
	Single	0.86 ± 0.02	0.60 ± 0.09	0.63 ± 0.08	0.82 ± 0.02	0.46 ± 0.12	0.52 ± 0.11
	OPTICS	0.59 ± 0.07	0.21 ± 0.09	0.44 ± 0.06	0.58 ± 0.05	0.19 ± 0.06	0.45 ± 0.04

Table 9

Results of clustering nodes that belong to gold clusters whose size is greater or equal to 20 for graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Given a gold clustering, gray cells indicate the best results among clustering algorithms and underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$

		$\mathcal{G}_0$			$\mathcal{G}_5$		
		ACC	ARI	NMI	ACC	ARI	NMI
$\mathcal{C}_0$	Ward	0.17 ± 0.01	0.04 ± 0.00	0.32 ± 0.01	0.17 ± 0.02	0.04 ± 0.00	0.31 ± 0.01
	Single	0.79 ± 0.08	0.64 ± 0.11	0.54 ± 0.05	0.86 ± 0.01	0.69 ± 0.01	0.57 ± 0.01
	OPTICS	0.45 ± 0.03	0.09 ± 0.02	0.17 ± 0.01	0.50 ± 0.01	0.13 ± 0.01	0.19 ± 0.01
$\mathcal{C}_1$	Ward	0.15 ± 0.01	0.03 ± 0.00	0.31 ± 0.01	0.15 ± 0.01	0.03 ± 0.00	0.30 ± 0.00
	Single	0.64 ± 0.22	0.38 ± 0.19	0.45 ± 0.06	0.82 ± 0.01	0.58 ± 0.03	0.52 ± 0.03
	OPTICS	0.47 ± 0.02	0.08 ± 0.01	0.20 ± 0.01	0.51 ± 0.02	0.11 ± 0.03	0.20 ± 0.01
$\mathcal{C}_2$	Ward	0.98 ± 0.00	0.98 ± 0.02	0.99 ± 0.00	0.98 ± 0.00	0.99 ± 0.01	0.99 ± 0.00
	Single	0.97 ± 0.03	0.95 ± 0.05	0.98 ± 0.01	0.98 ± 0.00	0.98 ± 0.01	0.99 ± 0.00
	OPTICS	0.69 ± 0.01	0.44 ± 0.04	0.78 ± 0.01	0.73 ± 0.03	0.48 ± 0.04	0.81 ± 0.02
$\mathcal{C}_3$	Ward	0.92 ± 0.06	0.89 ± 0.08	0.95 ± 0.03	0.89 ± 0.05	0.84 ± 0.08	0.93 ± 0.03
	Single	0.91 ± 0.05	0.87 ± 0.06	0.95 ± 0.03	0.88 ± 0.07	0.84 ± 0.09	0.93 ± 0.04
	OPTICS	0.89 ± 0.07	0.87 ± 0.08	0.94 ± 0.08	0.92 ± 0.06	0.90 ± 0.09	0.95 ± 0.04
$\mathcal{C}_4$	Ward	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Single	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	OPTICS	0.29 ± 0.05	0.01 ± 0.01	0.09 ± 0.02	0.34 ± 0.05	0.03 ± 0.01	0.11 ± 0.02
$\mathcal{C}_5$	Ward	0.12 ± 0.01	0.02 ± 0.00	0.21 ± 0.01	0.10 ± 0.00	0.01 ± 0.00	0.17 ± 0.00
	Single	0.85 ± 0.01	0.32 ± 0.09	0.28 ± 0.06	0.86 ± 0.00	0.20 ± 0.03	0.27 ± 0.02
	OPTICS	0.48 ± 0.02	0.05 ± 0.01	0.10 ± 0.01	0.52 ± 0.02	0.05 ± 0.02	0.09 ± 0.01
$\mathcal{C}_6$	Ward	0.56 ± 0.05	0.39 ± 0.10	0.67 ± 0.02	0.50 ± 0.06	0.29 ± 0.08	0.65 ± 0.03
	Single	0.64 ± 0.07	0.43 ± 0.13	0.62 ± 0.05	0.78 ± 0.01	0.67 ± 0.06	0.71 ± 0.03
	OPTICS	0.44 ± 0.03	0.08 ± 0.03	0.38 ± 0.02	0.47 ± 0.05	0.08 ± 0.08	0.37 ± 0.05

Table 10

Results of clustering nodes that belong to gold clusters whose size is greater or equal to 10 for graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Given a gold clustering, gray cells indicate the best results among clustering algorithms and underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$

		$\mathcal{G}_0$			$\mathcal{G}_5$		
		ACC	ARI	NMI	ACC	ARI	NMI
$\mathcal{C}_0$	Ward	0.14 ± 0.01	0.02 ± 0.00	0.29 ± 0.01	0.13 ± 0.01	0.02 ± 0.00	0.28 ± 0.02
	Single	0.66 ± 0.17	0.53 ± 0.22	0.52 ± 0.06	0.74 ± 0.15	0.61 ± 0.16	0.54 ± 0.06
	OPTICS	0.25 ± 0.02	0.02 ± 0.01	0.12 ± 0.01	0.27 ± 0.01	0.03 ± 0.01	0.11 ± 0.01
$\mathcal{C}_1$	Ward	0.13 ± 0.01	0.01 ± 0.00	0.28 ± 0.01	0.14 ± 0.01	0.01 ± 0.00	0.27 ± 0.01
	Single	0.41 ± 0.12	0.18 ± 0.07	0.41 ± 0.02	0.72 ± 0.15	0.53 ± 0.14	0.52 ± 0.04
	OPTICS	0.28 ± 0.01	0.02 ± 0.00	0.13 ± 0.01	0.28 ± 0.00	0.02 ± 0.00	0.13 ± 0.01
$\mathcal{C}_2$	Ward	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	Single	0.99 ± 0.01	0.99 ± 0.02	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	OPTICS	0.63 ± 0.01	0.31 ± 0.01	0.67 ± 0.01	0.62 ± 0.01	0.29 ± 0.01	0.66 ± 0.01
$\mathcal{C}_3$	Ward	0.92 ± 0.00	0.90 ± 0.10	0.94 ± 0.05	0.86 ± 0.04	0.81 ± 0.05	0.89 ± 0.02
	Single	0.90 ± 0.07	0.88 ± 0.12	0.93 ± 0.05	0.83 ± 0.05	0.77 ± 0.08	0.88 ± 0.04
	OPTICS	0.75 ± 0.02	0.58 ± 0.03	0.78 ± 0.02	0.72 ± 0.03	0.49 ± 0.07	0.73 ± 0.05
$\mathcal{C}_4$	Ward	0.99 ± 0.00	0.90 ± 0.07	0.86 ± 0.08	0.99 ± 0.00	0.91 ± 0.05	0.88 ± 0.04
	Single	0.98 ± 0.01	0.83 ± 0.10	0.78 ± 0.15	0.99 ± 0.00	0.88 ± 0.05	0.85 ± 0.07
	OPTICS	0.18 ± 0.03	0.01 ± 0.01	0.06 ± 0.01	0.18 ± 0.01	0.00 ± 0.00	0.06 ± 0.01
$\mathcal{C}_5$	Ward	0.09 ± 0.01	0.01 ± 0.00	0.18 ± 0.01	0.07 ± 0.00	0.01 ± 0.00	0.14 ± 0.01
	Single	0.81 ± 0.01	0.31 ± 0.12	0.25 ± 0.08	0.82 ± 0.01	0.32 ± 0.08	0.26 ± 0.05
	OPTICS	0.27 ± 0.01	0.01 ± 0.00	0.06 ± 0.01	0.28 ± 0.01	0.01 ± 0.01	0.05 ± 0.01
$\mathcal{C}_6$	Ward	0.48 ± 0.03	0.24 ± 0.05	0.64 ± 0.01	0.44 ± 0.02	0.16 ± 0.03	0.60 ± 0.02
	Single	0.63 ± 0.07	0.56 ± 0.14	0.70 ± 0.04	0.74 ± 0.02	0.76 ± 0.05	0.76 ± 0.03
	OPTICS	0.37 ± 0.02	0.02 ± 0.01	0.29 ± 0.02	0.37 ± 0.02	0.03 ± 0.01	0.29 ± 0.01

Table 11

Results of clustering nodes that belong to gold clusters whose size is greater or equal to 50 for $\mathcal{C}_0$ and all graphs. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Gray cells indicate the best result among clustering algorithms. Underlined values indicate the best result between graphs. ↓ indicates a lower value with regard to $\mathcal{G}_0$

			Ward		Single		OPTICS
$\mathcal{G}_0$	ACC		0.24 ± 0.02		0.84 ± 0.08		0.61 ± 0.05
	ARI		0.07 ± 0.01		0.66 ± 0.13		0.21 ± 0.08
	NMI		$\underline{0.37 \pm 0.01}$		0.59 ± 0.05		0.25 ± 0.04
$\mathcal{G}_1$	ACC		0.24 ± 0.02		0.86 ± 0.00	↓	0.58 ± 0.03
	ARI		0.07 ± 0.01		0.70 ± 0.02	↓	0.16 ± 0.03
	NMI	↓	0.35 ± 0.02	↓	0.58 ± 0.02	↓	0.23 ± 0.01
$\mathcal{G}_2$	ACC		0.24 ± 0.01		0.89 ± 0.02		$\underline{0.70 \pm 0.01}$
	ARI		0.07 ± 0.00		0.72 ± 0.03		$\underline{0.32 \pm 0.03}$
	NMI	↓	0.34 ± 0.01		0.61 ± 0.04		$\underline{0.28 \pm 0.01}$
$\mathcal{G}_3$	ACC	↓	0.22 ± 0.03		0.89 ± 0.02		0.63 ± 0.03
	ARI		0.07 ± 0.01		$\underline{0.75 \pm 0.02}$		0.29 ± 0.04
	NMI	↓	0.36 ± 0.01		0.63 ± 0.03		$\underline{0.28 \pm 0.02}$
$\mathcal{G}_4$	ACC	↓	0.23 ± 0.02	↓	0.80 ± 0.16		0.62 ± 0.02
	ARI		0.07 ± 0.01	↓	0.63 ± 0.21	↓	0.20 ± 0.03
	NMI	↓	0.36 ± 0.01	↓	0.58 ± 0.08	↓	0.24 ± 0.01
$\mathcal{G}_5$	ACC		$\underline{0.25 \pm 0.02}$		$\underline{0.90 \pm 0.00}$		0.68 ± 0.02
	ARI		0.07 ± 0.01		$\underline{0.75 \pm 0.01}$		0.27 ± 0.05
	NMI	↓	0.35 ± 0.01		$\underline{0.64 \pm 0.02}$		0.27 ± 0.02

Table 12

Results of clustering nodes that belong to gold clusters whose size is greater or equal to 20 for $\mathcal{C}_0$ and all graphs. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Gray cells indicate the best result among clustering algorithms. Underlined values indicate the best result between graphs. ↓ indicates a lower value with regard to $\mathcal{G}_0$

			Ward		Single		OPTICS
$\mathcal{G}_0$	ACC		0.17 ± 0.01		0.79 ± 0.08		0.45 ± 0.03
	ARI		$\underline{0.04 \pm 0.00}$		0.64 ± 0.11		0.09 ± 0.02
	NMI		$\underline{0.32 \pm 0.01}$		0.54 ± 0.05		0.17 ± 0.01
$\mathcal{G}_1$	ACC		$\underline{0.19 \pm 0.01}$		0.81 ± 0.01	↓	0.43 ± 0.02
	ARI		$\underline{0.04 \pm 0.00}$		0.64 ± 0.02	↓	0.07 ± 0.03
	NMI		$\underline{0.32 \pm 0.01}$	↓	0.52 ± 0.01	↓	0.16 ± 0.02
$\mathcal{G}_2$	ACC		0.17 ± 0.01		0.81 ± 0.08		0.48 ± 0.01
	ARI		$\underline{0.04 \pm 0.00}$	↓	0.63 ± 0.09		0.11 ± 0.02
	NMI	↓	0.30 ± 0.01		0.54 ± 0.05		0.17 ± 0.01
$\mathcal{G}_3$	ACC	↓	0.15 ± 0.01		0.81 ± 0.06		0.46 ± 0.01
	ARI	↓	0.03 ± 0.00		0.64 ± 0.12		0.12 ± 0.02
	NMI		$\underline{0.32 \pm 0.01}$		0.55 ± 0.05		0.18 ± 0.01
$\mathcal{G}_4$	ACC		0.17 ± 0.01	↓	0.69 ± 0.22	↓	0.43 ± 0.02
	ARI	↓	0.03 ± 0.00	↓	0.54 ± 0.24	↓	0.08 ± 0.02
	NMI		$\underline{0.32 \pm 0.01}$	↓	0.52 ± 0.08		0.17 ± 0.01
$\mathcal{G}_5$	ACC		0.17 ± 0.02		$\underline{0.86 \pm 0.01}$		$\underline{0.50 \pm 0.01}$
	ARI		$\underline{0.04 \pm 0.00}$		$\underline{0.69 \pm 0.01}$		$\underline{0.13 \pm 0.01}$
	NMI	↓	0.31 ± 0.01		$\underline{0.57 \pm 0.01}$		$\underline{0.19 \pm 0.01}$

Table 13

Results of clustering nodes that belong to gold clusters whose size is greater or equal to 10 for $\mathcal{C}_0$ and all graphs. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Gray cells indicate the best result among clustering algorithms. Underlined values indicate the best result between graphs. ↓ indicates a lower value with regard to $\mathcal{G}_0$

			Ward		Single		OPTICS
$\mathcal{G}_0$	ACC		0.14 ± 0.01		0.66 ± 0.17		0.25 ± 0.02
	ARI		0.02 ± 0.00		0.53 ± 0.22		0.02 ± 0.01
	NMI		0.29 ± 0.01		0.52 ± 0.06		$\underline{0.12 \pm 0.01}$
$\mathcal{G}_1$	ACC		$\underline{0.15 \pm 0.01}$		0.73 ± 0.10		0.25 ± 0.01
	ARI		0.02 ± 0.00		0.58 ± 0.13		0.02 ± 0.01
	NMI		$\underline{0.30 \pm 0.01}$	↓	0.51 ± 0.03		$\underline{0.12 \pm 0.01}$
$\mathcal{G}_2$	ACC	↓	0.12 ± 0.01	↓	0.62 ± 0.16		$\underline{0.27 \pm 0.01}$
	ARI		0.02 ± 0.00	↓	0.47 ± 0.19		$\underline{0.03 \pm 0.01}$
	NMI	↓	0.26 ± 0.01	↓	0.48 ± 0.05	↓	0.11 ± 0.00
$\mathcal{G}_3$	ACC	↓	0.12 ± 0.00		0.70 ± 0.18		0.26 ± 0.01
	ARI		0.02 ± 0.00		0.58 ± 0.23		$\underline{0.03 \pm 0.01}$
	NMI	↓	0.28 ± 0.01		0.52 ± 0.06		$\underline{0.12 \pm 0.01}$
$\mathcal{G}_4$	ACC		0.14 ± 0.01	↓	0.56 ± 0.18		0.25 ± 0.01
	ARI		0.02 ± 0.00	↓	0.42 ± 0.20		0.02 ± 0.00
	NMI		0.29 ± 0.01	↓	0.50 ± 0.06		$\underline{0.12 \pm 0.00}$
$\mathcal{G}_5$	ACC	↓	0.13 ± 0.01		$\underline{0.74 \pm 0.15}$		$\underline{0.27 \pm 0.01}$
	ARI		0.02 ± 0.00		0.61 ± 0.16		$\underline{0.03 \pm 0.01}$
	NMI	↓	0.28 ± 0.02		0.54 ± 0.06	↓	0.11 ± 0.01

References

[1] M. Ankerst, M.M. Breunig, H. Kriegel and J. Sander, OPTICS: Ordering points to identify the clustering structure, in: *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data*, Philadelphia, Pennsylvania, USA, June 1–3, 1999, ACM Press, 1999, pp. 49–60. doi:10.1145/304182.304187.

[2] F. Baader et al. (eds), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003.

[3] T. Berners-Lee, J. Hendler, O. Lassila et al., The semantic web, *Scientific American* **284**(5) (2001), 28–37.

[4] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 2787–2795.

[5] H. Cai, V.W. Zheng and K.C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, *IEEE Trans. Knowl. Data Eng.* **30**(9) (2018), 1616–1637. doi:10.1109/TKDE.2018.2807452.

[6] K.E. Caudle et al., Incorporation of pharmacogenomics into routine clinical practice: The Clinical Pharmacogenetics Implementation Consortium (CPIC) guideline development process, *Current Drug Metabolism* **15**(2) (2014), 209–217. doi:10.2174/1389200215666140130124910.

[7] J. Chen, P. Hu, E. Jiménez-Ruiz, O.M. Holter, D. Antonyrajah and I. Horrocks, OWL2Vec*: Embedding of OWL ontologies, *Machine Learning* **110**(7) (2021), 1813–1845. doi:10.1007/s10994-021-05997-6.

[8] J. Chen, E. Jiménez-Ruiz, I. Horrocks, D. Antonyrajah, A. Hadian and J. Lee, Augmenting ontology alignment by semantic embedding and distant supervision, in: *The Semantic Web – 18th International Conference, ESWC 2021*, Virtual Event, June 6–10, 2021, Proceedings, R. Verborgh, K. Hose, H. Paulheim, P. Champin, M. Maleshkova, Ó. Corcho, P. Ristoski and M. Alam, eds, Lecture Notes in Computer Science, Vol. 12731, Springer, 2021, pp. 392–408. doi:10.1007/978-3-030-77385-4_23.

[9] A. Coulet and M. Smaïl-Tabbone, Mining electronic health records to validate knowledge in pharmacogenomics, *ERCIM News* **2016**(104) (2016).

[10] C. d'Amato, N.F. Quatraro and N. Fanizzi, Injecting background knowledge into embedding models for predictive tasks on knowledge graphs, in: *The Semantic Web – 18th International Conference, ESWC 2021*, Virtual Event, June 6–10, 2021, Proceedings, R. Verborgh, K. Hose, H. Paulheim, P. Champin, M. Maleshkova, Ó. Corcho, P. Ristoski and M. Alam, eds, Lecture Notes in Computer Science, Vol. 12731, Springer, 2021, pp. 441–457. doi:10.1007/978-3-030-77385-4_26.

[11] V.P. Dwivedi, C.K. Joshi, T. Laurent, Y. Bengio and X. Bresson, Benchmarking graph neural networks, *CoRR* **abs/2003.00982** (2020).

[12] J. Euzenat and P. Shvaiko, *Ontology Matching*, 2nd edn, Springer, 2013. ISBN 978-3-642-38720-3.

[13] N. Frosst, N. Papernot and G.E. Hinton, Analyzing and improving representations with the soft nearest neighbor loss, in: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, Long Beach, California, USA, 9–15 June 2019, Proceedings of Machine Learning Research, Vol. 97, PMLR, 2019, pp. 2012–2020.

[14] T.R. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition* **5**(2) (1993), 199–220. doi:10.1006/knac.1993.1008.

[15] R.V. Guha, Towards a model theory for distributed representations, in: *2015 AAAI Spring Symposia*, Stanford University, Palo Alto, California, USA, March 22–25, 2015, AAAI Press, 2015, http://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10220.

[16] V. Gutiérrez-Basulto and S. Schockaert, From knowledge graph embedding to ontology embedding? An analysis of the compatibility between vector space representations and rules, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018*, Tempe, Arizona, 30 October–2 November 2018, AAAI Press, 2018, pp. 379–388.

[17] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, J.E.L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.N. Ngomo, S.M. Rashid, A. Rula, L. Schmelzeisen, J.F. Sequeda, S. Staab and A. Zimmermann, Knowledge graphs, *CoRR* **abs/2003.02320** (2020). https://arxiv.org/abs/2003.02320 abs/2003.02320.

[18] A. Iana and H. Paulheim, More is not always better: The negative impact of A-box materialization on RDF2vec knowledge graph embeddings, in: *Proceedings of the CIKM 2020 Workshops Co-Located with 29th ACM International Conference on Information and Knowledge Management (CIKM 2020)*, Galway, Ireland, October 19–23, 2020, S. Conrad and I. Tiddi, eds, CEUR Workshop Proceedings, Vol. 2699, CEUR-WS.org, 2020, http://ceur-ws.org/Vol-2699/paper05.pdf.

[19] D.P. Kingma and J. Ba, Adam: A method for stochastic optimization, in: *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015.

[20] T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in: *5th International Conference on Learning Representations, ICLR 2017*, Toulon, France, April 24–26, 2017, Conference Track Proceedings, OpenReview.net, 2017.

[21] P. Monnin, M. Couceiro, A. Napoli and A. Coulet, Knowledge-based matching of n-ary tuples, in: *Ontologies and Concepts in Mind and Machine – 25th International Conference on Conceptual Structures, ICCS 2020*, Bolzano, Italy, September 18–20, 2020, Proceedings, M. Alam, T. Braun and B. Yun, eds, Lecture Notes in Computer Science, Vol. 12277, Springer, 2020, pp. 48–56. doi:10.1007/978-3-030-57855-8_4.

[22] P. Monnin, J. Legrand, G. Husson, P. Ringot, A. Tchechmedjiev, C. Jonquet, A. Napoli and A. Coulet, PGxO and PGxLOD: A reconciliation of pharmacogenomic knowledge of various provenances, enabling further comparison, *BMC Bioinformatics* **20-S**(4) (2019), 139:1–139:16. doi:10.1186/s12859-019-2693-9.

[23] P. Monnin, C. Raïssi, A. Napoli and A. Coulet, Knowledge reconciliation with graph convolutional networks: Preliminary results, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-Located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019)*, Portoroz, Slovenia, June 2, 2019, CEUR Workshop Proceedings, Vol. 2377, CEUR-WS.org, 2019, pp. 47–56.

[24] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proceedings of the IEEE* **104**(1) (2016), 11–33. doi:10.1109/JPROC.2015.2483592.

[25] N. Noy, A. Rector, P. Hayes and C. Welty, Defining N-ary relations on the semantic web, *W3C Working Group Note* **12**(4) (2006).

[26] N. Pang, W. Zeng, J. Tang, Z. Tan and X. Zhao, Iterative entity alignment with improved neural attribute embedding, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-Located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019)*, Portoroz, Slovenia, June 2, 2019, CEUR Workshop Proceedings, Vol. 2377, CEUR-WS.org, 2019, pp. 41–46.

[27] H. Paulheim, Make embeddings semantic again! in: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks Co-Located with 17th International Semantic Web Conference (ISWC 2018)*, Monterey, USA, October 8th – to – 12th, 2018, CEUR Workshop Proceedings, Vol. 2180, CEUR-WS.org, 2018.

[28] P. Ristoski and H. Paulheim, RDF2Vec: RDF graph embeddings for data mining, in: *The Semantic Web – ISWC 2016 – 15th International Semantic Web Conference*, Kobe, Japan, October 17–21, 2016, Proceedings, Part I, Lecture Notes in Computer Science, Vol. 9981, 2016, pp. 498–514. doi:10.1007/978-3-319-46523-4_30.

[29] M.S. Schlichtkrull, T.N. Kipf, P. Bloem, R. van den Berg, I. Titov and M. Welling, Modeling relational data with graph convolutional networks, in: *The Semantic Web – 15th International Conference, ESWC 2018*, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings, Lecture Notes in Computer Science, Vol. 10843, Springer, 2018, pp. 593–607. doi:10.1007/978-3-319-93417-4_38.

[30] L. Serafini and A.S. d'Avila Garcez, Logic tensor networks: Deep learning and logical reasoning from data and knowledge, in: *Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy '16) Co-Located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016)*, New York City, NY, USA, July 16–17, 2016, CEUR Workshop Proceedings, Vol. 1768, CEUR-WS.org, 2016.

[31] P. Wang, J. Han, C. Li and R. Pan, Logic attention based neighborhood aggregation for inductive knowledge graph embedding, in: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, the Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, the Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, Honolulu, Hawaii, USA, January 27–February 1, 2019, AAAI Press, 2019, pp. 7152–7159. doi:10.1609/aaai.v33i01.33017152.

[32] Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge graph embedding: A survey of approaches and applications, *IEEE Trans. Knowl. Data Eng.* **29**(12) (2017), 2724–2743. doi:10.1109/TKDE.2017.2754499.

[33] Z. Wang, Q. Lv, X. Lan and Y. Zhang, Cross-lingual knowledge graph alignment via graph convolutional networks, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, October 31–November 4, 2018, Association for Computational Linguistics, 2018, pp. 349–357. doi:10.18653/v1/d18-1032.

[34] M. Whirl-Carrillo, E.M. McDonagh, J.M. Hebert, L. Gong, K. Sangkuhl, C.F. Thorn, R.B. Altman and T.E. Klein, Pharmacogenomics knowledge for personalized medicine, *Clinical Pharmacology and Therapeutics* **92**(4) (2012), 414. doi:10.1038/clpt.2012.96.