From Axioms over Graphs to Vectors, and Back Again: Evaluating the Properties of Graph-based Ontology Embeddings

Fernando Zhapa-Camacho[1], Robert Hoehndorf[1,*]

[1]*Computational Bioscience Research Center, Computer, Electrical & Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology, 4700 KAUST, 23955 Thuwal, Saudi Arabia*

Abstract

Several approaches have been developed that generate embeddings for Description Logic ontologies and use these embeddings in machine learning. One approach of generating ontologies embeddings is by first embedding the ontologies into a graph structure, i.e., introducing a set of nodes and edges for named entities and logical axioms, and then applying a graph embedding to embed the graph in $\mathbb{R}^n$. Methods that embed ontologies in graphs (graph projections) have different formal properties related to the type of axioms they can utilize, whether the projections are invertible or not, and whether they can be applied to asserted axioms or their deductive closure. We analyze, qualitatively and quantitatively, several graph projection methods that have been used to embed ontologies, and we demonstrate the effect of the properties of graph projections on the performance of predicting axioms from ontology embeddings. We find that there are substantial differences between different projection methods, and both the projection of axioms into nodes and edges as well ontological choices in representing knowledge will impact the success of using ontology embeddings to predict axioms.

1. Introduction

Ontologies are widely used to integrate and standardize data across databases. In recent years, ontologies and their associated knowledge in databases have been used in machine learning to constrain the solution space by the ontology structure, with several applications in the biomedical domain [1]. One form of using ontologies in machine learning tasks is based on graphs [2, 3, 4]. We use the term *graph projection* to refer to the transformation of an ontology into a graph. With recent developments in machine learning over graphs [5], several approaches to project ontologies into graphs have emerged. These approaches are able to capture the ontology structure at some level and have been evaluated on tasks such as similarity computation, link (axiom) prediction [4], or ontology alignment [6].


Graphs as a form of representation of knowledge have been studied for several decades [7, 8]. Existential Graphs (EGs) were proposed by C. S. Peirce as a means to depict logical expressions through diagrams. EGs enabled the representation of first order logic formulas [9]. Semantic Networks, like EGs, are graphs which contain representations of concepts as nodes and relations between concepts as edges [10, 11]. Conceptual graphs arose from both EGs and Semantic Networks, by leveraging the logical foundation of EGs with the properties of Semantic Networks [12]. The diagrammatic representation of logical expressions has not only been developed to make such expressions more readable and understandable, but also to enable certain operations, for example those that correspond to forms of computing entailments and reasoning. In Dau [13], the formalization of the diagrams of Existential Graphs and their use in logic calculus and reasoning is explored. Further work [14, 15] investigates the reasoning capabilities that existential and conceptual graphs can have for Description Logics.

There has been a renewed interest in graph-based representations of ontologies with the emergence of graph-based machine learning methods. Graph embedding methods and knowledge graph embeddings [5] have been developed to embed (knowledge) graphs in the $\mathbb{R}^n$ where gradient-based methods can be used to solve optimization problems that allow these embeddings to be used, for example, for the prediction of edges or for determining similarity between nodes. A graph projection embeds an ontology in a graph structure; this graph structure can then be used to generate embeddings of ontology entities in $\mathbb{R}^n$. Because graphs can be intermediate steps in generating ontology embeddings (in $\mathbb{R}^n$), it becomes important to investigate properties of graph projections. A graph projection is *total* if it uses of all the axioms in the ontology to generate a graph, and *partial* otherwise. Totality can be defined with respect to asserted axioms in an ontology, or with respect to the deductive closure. Relating a (predicted) edge in a graph to an axiom in an ontology requires that the graph projection is *injective* (i.e., that different axioms induce different subgraphs). This is not true for every graph projection method. Furthermore, graph projections can project axioms into single edges or into subgraphs (potentially with multiple edges); some graph embedding methods are designed to predict single edges and not subgraphs, and graph projections that generate subgraphs may therefore not be suitable to be used jointly with those graph embeddings. Analyzing these properties is crucial to understanding how the information provided by ontologies is utilized by each method and what their limitations are; understanding the limitations enables the development of new methods that can address them. While graph-based embeddings are not the only way to embed ontologies, graphs are widely used due to the large availability of graph-based machine learning methods. Here, we analyze graph projections and their properties in the context of embedding ontologies in $\mathbb{R}^n$. Our contributions are the following: (a) We provide a qualitative analysis of graph projection methods with respect to totality, injectivity, and use of deductive closure; (b) We quantitatively analyze the effect of properties of graph projections in experiments that predict axioms in the deductive closure of ontologies.

2. Ontology embeddings and graph embeddings

An embedding is a structure-preserving mapping between two mathematical structures. A graph-based embedding is a two-step process where an ontology is first embedded into a graph,
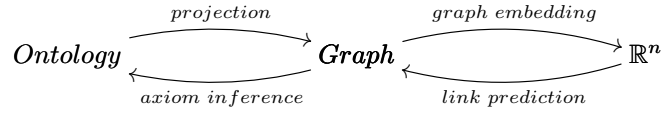
Figure 1: Ontology embedding (left to right) and inference (right to left) processes.

and then a graph-embedding is used to embed the graph in the $\mathbb{R}^n$. We call the first embedding an "graph projection" or "projection" and the second embedding a "graph embedding". Within $\mathbb{R}^n$, inferences may then be done approximately and translated back to the ontology. The inference computation is done by computing plausibility of edges (links) in the graph that were generated by a query axiom (Figure 1).

While the machine learning and the Semantic Web communities have spent substantial effort on designing methods that achieve the second part (graph embeddings), the first part (graph projection) remains rather unexplored. However, the property of the graph projection itself has consequences for the types of operations that can be performed in the embedding space ($\mathbb{R}^n$). The main question that we investigate is how the mathematical properties of graph projections affect the inference task. We analyze totality and injectivity, as well as, the use of semantic information in the process of graph generation.

3. Graph projection methods

3.1. Preliminaries

An ontology $\mathcal{O} = (\Sigma, Ax)$ is a tuple consisting of a signature Σ and a set of (Description Logic) axioms Ax. The signature $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ consists of a set of class names $\mathbf{C}$, a set of role names $\mathbf{R}$, a set of individual names $\mathbf{I}$. The set Ax is a set of formulas in a language $\mathcal{L}(\Sigma)$. We consider here only ontologies where the set of axioms Ax are formulated in a Description Logic [16] language (see Appendix B). The deductive closure $\mathcal{O}^{\vdash}$ of $\mathcal{O}$ is defined as $\mathcal{O}^{\vdash} = \{\phi \mid Ax \vdash \phi\}$.

Relational graphs are intermediate structures during the ontology embedding process. A relational graph G is a triple (V, E, L), where a V is a set of vertices, L is a set of edge labels, and $E \subseteq V \times L \times V$ is a set of edges between vertices and with a label from L.

The prediction of edges and the prediction of axioms can be formulated as ranking problems where edges and axioms are scored using a scoring function, with the intended meaning that a higher-scoring edge or axiom is preferred over a lower-scoring edge or axiom.

A graph projection for $\mathcal{O} = (\Sigma, Ax)$ is a function p that maps an ontology into a relational graph $G = (V, E, L)$ such that $\mathbf{C} \cup \mathbf{R} \cup \mathbf{I} \subseteq V \cup L$ (i.e., each class, individual, and role name is represented as a node or edge label in G) and for each $a \in Ax$, $p(a) \subseteq E$ (i.e., p maps an axiom onto a subgraph of G). The function p may be total or partial with respect to $\mathcal{O}$ depending on whether it is defined for all axioms in the set Ax or only for some axioms. p may also be total or partial with respect to the deductive closure of $\mathcal{O}$ based on whether it is defined for all axioms in $\mathcal{O}^{\vdash}$. We call p simple if the cardinality of $p(a)$ is 1 for all axioms a, i.e., if the projection function maps each axiom onto exactly one edge. p takes an axiom as argument

and generates a graph. If p is injective, it will generate different subgraphs for different axioms and the projection function is therefore invertible, i.e., from a (predicted) subgraph it becomes possible to generate a corresponding axiom.

Here, we analyze different projection methods developed for ontologies and their use in machine learning: (i) taxonomic projection, OWL2Vec* and DL2Vec [4, 3], Onto2Graph [2], and the graphs constructed from the RDF rendering of OWL. Examples of graphs generated by each projection method can be found in Appendix D.

3.2. Taxonomy projection

A taxonomy projection generates a graph from subclass axioms between named classes. From ontology $\mathcal{O}$, the axioms used are those of the form $C \sqsubseteq D$ where C, D are class names; the projection function is simple and generates a single edge, from C to D. The projection is partial if $\mathcal{O}$ contains axioms besides $C \sqsubseteq D$, and total otherwise; if it is total for O, the projection is also total for $\mathcal{O}^\vdash$. Furthermore, this projection is both simple and injective, which means we can infer a single axiom from a predicted edge.

3.3. OWL2Vec*

OWL2Vec* [4] (and variants such as DL2Vec [3]) targets the Description Logic $\mathcal{SROIQ}$ [17] underlying OWL 2 DL [18]. The projection rules for the graph component in OWL2Vec* are shown in Appendix C. In addition to the taxonomic structure, the OWL2Vec* projection includes projections for axioms involving complex class descriptions, including quantifiers and roles. The role names used with quantifiers are used as labels in the relational graph. For example, an axiom of the form $Parent \sqsubseteq \exists hasChild.Person$ is transformed into the edge $(Parent, hasChild, Person)$, which relates two nodes using a labeled edge corresponding to the role $hasChild$. The OWL2Vec* projection does not differentiate between quantifiers, i.e., $p_{owl2vec*}(A \sqsubseteq \exists R.B) = p_{owl2vec*}(A \sqsubseteq \forall R.B) = \{(A, R, B)\}$. Similarly, union ($\sqcap$) and intersection ($\sqcup$) operators are not distinguished, i.e., $p_{owl2vec*}(A \sqsubseteq \exists R.(B \sqcap C)) = p_{owl2vec*}(A \sqsubseteq \exists R.(B \sqcup C)) = \{(A, R, B), (A, R, C)\}$. The OWL2Vec* projection is a partial function in both the set of axioms Ax and the deductive closure $\mathcal{O}^\vdash$ because concept descriptions including operators such as negation ($\neg$) are not defined for $p_{owl2vec*}$. The OWL2Vec* projection is not injective because different axioms will produce the same edge, such as when quantifiers are not distinguished. When inferring an axiom from a graph, several axioms would obtain exactly the same score because they generate the same edge or set of edges (and therefore the inverse of $p_{owl2vec*}$ produces a set of axioms instead of a single axiom). For example, if we query axioms $A \sqsubseteq \exists R.B$ or $A \sqsubseteq \forall R.B$, both will receive the score given to the edge (A, R, B). Moreover, $p_{owl2vec*}$ is not simple; for example, the cardinality of $p_{owl2vec*}(A \sqsubseteq \exists R.(B \sqcap C))$ is greater than 1 because the projection maps to edges $\{(A, R, B), (A, R, C)\}$ (Figure 2a).

3.4. Syntax trees and RDF graphs

We can use the syntactic representation of Ax directly to generate graphs using, for example, syntax trees. One option is to use the graph-based rendering of the OWL syntax in RDF [19], which is a representation of the syntax tree underlying the Description Logic axioms in OWL.
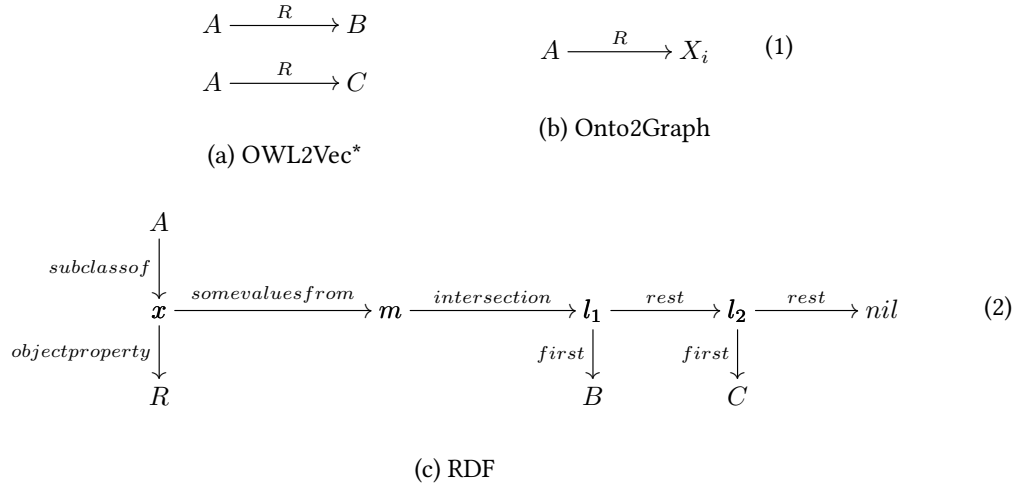
The main advantage of using a syntax tree as a graph representation is that the totality of the projection function are guaranteed, both for axioms in $\mathcal{O}$ and the deductive closure $\mathcal{O}^\vdash$. However, nodes in the relational graph generated from the projection no longer correspond to named entities in the signature of $\mathcal{O}$ (due to the introduction of internal nodes in the syntax tree, or blank nodes in RDF). For example, to represent the axiom $C \sqcap D \sqsubseteq \bot$, four blank nodes are created (Appendix Figure 3). Similarly, to represent the axioms $A \sqsubseteq \exists R.(B \sqcap C)$, five blank nodes are introduced (Figure 2c). A major difference to methods such as OWL2Vec* is that single axioms do not correspond to a single edge but rather to a subgraph, i.e., the projection is, in general, not simple. This raises an issue during axiom inference when axioms need to be generated and scored, because the score will be computed from subgraphs instead of single edges.

3.5. Relational axiom patterns

Onto2Graph [2] is a method that implements graph projection based on (relational) ontology design patterns [20]. In the past, ontologies in the biomedical domain were often represented as directed acyclic graphs [21] and not using a formal language based on a model-theoretic semantics. It took several years before the graph representation of the ontologies was put on a formal semantic foundation [22, 23, 20]. Two approaches provided this foundation, one based on a correspondence between edges and axioms of a certain type as in the OWL2Vec* projection [22, 23], and others based on relational ontology design patterns [20]. The OBO Relation Ontology [24] contains a large number of such patterns used in biomedical ontologies.

A relational pattern is defined using variables that stand for symbols and are used to define edges in a graph. An example of a relational pattern is $?X \sqsubseteq \exists ?R.?Y$ from which an edge $(?X, ?R, ?Y)$ can be created in a graph. More commonly, patterns that use specific roles are used, such as $?X \sqsubseteq \exists \text{part-of}.?Y$ to create an edge labeled "part-of" from $?X$ to $?Y$. Relational patterns are flexible and can be defined for arbitrary axioms. For example, a set of "disjointness" edges can be created from an axiom pattern such as $?X \sqcap ?Y \sqsubseteq \bot$. In Figure 2b, the pattern $A \sqsubseteq \exists R.?X$ is selected, where $?X \equiv B \sqcap C$ is the query to apply to the ontology. Given a relational pattern, an ontology can be queried for pairs or triples that satisfy these patterns in quadratic ($O(|\mathbf{C}|^2)$) or cubic ($O(|\mathbf{C}|^2 |\mathbf{R}|)$) time, respectively, by substituting every class name and role name in the variables of the relational patterns. This querying can be applied to either the set of axioms Ax or their deductive closure. The Onto2Graph [2] method implements an algorithm to generate graph edges $(?X, R, ?Y)$ more efficiently for some axiom patterns.

$p_{onto2graph}$ is a partial function unless a pattern for every type of axiom is defined (and the patterns are only generated from asserted axiom and not their deductive closure). $p_{onto2graph}$ may be injective if the patterns are defined so that different axioms map to different subgraphs. However, the OWL2Vec* projection function can be seen as a special case of relational patterns (where no domain knowledge is used to specify patterns), and since the OWL2Vec* projection is not injective, relational patterns may also not be injective. In general, $p_{onto2graph}$ is not simple as multiple edges can be generated from a single axiom; however, in practice, the Onto2Graph projection is usually simple (i.e., the library of relational patterns defined in the OBO Relation Ontology [24] implements only simple projections).

$$A \xrightarrow{\quad R \quad} B$$
$$A \xrightarrow{\quad R \quad} C$$
$$A \xrightarrow{\quad R \quad} X_i \qquad (1)$$

(b) Onto2Graph

(a) OWL2Vec*

$$A$$
$$\downarrow {\scriptstyle subclassof}$$
$$x \xrightarrow{\quad somevaluesfrom \quad} m \xrightarrow{\quad intersection \quad} l_1 \xrightarrow{\quad rest \quad} l_2 \xrightarrow{\quad rest \quad} nil \qquad (2)$$
$$\downarrow {\scriptstyle objectproperty} \qquad\qquad\qquad\qquad\qquad {\scriptstyle first} \downarrow \qquad {\scriptstyle first} \downarrow$$
$$R \qquad\qquad\qquad\qquad\qquad\qquad\qquad B \qquad\qquad C$$

(c) RDF

Figure 2: Graphs generated by each projection method on the axiom $A \sqsubseteq \exists R.(B \sqcap C)$. For the case of Onto2Graph, we can query every class $X_i \equiv B \sqcap C$ and generate the corresponding edge.

4. Machine learning with graph projections

4.1. Queries and axiom scoring

The main reason we investigate graph projections is due to the availability of machine learning methods for graphs. In particular, (knowledge) graph embeddings can be used for tasks such as determining similarity between nodes [3], to predict edges that may be added to a knowledge graph [25], or (approximately) answer complex queries corresponding to subgraphs of the knowledge graph [26, 27, 28]. Here, we investigate the impact of the properties of graph projections on machine learning with ontologies. Specifically, for graph edges (h, r, t) that can be added to the graph, knowledge graph embeddings can be used to define scores $(score(h, r, t))$, and we can use $score(h, r, t)$ to score axioms that may be added to ontology $\mathcal{O}$ ("axiom inference").

The main operation that allows us to score and infer axioms is the inverse of the projection function p. If p is simple and injective, its inverse p^{-1} will yield exactly one axiom a for an edge (h, r, t), and we can define $score(a) := score(h, r, t)$ to score axioms; we also refer to scoring an axiom as a "query". For example, to query the axiom $C \sqsubseteq \forall R.D$, we first project the axiom onto a graph edge (for example, the edge (C, R, D) using the OWL2Vec* projection); then, we determine the score of the edge (C, R, D) using a knowledge graph embedding method; and, finally, we apply the inverse of the projection function to determine $score(C \sqsubseteq \forall R.D)$. If the projection is not injective (such as the OWL2Vec* projection), multiple axioms are generated with the same score; if axioms are added to ontology $\mathcal{O}$ by ranking their scores, this needs to be considered. For example, the inverse of the OWL2Vec* projection for the edge (C, R, D) will produce a set of axioms containing at least $C \sqsubseteq \forall R.D$ and $C \sqsubseteq \exists R.D$, with the same score.

Non-simple projections produce multiple edges for single axioms, and computing the inverse requires determining a score for a subgraph and transferring it to the axiom. While there

are methods to directly score subgraphs using knowledge graph embeddings, we will use the arithmetic mean of the scores of each edge in the subgraph as the score of the subgraph.

4.2. Experimental setup

To test the performance of different projection methods, we test their ability to predict axioms in the deductive closure of an ontology. We evaluate axiom prediction in two different ways: (i) we generate embeddings using the original ontology ($\mathcal{O}$), and (ii) we generate embeddings from a reduced version of the ontology ($\mathcal{O}_{reduced}$) by removing some axioms. The test set consists of axioms that are in $\mathcal{O}^{\vdash}$ but not in $\mathcal{O}^{\vdash}_{reduced}$. The first case corresponds to computing entailments analogously to an automated reasoner, and the second case corresponds to "prediction" of statements that may hold true although they are not entailed; the second case may also be considered a form of approximate entailment [29].

We used two large biomedical ontologies for evaluation, the Gene Ontology [30] (GO) and the Food Ontology (FoodOn) [31]. In GO, we tested prediction of subclass (sub) axioms of the form $C \sqsubseteq D$ and axioms involving existential restrictions (ex) of the form $C \sqsubseteq \exists R.D$. We generated reduced ontologies GO_{sub} and GO_{ex} by randomly removing 10% of the sub and ex axioms, respectively. In contrast to GO, FoodOn axioms use both quantifiers ($\exists, \forall$). We used FoodOn to investigate the effect of injective projections by testing the methods on the axiom patterns $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$. Furthermore, to obtain a substantial difference between the $FoodOn^{\vdash}$ and $FoodOn^{\vdash}_{sub\_ex}$, we randomly removed 30% of the sub and ex axioms from FoodOn. To generate the deductive closure for all the ontologies and their reduced versions, we used the OWLAPI and the ELK reasoner [32]. ELK is a reasoner for OWL 2 EL ontologies. Other reasoners can be used, especially for FoodOn, where axioms can belong to more complex description logics than $\mathcal{EL}$ [16]. However, due to the complexity of reasoning in expressive description logics, we use ELK for our experiments. Projection methods can handle axioms involving existential and universal quantifiers, but axioms in the deductive closure will not involve universal restrictions due to the use of ELK.

To embed the graphs generated from projection methods, we first use the knowledge graph embedding method TransE [33] (see Appendix F for details). While there are many methods to embed knowledge graphs [5], we use TransE because it is a simple approach to embed graphs. We also use TransR [34] to evaluate the effect of a different graph embedding method.

In the evaluation, for every sample axiom $C \sqsubseteq D$ or $C \sqsubseteq \exists R.D$ in the testing set, we generate predictions for every other axiom $C \sqsubseteq D'$ or $C \sqsubseteq \exists R.D'$ for every named class D'. Then, we compute the rank of the positive axiom based on the score obtained from the projected graph. We report mean rank, hits at $\{1, 10, 100\}$ and ROC AUC. We report filtered metrics by not considering the predictions that exist in the deductive closure. We performed hyperparameter optimization (see Appendix G) and provide complete source code for our experiments at https://github.com/bio-ontology-research-group/ontology-graph-projections.

4.3. Evaluation results

We test the performance of ontology embeddings in two tasks. First, we test on prediction of plausible axioms that cannot be inferred but which may hold true given the other axioms. We

Table 1

Results of prediction of axioms of the form $C \sqsubseteq D$ and $C \sqsubseteq \exists R.D$ of different projection methods on different versions of the Gene Ontology.

| | Predictions of axioms $C \sqsubseteq D$ | | | | | | | | | |
| Method | GO | | | | | GO_{sub} | | | | |
	MR	H@1	H@10	H@100	AUC	MR	H@1	H@10	H@100	AUC
Onto2Graph-TransE	**2947.81**	0.41	0.85	12.32	**94.23**	4439.27	0.41	1.35	10.09	91.31
Onto2Graph-TransR	3599.95	0.03	0.74	6.86	92.95	**3704.74**	0.09	0.45	8.17	**92.75**
OWL2Vec*-TransE	4514.51	1.98	**10.03**	**33.87**	91.16	5374.64	**1.79**	**9.55**	34.34	89.48
OWL2Vec*-TransR	3083.20	**2.98**	5.40	22.71	93.96	4604.52	0.01	5.90	15.24	90.98
RDF-TransE	4158.77	0.36	2.85	10.03	91.86	4122.82	0.41	2.82	10.10	91.93
RDF-TransR	4116.13	0.04	0.62	5.41	91.94	4899.87	0.02	0.52	6.16	90.41

| | Predictions of axioms $C \sqsubseteq \exists R.D$ | | | | | | | | | |
| | GO | | | | | GO_{ex} | | | | |
	MR	H@1	H@10	H@100	AUC	MR	H@1	H@10	H@100	AUC
Onto2Graph-TransE	9250.77	**5.21**	14.26	25.53	81.93	10214.69	**4.26**	13.19	22.45	80.05
Onto2Graph-TransR	9430.42	0.00	6.28	7.45	81.61	10644.37	0.00	0.00	0.43	79.23
OWL2Vec*-TransE	**8956.46**	0.74	**21.60**	**28.09**	**82.51**	9037.64	0.21	**21.17**	**28.30**	**82.36**
OWL2Vec*-TransR	12746.62	0.00	0.64	4.04	75.09	13342.04	0.00	0.53	3.09	73.93
RDF-TransE	12240.94	1.49	3.51	5.74	76.08	12864.34	0.00	0.21	1.49	74.86
RDF-TransR	11976.59	0.00	0.21	4.04	76.65	10740.08	0.00	0.00	0.00	79.07

use embeddings from ontologies with removed axioms GO_{sub} and GO_{ex} and evaluate on axioms $C \sqsubseteq D$ and $C \sqsubseteq \exists R.D$ existing in the deductive closure of GO, but not in the deductive closure of GO_{sub} and GO_{ex}, respectively; this task is a form of "ontology completion" [4, 35]; we focus on axioms in the deductive closure instead of asserted axioms to evaluate whether the regularities hold semantically instead of merely syntactically. Second, we test on a deductive inference task (i.e., test the prediction of axioms in the deductive closure), similarly to an automated reasoner. We use embeddings from GO and to compare directly with the first task, we evaluated on the same test set of axioms used in the first task. Table 1 shows the results.

OWL2Vec* can parse complex axioms (i.e., $C \equiv D \sqcap \exists R.E$) and generates edges (C, subclassof, D) that Onto2Graph does not generate (given the relational patterns we employ). Furthermore, OWL2Vec* generates several inverse edges not created by Onto2Graph. These differences allow OWL2Vec* to rank axioms of type $C \sqsubseteq D$ higher in Hits@k metrics (See Appendix A). The RDF projection generates edges of the form (C, subclassof, D) where and C or D are not necessarily named classes but can be blank nodes. The presence on blank nodes adds noise when embedding RDF graphs, which causes lower values in Hits@k compared to other methods.

For axioms of type $C \sqsubseteq \exists R.D$, Onto2Graph and OWL2Vec* behave differently. For GO, both methods generate approximately the same number of edges (30,266 and 31,429) containing R roles as labels. However, only around half (18,339) of the edges are shared by both graphs. The remaining edges for Onto2Graph correspond to those generated by the reasoning process and for OWL2Vec* correspond to edges generated from subroles and inverse roles which are ignored by Onto2Graph. Inverse edges create cycles in the graph. These differences suffice so that OWL2Vec* outperforms Onto2Graph in almost all the metrics.

TransR embeds nodes and each relation in different spaces. In OWL2Vec* projections and Onto2Graph, axioms $C \sqsubseteq D$ and $C \sqsubseteq \exists R.D$ have the same graph structure ((C,subclassof,D)

Table 2
Prediction of axioms of the form $C \sqsubseteq \exists R.D$ by ranking over (Case A) axioms $\{C \sqsubseteq \exists R.D'|D' \in \mathbf{C}\}$ and (Case B) axioms $\{C \sqsubseteq \Box R.D'|D' \in \mathbf{C}, \Box \in \{\exists, \forall\}\}$. We performed this test on FoodOn.

Method	Case A				
	MR	H@1	H@10	H@100	AUC
Onto2Graph-TransE	**7332.68**	0.06	0.23	1.67	**78.51**
Onto2Graph-TransR	8798.84	0.04	0.56	2.85	74.21
OWL2Vec*-TransE	8361.63	**0.16**	**1.46**	**8.43**	75.49
OWL2Vec*-TransR	8003.53	0.00	0.14	5.37	76.54
RDF-TransE	8417.59	0.02	0.04	1.34	75.33
RDF TransR	13282.75	0.00	0.01	0.61	61.07
	Case B				
Onto2Graph-TransE	14676.41	0.02	0.15	0.91	78.49
Onto2Graph-TransR	17609.04	0.04	0.40	2.56	74.19
OWL2Vec*-TransE	16732.26	**0.08**	**0.73**	**5.25**	75.48
OWL2Vec*-TransR	16017.71	0.00	0.01	2.10	76.52
RDF	16605.67	0.01	0.03	0.74	75.66
RDF TransR	**14330.19**	0.00	0.00	0.08	**79.00**

and (C,R,D), respectively). TransR helps to differentiate the label "subclassof" from other labels in the graph, improving average ranking metrics such as Mean Rank and AUC. However, for axioms $C \sqsubseteq \exists R.D$, where a number of edge labels must be considered, TransR underperforms compared to TransE for OWL2Vec* and Onto2Graph, while improving for RDF projection.

Additionally, we also evaluated over FoodOn which contains more complex axioms to determine the effect of injectivity on predicting axioms, specifically axioms of the type $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$. As in the evaluation of GO, we generated embeddings for FoodOn and evaluated the performance on the prediction of axioms $C \sqsubseteq \exists R.D$ existing in the deductive closure of FoodOn but not in the deductive closure of FoodOn with some axioms removed (FoodOn$_{sub\_ex}$).

We evaluate the ranking of testing samples $C \sqsubseteq \exists R.D$ among a set of predictions. We have two cases: we rank axioms $C \sqsubseteq \exists R.D$ among all axioms $C \sqsubseteq \exists R.D'$ for all named classes D' (case A in Table 2), and, secondly, we rank axioms $C \sqsubseteq \exists R.D$ among axioms $C \sqsubseteq \Box R.D'$ for all named class D' and $\Box = \{\exists, \forall\}$ (case B in Table 2). Non-injective methods (such as OWL2Vec*) generate multiple axioms when inverting the projection of $C \sqsubseteq \exists R.D$, making it necessary to consider the scores of multiple axioms when evaluating axiom inference. We limit the choice of quantifier to only $\exists$ and $\forall$ and ignore cardinality restrictions We also evaluate Onto2Graph projections that are non-injective and project both $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$ onto the same edge (similarly to OWL2Vec*).

Obviously, methods such as OWL2Vec* and (non-injective) Onto2Graph decrease the performance from case A to case B. More specifically, the mean rank doubles because for every axiom $C \sqsubseteq \exists R.D$, another axiom ($C \sqsubseteq \forall R.D$) has the same score and is ranked at the same position. In the RDF projection, both $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$ will, usually, obtain different scores. Nevertheless, we observe that the mean ranks almost doubles between case A and B

when using TransE. This is because the subgraphs projected from $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$ differ only in the edge labels "somevaluesfrom" and "allvaluesfrom" (Appendix Figure 4), and, in our case, TransE generates similar embeddings for "somevaluesfrom" and "allvaluesfrom". We also included another experiment using TransR [34] instead of TransE; TransR uses a different embedding for relations which are embedded in a different space than nodes. Although the overall performance drops in case A, TransR represents relations corresponding to "somevaluesfrom" and "allvaluesfrom" differently, illustrated by an increase in mean rank from case A to case B, and an increase in AUC. In the future, further knowledge graph embedding approaches need to be evaluated. Furthermore, Onto2Graph produces a lower mean rank than OWL2Vec*. A potential reason is that FoodOn contains complex axioms that, alike OWL2Vec*, Onto2Graph can represent through its reasoning step (see Appendix A).

5. Conclusion

Graph representations of ontologies enable the use of graph-based machine learning methods to predict axioms. Machine learning methods on graphs have been extensively studied, and we analyzed the properties of different methods that project ontologies onto graphs and their effects on axiom inference using machine learning. We find that the properties of graph projections can have a significant effect on the inference of axioms using ontology embeddings. Our analysis can be used to further improve graph-based ontology embeddings and their applications.

References

[1] M. Kulmanov, F. Z. Smaili, X. Gao, R. Hoehndorf, Semantic similarity and machine learning with ontologies, Briefings in Bioinformatics 22 (2020). doi:`10.1093/bib/bbaa199`, bbaa199.

[2] M. Á. Rodríguez-García, R. Hoehndorf, Inferring ontology graph structures using OWL reasoning, BMC Bioinformatics 19 (2018). doi:`10.1186/s12859-017-1999-8`.

[3] J. Chen, A. Althagafi, R. Hoehndorf, Predicting candidate genes from phenotypes, functions and anatomical site of expression, Bioinformatics 37 (2020) 853–860. doi:`10.1093/bioinformatics/btaa879`.

[4] J. Chen, P. Hu, E. Jimenez-Ruiz, O. M. Holter, D. Antonyrajah, I. Horrocks, OWL2Vec*: embedding of OWL ontologies, Machine Learning (2021). doi:`10.1007/s10994-021-05997-6`.

[5] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2724–2743. doi:`10.1109/tkde.2017.2754499`.

[6] J. Chen, E. Jiménez-Ruiz, I. Horrocks, D. Antonyrajah, A. Hadian, J. Lee, Augmenting ontology alignment by semantic embedding and distant supervision, in: Extended Semantic Web Conference, 2021.

[7] F. Dau, Concept graphs and predicate logic, in: H. S. Delugach, G. Stumme (Eds.), Conceptual Structures: Broadening the Base, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, p. 72–86.

[8] M. Grohe, Logic, graphs, and algorithms, Electron. Colloquium Comput. Complex. TR07 (2007).

[9] J. F. Sowa, Peirce's tutorial on existential graphs, Semiotica 2011 (2011). doi:`10.1515/semi.2011.060`.

[10] W. A. Woods, WHAT'S IN A LINK: Foundations for Semantic Networks, Morgan Kaufmann, San Diego, 1975, p. 35–82. doi:`10.1016/B978-0-12-108550-6.50007-0`.

[11] J. F. Allen, A. M. Frisch, What's in a semantic network?, in: 20th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Toronto, Ontario, Canada, 1982, p. 19–27. doi:`10.3115/981251.981256`.

[12] J. F. Sowa, From existential graphs to conceptual graphs, International Journal of Conceptual Structures and Smart Applications 1 (2013) 39–72. doi:`10.4018/ijcssa.2013010103`.

[13] F. Dau, Mathematical logic with diagrams based on the existential graphs of peirce, 2005.

[14] F. Dau, P. Eklund, A diagrammatic reasoning system for $\mathcal{ALC}$, in: Z. Zhang, J. Siekmann (Eds.), Knowledge Science, Engineering and Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, p. 39–51.

[15] D. R. Corbett, Graph-Based Representation and Reasoning for Ontologies, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, p. 351–379. doi:`10.1007/978-3-540-78293-3_8`.

[16] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[17] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible sroiq, in: International Confer-

ence on Principles of Knowledge Representation and Reasoning, 2006.

[18] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, Owl 2: The next step for owl, Journal of Web Semantics 6 (2008) 309–322. doi:10.1016/j.websem.2008.05.001.

[19] J. Carroll, I. Herman, P. Patel-Schneider, Owl 2 web ontology language rdf-based semantics (second edition), 2012.

[20] R. Hoehndorf, A. Oellrich, M. Dumontier, J. Kelso, D. Rebholz-Schuhmann, H. Herre, Relations as patterns: bridging the gap between OBO and OWL, BMC Bioinformatics 11 (2010). URL: https://doi.org/10.1186/1471-2105-11-441. doi:10.1186/1471-2105-11-441.

[21] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, S. Lewis, The obo foundry: coordinated evolution of ontologies to support biomedical data integration, Nature biotechnology 25 (2007) 1251. doi:10.1038/nbt1346.

[22] I. Horrocks, Obo flat file format syntax and semantics and mapping to owl web ontology language, 2007.

[23] C. Golbreich, I. Horrocks, The obo to owl mapping, go to owl 1.1!, in: OWL: Experiences and Directions, 2007.

[24] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. L. Rector, C. Rosse, Genome Biology 6 (2005) R46. doi:10.1186/gb-2005-6-5-r46.

[25] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, Z. Duan, Knowledge graph completion: A review, IEEE Access 8 (2020) 192435–192456. doi:10.1109/ACCESS.2020.3030076.

[26] H. Ren, J. Leskovec, Beta embeddings for multi-hop logical reasoning in knowledge graphs, ArXiv abs/2010.11465 (2020).

[27] D. Yang, P. Qing, Y. Li, H. Lu, X. Lin, Gammae: Gamma embeddings for logical queries on knowledge graphs, in: Conference on Empirical Methods in Natural Language Processing, 2022.

[28] Z. Tang, S. Pei, X. Peng, F. Zhuang, X. Zhang, R. Hoehndorf, Tar: Neural logical reasoning across tbox and abox, 2022. doi:10.48550/ARXIV.2205.14591.

[29] Z. Tang, T. Hinnerichs, X. Peng, X. Zhang, R. Hoehndorf, Falcon: Faithful neural semantic entailment over alc ontologies, 2022. doi:10.48550/ARXIV.2208.07628.

[30] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, G. Sherlock, Gene ontology: tool for the unification of biology, Nature Genetics 25 (2000) 25–29. doi:10.1038/75556.

[31] D. M. Dooley, E. J. Griffiths, G. S. Gosal, P. L. Buttigieg, R. Hoehndorf, M. C. Lange, L. M. Schriml, F. S. L. Brinkman, W. W. L. Hsiao, FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration, npj Science of Food 2 (2018). doi:10.1038/s41538-018-0032-6.

[32] Y. Kazakov, M. Krötzsch, F. Simančík, The incredible elk, Journal of Automated Reasoning 53 (2013) 1–61. doi:10.1007/s10817-013-9296-3.

[33] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems,

volume 26, Curran Associates, Inc., 2013.

[34] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: AAAI Conference on Artificial Intelligence, 2015.

[35] J. Chen, Y. He, Y. Geng, E. Jimenez-Ruiz, H. Dong, I. Horrocks, Contextual semantic embeddings for ontology subsumption prediction, 2022. doi:`10.48550/ARXIV.2202.09791`.

[36] F. Zhapa-Camacho, M. Kulmanov, R. Hoehndorf, mOWL: Python library for machine learning with biomedical ontologies, Bioinformatics (2022). doi:`10.1093/bioinformatics/btac811`, btac811.

A. Analysis of the performance of projection methods

As shown in Section 4.3, in most cases OWL2Vec* obtains higher values on Hits@1 but Onto2Graph obtains lower mean rank. This phenomenon is due to the different capabilities of each method. We show the following example in GO using the class GO_2000859, which is involved in the following axiom in the training set:

$$\text{GO\_2000859} \equiv \text{GO\_0065007} \sqcap \exists \text{RO\_0002212.GO\_0035932} \tag{3}$$

And is involved in the following axiom in the testing set:

$$\text{GO\_2000859} \sqsubseteq \text{GO\_0023051} \tag{4}$$

OWL2Vec* generates an edge (GO_2000859, subclassof, GO_0065007) from axiom 3 but Onto2Graph does not. Furthermore, classes GO_0065007 and GO_0023051 are involved in training axioms, and the edge generated by OWL2Vec* is important in the prediction of the testing axiom 4 and contributing to the high value at Hits@k.

Furthermore, RDF performs worse than other methods because of the noise introduced by blank nodes. For example, axiom 3, will be projected as: (GO_2000859, subclassof, m), (m, intersection, l), (l, first, GO_0065007).

In the case of FoodOn, we notice that Onto2Graph obtains much lower mean rank than OWL2Vec*. This happens because Onto2Graph, through its reasoning step, generates edges than OWL2Vec* cannot generate due to the complexity of the axioms in FoodOn. For example, from the following axiom involving the entity CDNO_0200195:

$$\text{CDNO\_0200195} \equiv \text{PATO\_0000033} \sqcap \exists \text{RO\_0000052.} (\text{CHEBI\_12777} \sqcap \exists \text{BFO\_0000050.BFO\_0000040})$$

Onto2Graph generates

$$(\text{CDNO\_0200195}, \text{RO\_0000052}, \text{CHEBI\_12777})$$

whereas OWL2Vec* does not generate any edge with roles as edge labels. This difference between both graphs enables Onto2Graph to get lower mean rank when predicting axioms $C \sqsubseteq \exists R.D$.

B. Description logics and ontologies

Ontologies can be constructed using Description Logics. A Description Logic (DL) [16] theory is defined over a signature $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ where $\mathbf{C}$ is a set of class names, $\mathbf{R}$ a set of role names, $\mathbf{I}$ a set of individual names. There are several description logic languages that differ from each other on the operators that they support. In the description logic $\mathcal{ALC}$, a concept description is constructed inductively from class names using the operations of negation ($\neg$), intersection ($\sqcap$), union ($\sqcup$), existential ($\exists$) and universal quantification ($\forall$).

In DLs, subsumption axioms between concept descriptions can be defined using the subsumption operation ($\sqsubseteq$). To define the semantics of a DL, we need an interpretation domain $\Delta^{\mathcal{I}}$

and an interpretation function $\cdot^{\mathcal{I}}$. In $\mathcal{ALC}$, for a class name $A \in \mathbf{C}$, its interpretation is the set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. The semantics of concept descriptions is constructed inductively:

$$
\begin{aligned}
\bot^{\mathcal{I}} &= \emptyset \\
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \left\{ a \in \Delta^{\mathcal{I}} \mid \forall b.(a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}} \right\} \\
(\exists R.\top)^{\mathcal{I}} &= \left\{ a \in \Delta^{\mathcal{I}} \mid \exists b.(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \right\}.
\end{aligned}
\tag{5}
$$

C. Details on implementations of projection methods

For Onto2Graph, we relied on the original implementation of [2] found at https://github.com/bio-ontology-research-group/Onto2Graph. In the case of RDF, we used the Python library `rdflib`, found at https://github.com/RDFLib/rdflib

In the case of the projection found in OWL2Vec*, we used the implementation found in mOWL[36]. Both mOWL and the original implementation of OWL2Vec* projection, project axioms with complex superclasses such as $C \sqsubseteq D \sqcap E$. We added this rule in Table 3.

Table 3
Projection rules for OWL2Vec* model.

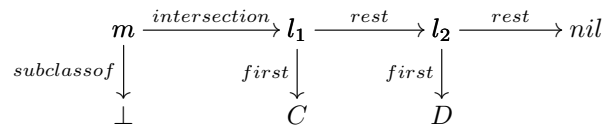| Axiom or triple(s) of condition 1	Axiom or triple(s) of condition 2	Projected triple(s)
$A \sqsubseteq \Box r.D$ or $\Box r.D \sqsubseteq A$	$D \equiv B\|B_1 \sqcup \ldots \sqcup B_n\|B_1 \sqcap \ldots \sqcap B_n$	$\langle A, r, B \rangle$ for $i \in 1, \ldots, n$
$\exists r.\top \sqsubseteq A(domain)$	$\top \sqsubseteq \forall r.B$ (range)	$\langle A, r, B_i \rangle$
$A \sqsubseteq \exists r.b$	$B(b)$	
$r \sqsubseteq r'$	$\langle A, r', B \rangle$ has been projected	
$r' \sqsubseteq r^{-1}$	$\langle B, r', A \rangle$ has been projected	
$s_1 \circ \ldots \circ s_n \sqsubseteq r$	$\langle A, s_1, C_1 \rangle \ldots \langle C_n, s_n, C_B \rangle$ has been projected	
$B \sqsubseteq A$	–	$\langle B, \text{subClassOf}, A \rangle$
		$\langle A, \text{subClassOf}^{-1}, B \rangle$
$A(a)$	–	$\langle a, \text{type}, A \rangle$
		$\langle A, \text{type}^{-1}, a \rangle$
$r(a, b)$	–	$\langle a, r, b \rangle$
New rule		
$A \sqsubseteq B \sqcap \exists R.C$		$\langle A, \text{subClassOf}, B \rangle$

D. Examples of graphs generated by different projection methods

To show the differences between the different projection results, Figure 3 shows the subgraphs generated by each method on the axiom $C \sqcap D \sqsubseteq \bot$. Similarly, Figure 4 shows the projections of axioms $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$ using the RDF projection.
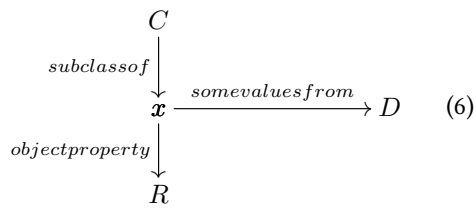
$$C \xrightarrow{\sqcap D} \bot$$
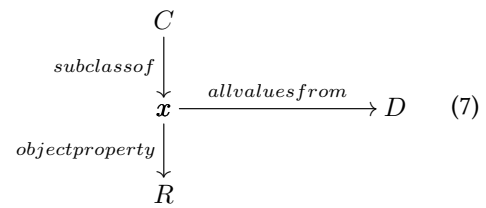
$$D \xrightarrow{\sqcap C} \bot$$

(a) Onto2Graph

$$m \xrightarrow{intersection} l_1 \xrightarrow{rest} l_2 \xrightarrow{rest} nil$$

with $subclassof$ from m to $\bot$, $first$ from l_1 to C, $first$ from l_2 to D.

(b) RDF

Figure 3: Graphs generated by each projection method on the axiom $C \sqcap D \sqsubseteq \bot$. This axiom is applicable for Onto2Graph (if the appropriate pattern is defined) and for RDF projection.

C with $subclassof$ to x, $x \xrightarrow{somevaluesfrom} D$, $objectproperty$ from x to R. (6)

C with $subclassof$ to x, $x \xrightarrow{allvaluesfrom} D$, $objectproperty$ from x to R. (7)

(a) $C \sqsubseteq \exists R.D$ (b) $C \sqsubseteq \forall R.D$

Figure 4: RDF projection of axioms $C \sqsubseteq \exists R.D$ and $C \sqsubseteq \forall R.D$

E. Graphs generated for GO and FOODON

For the quantitative analysis of projection methods, we chose two ontologies: GO and FoodOn. Table 4 shows the number of edges generated by each projection method on the different ontologies.

Table 4
Number of edges generated by projection methods. $\sqsubseteq$ represents the edges for subclass axioms between named classes (Onto2Graph, OWL2Vec*, RDF) or between a named class and a blank node (RDF). $\sqsubseteq^{-1}$ represents the inverse edge of $\sqsubseteq$.

Projection method	Edges generated on each ontology											
	GO			*GO_{sub}*			*GO_{ex}*			*FoodOn*		
	Total	$\sqsubseteq$	$\sqsubseteq^{-1}$	Total	$\sqsubseteq$	$\sqsubseteq^{-1}$	Total	$\sqsubseteq$	$\sqsubseteq^{-1}$	Total	$\sqsubseteq$	$\sqsubseteq^{-1}$
Onto2Graph	99619	69353(70%)	–	97660	66886(68%)	–	96294	69353(72%)	–	77203	36755(47%)	–
OWL2Vec*	189849	79210(42%)	79210(42%)	176281	72426(41%)	72426(41%)	188355	79210(42%)	79210(42%)	89980	40248(44%)	40248(44%)
RDF	709409	87602(12%)	–	702477	80667(11%)	–	702128	85781(12%)	–	210718	46668(22%)	–

Table 5
Number of edges projected from the FoodOn ontology from axioms $C \sqsubseteq \exists R.D$ for methods Onto2Graph and OWL2Vec*. "With self-loops" represent edges of the from (C, R, C) for some ontology role R. "Shared" represent the number of edges found in both graphs.

Projection method	Number of existential axioms $C \sqsubseteq \exists R.D$			
	Total	With self-loops	Shared	Not-shared
Onto2Graph	40448	32018	6583	1847
OWL2Vec*	9484	4	6583	2899

F. Knowledge graph embedding method TransE

In TransE[33], every edge (h, r, t) is assigned a distance score

$$d_{(h,r,t)} = ||h + r - t|| \tag{8}$$

That is, every edge label is considered as a translation betweeon the head and tail nodes. The training objective is denoted as $\mathcal{L}$:

$$\mathcal{L} = \max(0, d_{(h,r,t)} - d_{(h,r,t')} + \gamma) \tag{9}$$

where (h, r, t) is a *positive triple* that exists in the graph and (h, r, t') is a *negative triple* that does not exist in the graph and is computed by corrupting the node t by another node that is chosen randomly. $\mathcal{L}$ tries to minimize distance of positive triples with respect to negative ones. γ is a margin parameter that enforces a minimum separation between the scores of a positive and a negative sample.

G. Hyperparameter optimization

For all the methods, we performed hyperparameter optimization of the following parameters: embedding size $[64, 128, 256]$, margin (γ) $[0.0, 0.2, 0.4]$, L2 regularization factor $[0.0, 1e^{-4}, 5e^{-4}]$, batch size $[4096, 8192, 16384]$ and learning rate $[0.1, 0.01, 0.001]$.

Table 6
Hyperparameters chosen for projection methods in experiments regarding Table 1

Method	Chosen hyperparameter values				
	Dimension	Margin	L2 Reg.	Batch size	Learning rate
Prediction of $C \sqsubseteq D$ with GO and GO$_{sub}$					
Onto2Graph-TransE	128	0.4	0.0005	4096	0.010
Onto2Graph-TransR	128	0.4	0.0001	8192	0.001
OWL2Vec*-TransE	128	0.2	0.0001	8192	0.010
OWL2Vec*-TransR	128	0.4	0.0001	8192	0.001
RDF-TransE	64	0.2	0.0	4096	0.010
RDF-TransR	128	0.4	0.0	8192	0.001
Prediction of $C \sqsubseteq \exists R.D$ with GO and GO$_{ex}$					
Onto2Graph-TransE	64	0.4	0.0	8192	0.100
Onto2Graph-TransR	256	0.4	0.0001	4096	0.001
OWL2Vec*-TransE	128	0.4	0.0	16384	0.001
OWL2Vec*-TransR	64	0.0	0.0	4096	0.001
RDF-TransE	64	0.4	0.0	8192	0.010
RDF-TransR	64	0.4	0.0005	16384	0.001

Table 7
Hyperparameters chosen for projection methods in experiments regarding Table 2

Method	Chosen hyperparameter values				
	Dimension	Margin	L2 Reg.	Batch size	Learning rate
Case A and Case B					
Onto2Graph-TransE	64.0	0.0	0.0	16384.0	0.010
Onto2Graph-TransR	256	0.2	0.0005	16384	0.001
OWL2Vec*-TransE	256.0	0.4	0.0001	16384.0	0.001
OWL2Vec*-TransR	64	0.2	0.0001	16384	0.001
RDF-TransE	128.0	0.0	0.0	4096.0	0.001
RDF-TransR	128.0	0.0	0.0	4096.0	0.001