Machine learning with biomedical ontologies

Maxat Kulmanov[1], Fatima Zohra Smaili[1], Xin Gao[1] and Robert Hoehndorf[1,*]

[1]Computational Bioscience Research Center, Computer, Electrical and Mathematical Sciences & Engineering Division, King Abdullah University of Science and Technology, 4700 King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia

Ontologies have long been employed in the life sciences to formally represent and reason over domain knowledge, and they are employed in almost every major biological database. Recently, ontologies are increasingly being used to provide background knowledge in similarity-based analysis and machine learning models. The methods employed to combine ontologies and machine learning are still novel and actively being developed. We provide an overview over the methods that use ontologies to compute similarity and incorporate them in machine learning methods; in particular, we outline how semantic similarity measures and ontology embeddings can exploit the background knowledge in biomedical ontologies, and how ontologies can provide constraints that improve machine learning models. The methods and experiments we describe are available as a set of executable notebooks, and we also provide a set of slides and additional resources at `https://github.com/bio-ontology-research-group/machine-learning-with-ontologies`.

Key points

- Ontologies provide background knowledge that can be exploited in machine learning models.
- Ontology embeddings are structure-preserving maps from ontologies into vector spaces and provide an important method for utilizing ontologies in machine learning. Embeddings can preserve different structures in ontologies, including their graph structures, syntactic regularities, or their model-theoretic semantics.
- Axioms in ontologies, in particular those involving negation, can be used as constraints in optimization and machine learning to reduce the search space.

Key words

machine learning, semantic similarity, ontology, knowledge representation, neuro-symbolic integration

Author descriptions

Maxat Kulmanov is a postdoctoral researcher in computer science. His research interests include knowledge discovery and data integration using artificial intelligence and Semantic Web technologies in biology and biomedicine.

[*]To whom correspondence should be addressed. Email: robert.hoehndorf@kaust.edu.sa. Tel.: +966-12-8081643.

Fatima Zohra Smaili is a doctoral student in computer science at King Abdullah University of Science and Technology. Her research focuses on ontology-based knowledge representation.

Xin Gao is an Associate Professor in computer science, Acting Associate Director of the Computational Bioscience Research Center, and Lead of the Structural and Functional Bioinformatics Group at King Abdullah University of Science and Technology. His research focuses on bioinformatics and machine learning.

Robert Hoehndorf is an Assistant Professor in computer science and principal investigator of the Bio-Ontology Research Group at King Abdullah University of Science and Technology. His research focuses on combining knowledge representation and machine learning in biology.

1 Introduction

Machine learning methods are now applied widely across life sciences to develop predictive models [1]. These models commonly solve an optimization problem, i.e., they perform search for an optimal solution to a function in a continuous or discrete space. Domain-specific knowledge can be used to constrain search and find optimal or near-optimal solutions faster, or to find better solutions; this observation has led Feigenbaum in 1977 to suggest that the power of Artificial Intelligence systems lies in the domain-specific knowledge they encode and are able to exploit, leading to the paradigm that "in the knowledge lies the power" [2].

In the life sciences, domain-specific knowledge is often encoded in ontologies and in the data- and knowledge-bases that use ontologies for annotation. Hundreds of ontologies have been developed, spanning almost all domains of biological and biomedical research. The main features biomedical ontologies provide are controlled vocabularies for characterizing biological phenomena, and as formalized knowledge bases that formally describe the phenomena within a domain and link them to other related domains. For example, phenotype ontologies are used for characterizing the phenotypes observed in a variety of model organism databases [3–6] as well as in human genetics [7, 8], and these ontologies provide a controlled set of classes, their labels, and definitions for the purpose of annotating the phenotypes observed in conditions recorded in databases. Moreover, phenotype ontologies are also interlinked with other ontologies through the use of formal axioms and can be used to relate the phenotype observations to biological functions, anatomical locations, developmental stages, or chemical substances [9, 10]. The majority of biomedical ontologies are formalized using the Web Ontology Language (OWL) [11], a language based on Description Logic (a decidable fragment of first order predicate logic). OWL comes with an explicit semantics that defines how statements made in OWL constrain the world in which these statements are interpreted – the "models" in which these statements are true.

The background knowledge contained in ontologies can be used in machine learning models for at least two different purposes: to expand or enrich features to be used, and to constraint the search for an optimal solution to a learning problem. Expanding or enriching features may make information available to a machine learning model that it would not be able to access without relying on ontologies. For example, linking phenotypes such as *cardiomyopathy* to the anatomical structures that are affected (i.e., the *heart*) can create novel and direct associations with other datasets that do not otherwise exist. In the example of *cardiomyopathy*, the link to *heart* as the anatomical structure can be used to relate the phenotype to gene expression in *heart* tissue or in cardiomyocytes; this constrain is given *a priori* through the axioms in phenotype, anatomy, and celltype ontology, and does not need to be discovered.

A second application of the knowledge in ontologies is to constrain the search for solutions to an optimization problem, and thereby finding a solution faster, finding a better solution, or finding a solution that is generalized better. One example of such a constraint is the true path rule that was originally proposed in the Gene Ontology [12], which states that if a gene product G has the potential to be involved in a process P_1, and every process P_1 is a part of another process P_2, then G must also be involved in P_2. This constraint is 'hard' in that it is not an empirical law or observation, but should hold in virtue of the definition of P_1 and P_2 – it is *impossible* for G to participate in P_1 but not P_2. For example, a gene product involved in *developmental cell growth* (`GO:0048588`) must be involved in *cell development* (`GO:0048468`) simply based on the definition of the two classes in the Gene Ontology.

It is now a challenge to identify general ways in which ontologies, and their underlying formalisms based on first order logic, can be combined with the modern machine learning models that are becoming so widespread. This challenge is not only one of research in Artificial Intelligence but exists throughout the life sciences due to the widespread use of

Type	Method/Tool	Description	URL
Processing and preprocessing ontologies	OWLAPI	Reference library to process OWL ontologies, supports most OWL reasoners [13]	`https://github.com/owlcs/owlapi`
	funowl	Python library to process OWL ontologies	`https://github.com/hsolbrig/funowl`
	owlready2	Python library to process OWL ontologies	`https://pypi.org/project/Owlready2/`
	Apache Jena	RDF library with OWL support	`https://jena.apache.org/`
	rdflib	Python RDF library with OWL support	`https://github.com/RDFLib/rdflib`
	Protégé	Ontology editor and knowledge engineering environment [14]	`https://protege.stanford.edu/`
Computing entailments, reasoning	ELK	Very fast reasoner for the OWL 2 EL profile with polynomial worst-case time complexity [15]	`https://github.com/liveontologies/elk-reasoner`
	HermiT	Automated reasoner supporting most of OWL axioms with exponential worst-case complexity [16]	`http://www.hermit-reasoner.com/`
	Pellet	OWL reasoner supporting most of the OWL constructs and supporting several additional features [17]	`https://github.com/stardog-union/pellet`
Generating graphs from ontologies	OBOGraphs	Syntactic conversion of ontologies to graphs, targeted at OBO ontologies	`https://github.com/geneontology/obographs`
	Onto2Graph	Semantic conversion of OWL ontologies to graphs, following the axiom patterns of the OBO Relation Ontology [18]	`https://github.com/bio-ontology-research-group/Onto2Graph`
Computing Semantic Similarity	Semantic Measures Library	Comprehensive Java library to compute semantic similarity measures over ontologies [19]	`http://www.semantic-measures-library.org/sml/`
	sematch	Python library to compute semantic similarity on knowledge graphs [20]	`https://github.com/gsi-upm/sematch`
	DiShIn	Python library for semantic similarity on ontologies [21]	`https://github.com/lasigeBioTM/DiShIn`
Embedding graphs	OWL2Vec	Method that combines generation of graphs from ontologies, random walks on the generated graphs, and generation of embeddings using Word2Vec. Syntactically processes most OWL axioms [22]	`https://github.com/oholter/matcher-with-word-embedings`
	DL2Vec	Method that combines generation of graphs from ontologies, random walks on the generated graphs, and generation of embeddings using Word2Vec. Syntactically processes most OWL axioms [23]	`https://github.com/bio-ontology-research-group/DL2Vec`
	Walking RDF&OWL	Method that combines generation of graphs from ontologies, random walks on the generated graphs, and generation of embeddings using Word2Vec. Only considers the ontology taxonomy. [24]	`https://github.com/bio-ontology-research-group/walking-rdf-and-owl`
	RDF2Vec	Method to embed RDF graphs [25]	`https://github.com/IBCNServices/pyRDF2Vec`, `https://github.com/dwslab/jRDF2Vec`
	Node2Vec	Method to embed graphs using biased random walks [26]	`http://snap.stanford.edu/node2vec/`
	PyKEEN, BioKEEN	Toolkit for generating knowledge graph embeddings using several different approaches [27, 28]	`https://github.com/SmartDataAnalytics/PyKEEN`
	OpenKE	Library and toolkit for generating knowledge graph embeddings	`https://github.com/thunlp/OpenKE`
	PyTorch Geometric	Library for graph neural networks which can be used to generate graph embeddings [29]	`https://github.com/rusty1s/pytorch_geometric`
Embedding axioms	Onto2Vec	Embeddings based on treating logical axioms as a text corpus [30]	`https://github.com/bio-ontology-research-group/onto2vec`
	OPA2Vec	Embeddings that combine logical axioms with annotation properties and the literature [31]	`https://github.com/bio-ontology-research-group/opa2vec`
	EL Embeddings	Embeddings that approximate the interpretation function and preserve semantics for intersection, existential quantifiers, and bottom [32]	`https://github.com/bio-ontology-research-group/el-embeddings`
Ontology-based constrained learning	DeepGO	Implements an ontology-based hierarchical classifier for function prediction. The hierarchical classification module is generic and can be used with other ontologies and applications [33]	`https://github.com/bio-ontology-research-group/deepgo`
	DEEPred	Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks [34]	`https://github.com/cansyl/DEEPred`
	DeepMiR2GO	Inferring Functions of Human MicroRNAs Using a Deep Multi-Label Classification Model [35]	`https://github.com/JChander/DeepMiR2GO`

Table 1: An overview of software tools and applications involved in machine learning with biomedical ontologies.

ontologies and formalized knowledge bases in biology and biomedicine.

Here, we describe and review the state-of-the-art and recent advances in machine learning with biomedical ontologies. We use as a starting point in our review more traditional semanic similarity measures applied to ontologies; semantic similarity measures are a method from Artificial Intelligence that can determine the similarity between two or more entities using formalized background knowledge. We continue to introduce unsupervised, deep learning methods on ontologies that generate 'embeddings' for entities in ontologies, and we show that these embeddings can be used like semantic similarity measures while additionally allowing to overcome some of their limitations. Third, we highlight methods that use ontologies as constraints in optimization problems. We summarize the methods and tools we introduce in Table 1. We continue by introducing a novel benchmark dataset for machine learning with ontologies and demonstrating the methods we discuss on this dataset; we also make all experiments available as executable notebooks which can be adopted to other use cases. We finish by reviewing some of the main limitations and future research directions for the combination of ontologies and machine learning.

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \vert \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
generalized concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$r_1 \circ ... \circ r_n \sqsubseteq r$	$r_1^{\mathcal{I}} \circ ... \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

Table 2: Description Logic EL

2 Axioms, graphs, and knowledge graphs

An ontology is an "explicit specification of a conceptualization of a domain" [36], i.e., an ontology is an artifact used to formally specify the intended meaning of a vocabulary within a domain. Ontologies contain domain knowledge, encoded in the form of axioms, natural language labels, synonyms, definitions, and other types of annotation properties. The majority of ontologies in the life sciences are encoded using the Web Ontology Language (OWL) [11], a language that is a part of the Semantic Web stack [37] and based on Description Logics [38]. Description Logics enable a formal, machine-readable description of the types of entities within a domain and the relations in which they stand [38]. Table 2 illustrates how the semantics of such a formal language (in this case, the Description Logic EL) is specified; syntactic constructs are assigned an interpretation in a mathematical structure that resembles a world in which these constructs are true. For example, $C \sqsubseteq D$ will be true in those structures in which all entities that are in the interpretation of C are also in the interpretation of D.

The semantics of logical languages gives rise to entailment; a statement ϕ is logically entailed by a set of statements O if all the structures in which all statements in O are true also make ϕ true. For example, the two statements $\{C \sqsubseteq D, D \sqsubseteq E\}$ entail the statement $C \sqsubseteq E$. The process of computing entailments – deduction or logical inference – plays a crucial role in using ontologies because it allows to automatically derive statements that are not explicitly asserted in a knowledge base, and can also be used to detect whether a set of statements is contradictory.

Many analysis methods that rely on ontologies, including machine learning methods and semantic similarity measures, rely on generating some form of graph structures from the axioms in an ontology. There are several ways in which axioms can be used to generate a graph structure, and many can be formulated as computing entailments. An important ontology for generating graphs from biomedical ontologies is the OBO Relation Ontology [39] which provides a set of axiom patterns that must hold true for two classes if an edge between them should be created. An axiom pattern is an axiom with variables for classes or individuals; $X \sqsubseteq Y$ is an axiom pattern in which X and Y are variables and if this statement is true for two classes X and Y, an edge labels `is-a` should be created between them: $X \xrightarrow{\text{is-a}} Y$. More complex axiom patterns involve quantifiers, such as $X \sqsubseteq \exists \text{part-of}.Y$ which gives rise to the edge $X \xrightarrow{\text{part-of}} Y$. Axioms can also express disjointness between two classes such as $X \sqcap Y \sqsubseteq \bot$ based on which a `disjoint` edge can be created ($X \leftrightarrow \text{disjoint} Y$). To be generally applicable, these patterns must also be able to utilize entailments; for example, if $X \sqsubseteq \exists \text{part-of}.Y$ and $Y \sqsubseteq \exists \text{part-of}.Z$ are a part of an ontology, and the relation `part-of` is transitive ($\text{part-of} \circ \text{part-of} \subseteq \text{part-of}$), then $X \sqsubseteq \exists \text{part-of}.Z$ would be entailed and consequently a `part-of` edge between X and Z created ($X \xrightarrow{\text{part-of}} Y$). Depending on the algorithm that uses the graph, inferred edges can be added or not; for some relations, this amounts to adding their transitive closure,

Condition 1	**Condition 2**	**Edge**
$A \sqsubseteq QR_o.D$ or $QR_o.D \sqsubseteq A$	$D \equiv B \mid B_1 \sqcap \ldots \sqcap B_n \mid B_1 \sqcup \ldots \sqcup B_n$	$A \xrightarrow{R_o} B$ or $A \xrightarrow{R_o} B_i$ for $i \in 1 \ldots n$
$Domain(R_o) = A$	$Range(R_o) = B$	$A \xrightarrow{R_o} B$
$A \sqsubseteq \exists R_o.\{b\}$	$b : B$	$A \xrightarrow{R_o} B$
$R_o = R^-$	$A \xrightarrow{R} B$ in the graph	$B \xrightarrow{R_o} A$
$S_1 \circ \ldots \circ S_n \subseteq R_o$	$A \xrightarrow{S_1} C_1, \ldots, C_n \xrightarrow{S_n} B$ in the graph	$A \xrightarrow{R_o} B$
$B \sqsubseteq A$		$B \xrightarrow{is-a} A$, $A \xrightarrow{is-a^-} B$

Table 3: OWL2Vec rules for the projection of OWL axioms into an RDF graph. Q is any quantifier ($\exists$, $\forall$, $\leq n$, $\geq n$, $= n$). A, B, B_i and C_i are named classes, S_i, R_o, and R^- are object properties, b an individual name.

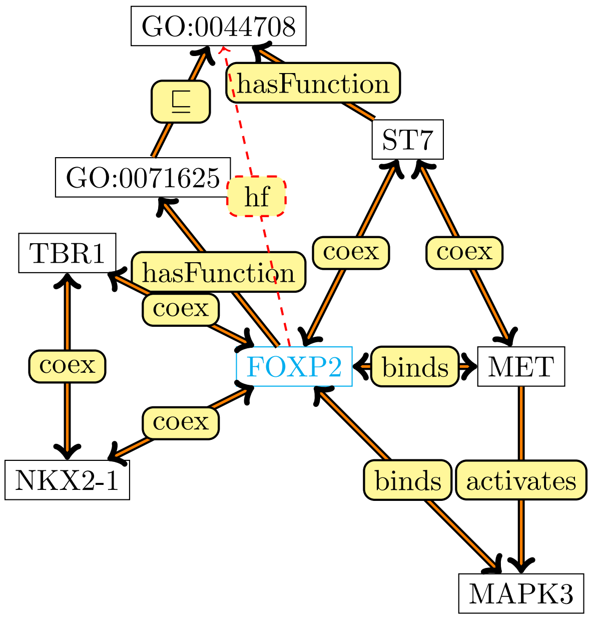

Figure 1: A knowledge graph centered around protein–protein interactions and functions of FOXP2.

or, alternatively, transitive reduction [18].

The types and complexity of axiom patterns giving rise to edges is an active research area. For example, OWL2Vec [22] uses the set of transformation rules shown in Table 3 to transform syntactic axiom patterns into edges.

The graphs generated from ontologies also interact with graph-based representations of data, in particular using the Resource Description Framework (RDF) [40]. Graphs in which nodes represent entities within a domain and edges represent the relations between the nodes are sometimes called *knowledge graphs*, and they correspond to a subset of the formalism underlying OWL in which only relations between individuals, and possibly certain axioms for relations, are considered. However, graph-based representations of the axioms in ontologies can also be considered knowledge graphs, in particular when both individuals and classes are included in the graph. For example, Figure 1 shows a graph in which interactions between proteins, the associations between proteins and their functions, and some axioms from the Gene Ontology are included. There are several ways in which such a graph could have been represented in OWL and then converted into such a graph representation using axiom patterns [41, 42]; for example, the edge between *MET* and *MAPK3* could arise from an axiom $MET \sqsubseteq \exists$ activates $.MAPK3$ and the edge between *FOXP2* and *GO:0071625* from the axiom $FOXP2 \sqsubseteq \exists$ hasFunction .GO:0071625. The dashed edge between *FOXP2* and *GO:0044708* is an edge that would be generated through entailment based on these axioms.

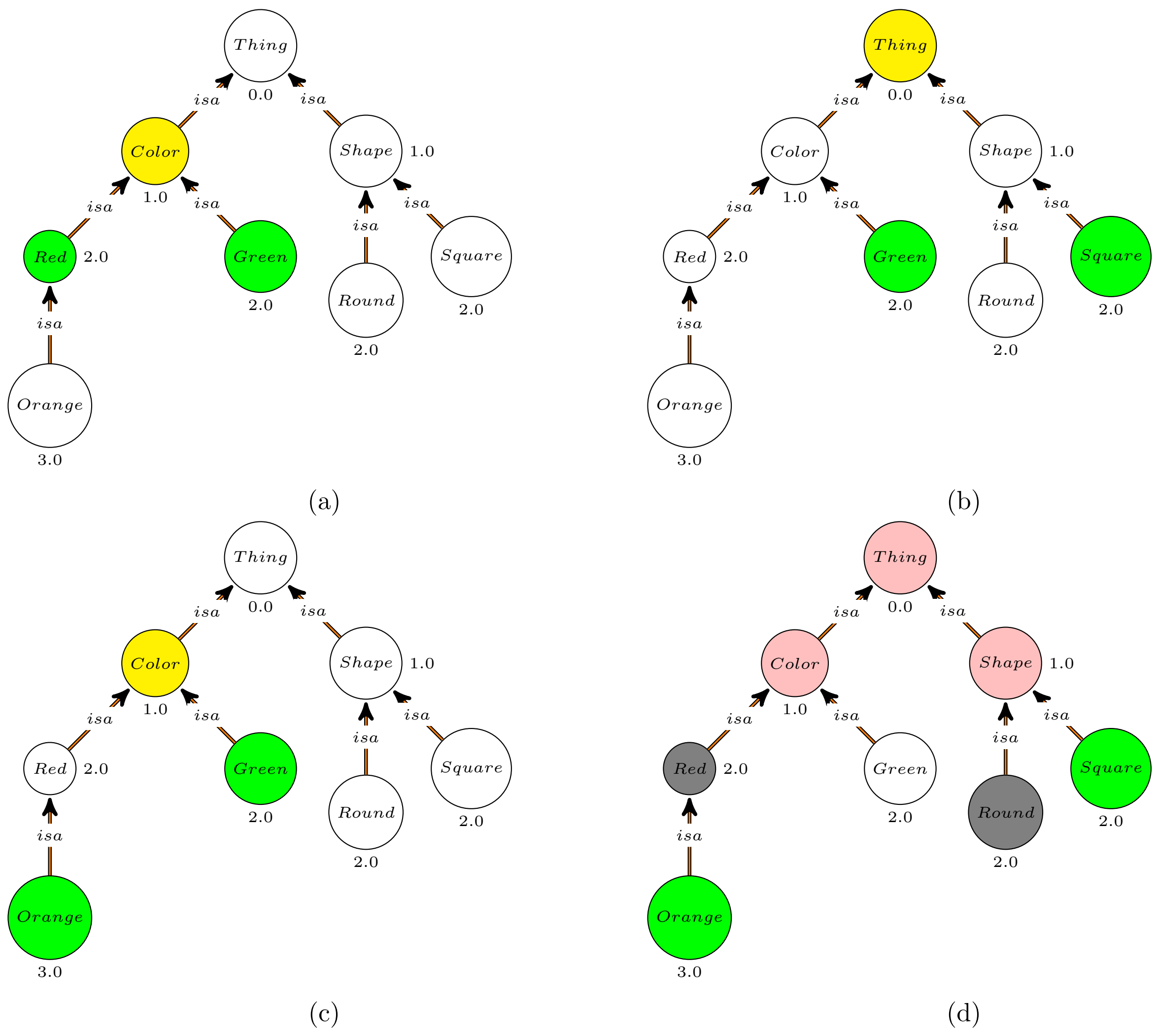

Figure 2: A fragment of the PATO ontology focusing on colors and shapes. Numbers near classes indicate specificity and information content of the classes.

3 Semantic similarity

Semantic similarity measures are widely used in biomedical domain. They are used to compare words, terms and ontology classes based on the background knowledge in the ontologies, annotations and large text corpora. Similarity measures are mostly hand-crafted and can be used as unsupervised classifiers for association prediction or as features in supervised learning models or a clustering algorithms. They can be applied to a variety of tasks such as predicting protein–protein interactions [43], gene–disease associations [44, 45], diagnosing patients [46, 47], determining sequence similarity [48], or evaluating computational methods which predict ontology class annotations [49].

We can compute semantic similarity measures between classes, class instances and annotated entities. A function $sim : D \times D$ is a similarity on a domain D if it is non-negative ($sim(x, y) \leq 0$), symmetric ($sim(x, y) = sim(y, x)$), and if self-similarity yields the highest similarity values within the domain ($sim(x, x) = \max_D$), or – as a weaker version – if self-similarity is higher than similarity to any other domain entity ($sim(x, x) > sim(x, y)$).

A simple similarity measure, sim_{Rada}, can be based on the shortest path between two nodes in the graph [50]. It can be defined as:

$$sim_{Rada}(x, y) = \frac{1}{dist_{SP}(x, y) + 1}$$

This similarity measure is useful when edges in a graph correspond mostly uniformly to some kind of semantic distance. However, when comparing ontology classes, edges represent axioms involving two classes which may not correspond to this assumption. For example, `is-a` edges order classes from general to more specific, such as in the ontology in Figure 2a. In this figure, $sim_{Rada}(Color, Shape)$ will have the same value as $sim_{Rada}(Red, Green)$ since these two classes have the same distance in the graph. However, in many applications *Red* and *Green* should be more similar than *Color* and *Shape* because they are both colors. In this case, distance based similarities might not be very intuitive and a measure of node specificity needs to be considered.

There are many ways to compute class specificity. For instance, we can consider specificity as a function of the depth, number of children, or the information content of a class. Formally, class specificity is a function $\sigma : C \mapsto \mathbb{R}$ which meets the condition that for all $x, y \in C$, if $x \sqsubseteq y$ then $\sigma(x) \geq \sigma(y)$ [51]. The specificity measure can be defined using only the classes within an ontology (such as measures that consider the number of super-classes a class has, or the distance of a class to the root), or using information such as the number of instances of a class, or the number of annotations of a class within a database.

One of the most widely used methods to determine class specificity is the Resnik measure [52], which defines the specificity of a class as its information content:

$$IC_{Resnik}(x) = -\log p(x)$$

where

$$p(x) = \frac{|I(x)|}{|I(\top)|}$$

and $I(x)$ is the set of instances of x (or the set of annotations of a class within a database).

A large number of semantic similarity measures have been developed [51]. Pairwise similarity measures compute the similarity value between two classes. Examples of pairwise similarities used in the biomedical field include Resnik's [52], Lin's [53], Jiang & Conrath's [54] and Schlicker's [44] similarity measures. Many of these measures are variations of the Resnik measure which defines the similarity between classes x and y as the information content of their *most informative common ancestor (MICA)*:

$$Sim_{Resnik}(x, y) = IC(MICA(x, y))$$

In the example in Figure 2a, $Sim_{Resnik}(Red, Green)$ is equal to 1.0 and $Sim_{Resnik}(Color, Shape)$ is equal to 0.0 although they have the same distance. The downside of this similarity measure is that it does not take into account the specificity of the compared classes and all classes under the same MICA will have the same similarity value. For instance, in Figure 2b $Sim_{Resnik}(Green, Square)$ is equal to 0.0 which is the same as $Sim_{Resnik}(Color, Shape)$ and in Figure 2c $Sim_{Resnik}(Red, Green)$ and $Sim_{Resnik}(Orange, Green)$ are both equal to 1.0. To solve this issue, Lin's measure [53] also considers information content of the compared classes:

$$Sim_{Lin}(x, y) = \frac{2 \cdot IC(MICA(x, y))}{IC(x) + IC(y)}$$

With this measure, $Sim_{Lin}(Red, Green)$ is equal to 0.5 whereas $Sim_{Lin}(Orange, Green)$ is equal to 0.4 which is more intuitive.

When comparing two instances of ontology classes, or two entities annotated with classes in an ontology, we usually need to compare sets of classes. For example, we would have to compute the similarity of the set of all Gene Ontology annotations of one protein with the set of all Gene Ontology annotations of a second protein. There are two ways of determining the similarity between two sets of classes A and B. First, we can compute the pairwise similarities between all pairs of classes (a, b) such that $a \in A$ and $b \in B$, and then combine similarity values according to some combination strategy (such as computing the average). Second, we can directly define a similarity measure between the two sets A and B using a set similarity measure. For instance, we can use the Jaccard index between the two sets:

$$Sim_{Jaccard}(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

To make this a semantic similarity, we would at least close each of the sets X and Y with respect to superclass axioms, i.e., if $C \sqsubseteq D$ and $C \in X$ then $D \in X$. Figure 2d depicts the propagation of ontology classes for computing the similarity between a square-and-orange thing and a round-and-red thing. Set similarity can also incorporate class specificity, such as the weighted Jaccard index in the SimGIC [55] measure.

Semantic similarity measures have a variety of applications and a large number of software packages have been developed to ease their use. One prominent example is the Semantic Measures Library [19] which is a comprehensive Java library that allows to compute hundreds of different semantic similarity measures.

A common problem of semantic similarity measures is that it is difficult to choose one for a particular application. Similarity measures behave differently depending on their applications. For example, predicting protein–protein interactions may result in different performance [55, 56] with similarity measures depending on the organisms. They are not immune to biases in data and different similarities may react to the biases differently [57]. Furthermore, they are hand-crafted measures that are not able to adapt to the underlying data or application.

4 Embedding ontologies

Another option to define similarity measures on ontologies is through the use of embeddings. An embedding is a structure-preserving map from one mathematical structure to another. We can use embeddings to project the elements of one structure into a second one. The idea behind using embeddings is that the second structure may enable different or additional operations which are not possible in the first structure. For example, if we take ontologies or graphs that are discrete entities and map them into a continuous space (or real-valued vector space), we can apply machine learning or continuous optimization algorithms which operate on continuous data; there are also 'natural' similarity measures between real-valued vectors such as the cosine similarity or other distance measures and metrics.

In the context of machine learning in most cases, we aim to embed ontologies within real-valued vector spaces. The key question when embedding ontologies is which structure (of the ontology) to preserve within $\mathbb{R}^n$ and under which operations in $\mathbb{R}^n$ this structure is preserved. We classify approaches of embedding ontologies in three main types based on what aspect of the ontologies is preserved in $\mathbb{R}^n$. First, there are *graph-based* approaches which treat ontologies as graphs similar to how ontologies are treated by many semantic similarity measures, and the embeddings preserve this graph structure within $\mathbb{R}^n$. Second, *syntactic* approaches treat axioms similar to "sentences" and preserve syntactic regularities (such as frequencies of co-occurrences) in $\mathbb{R}^n$. Third, we consider *model-theoretic* approaches which preserve model-theoretic properties within $\mathbb{R}^n$ as a part of the embedding.

4.1 Graph-based ontology embeddings

Graph-based embedding methods preserve a graph structure within $\mathbb{R}^n$. One form of graph embeddings is based on random walks. In these methods, graphs are generated from ontologies using the methods we described, then random walks are used to explore the neighborhood of each node in the graph, and finally the set of walks is used as the basis of the embeddings.

One of the first methods for learning graph embeddings through random walks was DeepWalk [58] which generates a 'corpus' of sentences (i.e., sequences of nodes in the graph) through random walks starting from each node in the graph, and then applies Word2Vec [59] on the resulting corpus to obtain embedding vectors; the embeddings generated by Word2Vec preserve co-occurrence relations within a context window. DeepWalk can also be extended to include labeled edges [22, 24, 25].

For example, for the graph in Figure 1, the random walks can generate sentences such as

- *FOXP2 cooex ST7 hasFunction GO:0044708 ...*
- *FOXP2 hasFunction GO:0071625 is-a GO:0044708 ...*

and Word2Vec will then embed each node and edge label while preserving co-occurrence relations within this corpus [60]. Node2Vec [26] is a modified model that does explore the original graph through 'biased' random walks and therefore can force walks to remain within a certain distance of the origin node or explore further away.

Random walks have long been used as a model that simulates diffusion of information within a network [61–63] and can be used to identify and score node importance. In graph embeddings, these walks explore node neighborhood and generate a 'linear' representation (i.e., sequences of symbols) in which nodes that are reached more often also occur more often (and co-occur more often with the original node). Word2Vec, as a model that embeds sequences of symbols while maintaining this co-occurrence, generates embeddings that maintain this syntactic structure within the walks, and therefore aspects of the graph structure as well. Furthermore, some of the semantics of the axioms in the ontology can be encoded as constraints on the random walks or encoded in the graph; for example, symmetry can be modeled as a bi-directional edge, disjointness as a 'barrier' preventing a walk's transition, etc. It is obvious that the graph that is generated from the ontology axioms, and the information it captures, is crucial for generating useful embeddings; this is also an active research area [22].

Translational embeddings methods are a family of representation learning methods on knowledge graphs which model relations in the knowledge graph as translation operations between graph node embeddings. The methods have been successfully applied for several tasks such as link prediction, knowledge-graph completion and others. The methods represent knowledge graphs as a set of edges (s, p, o) (triples) and define a translation operation which translates $f_\eta(s)$ to $f_\eta(o)$ depending on the relation p. Here, f_η is a graph embedding.

TransE [64] was the first translational embedding method. It uses a vector representation for relations that have the same dimensions as vectors representing nodes, and defines the translation operation as the addition of the relation vector to the node vector:

$$f_\eta(s) + f_\eta(p) \approx f_\eta(o)$$

and further defines a scoring function for an edge based on the translation operation:

$$f_{sc}(s, p, o) = \|f_\eta(s) + f_\eta(p) - f_\eta(o)\|$$

Then, it minimizes the following loss function to learn f_η:

$$\sum_{(s,p,o) \in KG} \sum_{(s',p,o') \in KG'} \left[\gamma + f_{sc}(s, p, o) - f_{sc}(s', p, o')\right]_+$$

where KG' is a set of negative or corrupted triples that are not in the graph, $[x]_+$ indicates the positive part of x, and γ is a hyperparameter. This model can only accurately represent one-to-one relations and it is not suitable for one-to-many and many-to-many relations; in graphs generated from ontologies, even when focusing only on the subclass hierarchy, there are many such relations. Furthermore, TransE does not support transitive, symmetric or reflexive relations which are all important for faithfully embedding ontologies.

Many TransE successors have been developed to overcome the original model's limitations. For example, TransH [65] extended TransE by moving the translation operation to a relation-specific hyperplane. TransH represents each relation by two embedding vectors, the norm vector of the hyperplane (denoted as a function w_η) and a translation vector (denoted as a function d_η). The scoring function is then defined as:

$$\begin{aligned} f_{sc}(s,p,o) =& \|(f_\eta(s) - w_\eta^\top(p) f_\eta(s) w_\eta(p)) + d_\eta(p) - \\ & (f_\eta(o) - w_\eta^\top(p) f_\eta(o) w_\eta(p))\| \end{aligned}$$

With an additional vector, TransH performs translation operation on an augmented hyperplane and can therefore model one-to-many and many-to-many relations better than TransE. There are also many other models with various advantages and disadvantages [66, 67].

Translational embeddings are able to explicitly capture the graph structure and preserve some interpretability through the use of vector operations; however, they cannot easily capture even simple axioms such as transitivity, symmetry, or reflexivity of relations. Furthermore, any graph-based method will focus on a small set of graph patterns and lose some information about the ontology axioms, while many ontological axioms such as disjointness and axioms involving combinations of different logical operators cannot be fully converted to a graph.

4.2 Syntactic approaches

Ontologies provide a structured representation of biological knowledge in the form of logical axioms, and not all the axioms in an ontology can be represented naturally in a graph; this limits the ability of these methods to utilize all information encoded in the ontology. Syntactic embeddings embed ontologies in $\mathbb{R}^n$ considering only the set of axioms without creating an intermediate graph-based representation.

Onto2Vec [30] is a method that generates embeddings for ontology classes and instances taking into account the logical axioms that define the semantics of ontology classes. Onto2Vec takes an ontology O as input, uses a reasoner to infer additional logical axioms, mainly subclass axioms between named classes; it then treats each asserted or inferred axiom as a sentence and embeds the set of axioms using the Word2Vec language model. This allows Onto2Vec to embed ontologies directly based on their axioms while considering all axiom types, no matter how complex they are.

OPA2Vec [31] extends Onto2Vec to not only include logical axioms but annotation properties as well. Annotation properties in biomedical ontologies provide labels, synonyms, definitions, and other types of information about classes and instances in ontologies. OPA2Vec combines the corpus generated from the asserted and inferred logical axioms in Onto2Vec with a corpus generated from all or selected annotation properties. For example, from the annotation assertion that an OWL class C has a label L (using the `rdfs:label` annotation property in the OWL annotation axiom), OPA2Vec generates the statement `C rdfs:label L`, using the complete identifier for C and `rdfs:label`, and expressing L as a string literal; for instance, the annotation assertion of the class *Nuclear periphery* (`GO:0034399`) and its label is expressed as the sentence `<http://purl.obolibrary.org/obo/GO_0034399> <http://www.w3.org/2000/01/rdf-schema#label> nuclear periphery`. The identifier for the class C occurs within the ontology axioms and obtains parts of its meaning through the axioms; to ensure that the natural language terms used in the annotation properties have their 'natural' meaning as used in biomedical texts, OPA2Vec uses transfer learning and pre-trains a Word2Vec language model on biomedical literature texts, and then tunes it to generate the embeddings from the axioms plus annotation properties.

4.3 Model-theoretic or semantic

None of the embedding methods discussed so far are ‘semantic’ in the sense that they use the semantics of the underlying logic (as, for example, shown in Table 2). Instead, the models are based on syntactic co-occurences or preserving certain graph properties. However, the main advantage of using languages with an explicit semantics is that it provides constrains on how symbols should be interpreted.

EL Embeddings [32] aim to embed ontologies by mapping the symbols in the ontology into one specific interpretation, i.e., the embedding is identical to, or approximates, the interpretation function $\mathcal{I}$ in Table 2. EL Embeedings find an embedding that maps the class, relation, and instance symbols $\Sigma(O)$ into $\mathbb{R}^n$, $f_e : \Sigma(O) \mapsto \mathbb{R}^n$ such that $f_e(\Sigma(O))$ is a model of O ($f_e(\Sigma(O)) \models O$). Such an embedding yields a faithful representation of logical operators and quantifiers.

Formally, EL Embeddings embed classes as n-balls in n-dimensional space and relations as n-dimensional vectors. The correspondence with the semantics of the axioms in the ontology is established by setting the domain of discourse to $\mathbb{R}^n$ and the following condition: for all classes $C \in \Sigma(T)$ and relations $r \in \Sigma(T)$ it defines $f_e(C) = C^{\mathcal{I}}$:

$$C^{\mathcal{I}} = \{x \in \mathbb{R}^n | \, \|f_e(C) - x\| < r_e(C)\}$$

where $r_e(C)$ is the radius of the n-ball that corresponds to C, and $f_e(r) = r^{\mathcal{I}}$:

$$r^{\mathcal{I}} = \{(x, y) | x + f_e(r) = y\}$$

The latter condition is similar to the TransE translation operation only applied to instances.

The embeddings are generated through optimization using a set of loss functions that correspond to different normal forms of the axioms in ontologies. Using these embeddings it is possible to approximate the intended semantics of the language within the embedding space. In particular, it can be shown that if the loss can be reduced to zero, the resulting embedding corresponds to a model of the ontology [32]. Similar approaches to EL Embeddings are also investigated for querying knowledge graphs using logic formulas [68].

4.4 Using embeddings as semantic similarity measures and for machine learning

Embeddings can generate distributed representations of the symbols in ontologies while preserving syntactic or semantic properties. These representations – vectors in $\mathbb{R}^n$ can be visualized using dimensionality reduction techniques such as principal component analysis or tSNE [69]. They can also be used to compute similarity using any kind of similarity or distance measure applicable to real-valued vectors, in particular the cosine similarity or the Euclidean distance. However, these similarity measures are still generic and do not adapt to particular applications.

One of the most useful applications of ontology embeddings is as a part of machine learning models in which either a single embedding is used as input or multiple embeddings are used as input. Single embeddings can be used in classification and related tasks while multiple embeddings are often used to predict relations between the entities which were embedded [67]. Such an application is similar to using a semantic similarity measure with the key difference that the actual function that computes the similarity can be ‘trained’ in a task-specific way using supervised learning. For example, if the similarity between two protein embeddings is supposed to be a measure of whether or not they interact, using a set of proteins that interact can be used to design a customized, task-specific function that predicts, given two embeddings as an input, whether the proteins they represent should interact. Many neural network architectures and other machine learning models can be used for this task, but architectures that are used for similarity learning, such as Siamese neural networks, seem to perform well in practice [23].

5 Ontologies as constraints

Ontologies embeddings are a useful technique to make information in ontologies available as background knowledge to define similarity measures or use as features in machine learning models. In these cases, ontologies are used as the *input* of a similarity function or a model. However, ontologies can also be used as an *output* of a machine learning model and the axioms in the ontology used to constrain the output of a function, such as in the case when determining if the predictions of a machine learning model are consistent with the axioms in the ontology.

Ontologies are used as structured output in many domains in which the primary task is to predict whether some entity has a relation with one or more ontology classes, such as predicting genotype–phenotype relations (using phenotype ontologies as output), predicting gene–disease or drug–disease associations (using disease ontologies as output), or predicting protein functions (using the Gene Ontology as output). At the very least, these tasks need to satisfy the hierarchical constraints imposed by the ontologies in the output space: if an entity e is predicted to be associated with a class C, and that class C is a subclass of D, then e must also be associated with D. Similar constraints arise from other axioms in the ontology.

In general, there are at least five different approaches to using hierarchical relationships as constraints in classification models: flat, local per node, local per parent, local per level, and global hierarchical classification [70]. Flat classification is when the hierarchical constraints are not used during the prediction or training and the classification is done only using individual classes, and the consistency with the hierarchical constraints is enforced by propagating scores along the hierarchy only after predictions are made. This approach employs the constraints imposed by the ontology independently from the training or prediction process. In a local per node setting, a binary classifier is built for every class and predictions are made starting from the most general classes first and then moving to more specific ones, and stopping the prediction process once classes are predicted as negative. In a local per parent and local per level setting, multi-class classifiers are used for children classes of a parent or classes at the same level, respectively. Similarly to local per node classifiers, the prediction is performed in a step-wise manner from the most general class to more specific ones, and terminated once predictions are negative. The main drawback of local classifiers is that all classification models are trained independently from each other, and during the prediction process errors will propagate from general classes to more specific ones. Global hierarchical classifiers include the hierarchical constraints during training of a machine learning model, either as soft constraints or hard constraints, and also during prediction so that the output labels are forced to be consistent with the ontology axioms. The advantages of these models are that they take the semantics into account during training and therefore potentially reduce the search space; and that they can exploit dependencies between classes during training and prediction; the disadvantage is often the increased complexity of these classifiers [70].

While hierarchical machine learning models are used across many different application domains, life science ontologies are standing out with their large size and complex set of axioms; it is no surprise that constrained optimization methods applicable to large ontologies have emerged from research in bioinformatics. In particular prediction of functions and phenotypes benefits from machine learning models that are constrained by ontologies, and the established evaluation measures for predictive performance of models in these domains are also ontology-based [71]. A large number of ontology-based prediction models have been developed in these domains, using hierarchical top-down phenotype prediction [72], using structured support vector machines for predicting functions [73] and phenotypes [74], and several methods that incorporate ontology-based constraints in artificial neural networks [35, 75, 76].

6 Use case and application

Ontologies are used in almost every major biological database. There are more than 800 ontologies in ontology repositories such as BioPortal [77] which are used to describe different biological and biomedical entities. Consequently, ontologies play a role in many different biomedical machine learning tasks such as genotype–phenotype association prediction [46, 47], protein function prediction [49], drug–target prediction [78, 79], protein–protein interaction prediction [21, 48, 56], gene–disease association prediction [80], and many others.

Here, we evaluate ontology embedding methods on the task of predicting interactions between proteins based on the hypothesis that functionally related proteins are more likely to interact. We demonstrate how different ontology embedding methods can be applied, and we provide Jupyter Notebooks for all our experiments at `https://github.com/bio-ontology-research-group/machine-learning-with-ontologies`.

Proteins do not function in isolation, and many biological processes and functions are regulated by multiple proteins and their interactions. Databases such as String [81] collect information about protein–protein interactions (PPIs) from different sources with experimental evidence as well as PPIs that are computationally inferred and automatically assigned, and the functions of proteins are described using the Gene Ontology [12].

Organism	Proteins	Total	Training	Validation	Testing
Yeast	6,157	119,051	76,193	19,048	23,810
Human	17,185	420,534	269,143	67,285	84,106

Table 4: The total number of proteins and number of unique interaction pairs in training, testing, and validation datasets

We created two PPI datasets, one for interactions occurring in humans and one for yeast, based on data from the String database [81]. We filtered out interactions with a confidence score less than 700 to retain only high confidence interactions. Table 4 provides the total number of proteins and interactions in each dataset. We split the two datasets consisting of interaction pairs into train and test sets, with a ratio of 80% and 20%, respectively, and we used 20% of the training set as a validation set. We used these two datasets as benchmark sets for evaluating ontology embedding and semantic similarity methods, and we made the datasets with documentation publicly available for download and provided the links in our public repository so that anybody can use the same data to benchmark and compare ontology-based prediction methods. The training and validation sets should be used to train and tune model parameters and select the best models, while the evaluation results and comparisons should be reported using the test set.

We predicted PPIs based on the associations of proteins with their functions and cellular locations represented in the GO, known interactions between proteins, and the information contained in the GO. One key question is how to represent these three types of knowledge as axioms in an ontology or knowledge base. We adopted a representation scheme in which all entities (proteins, functions, cellular locations) are classes and the relations between the entities are expressed as axioms [41, 42]. Specifically, if there is an interaction between proteins P_1 and P_2, we asserted the axioms $P_1 \sqsubseteq \exists \text{interacts-with}.P_2$ and $P_2 \sqsubseteq \exists \text{interacts-with}.P_1$; if protein P is associated with a GO class C, we asserted the axiom $P \sqsubseteq \exists \text{has-function}.C$. We combined this set of axioms with the GO (released on 22 February 2020) to form our knowledge base.

For graph-based embedding methods, we generated a graph by creating an edge for existential restrictions in subclass axioms: if $X \sqsubseteq \exists R.Y$ is an (asserted) axiom in the knowledge base (consisting of GO together with the axioms we added), we created nodes X and Y, and an edge from X to Y labeled R. For the Onto2Vec and OPA2Vec embedding methods, we only inputted GO together with the set of protein-to-GO associations.

Method	Raw Hits@10	Filtered Hits@10	Raw Hits@10	Filtered Hits@100	Raw Mean Rank	Filtered Mean Rank	Raw AUC	Filtered AUC
TransE	0.06	0.13	0.32	0.40	1125.4	1074.8	0.82	0.83
SimResnik	**0.09**	**0.17**	0.38	0.48	757.8	706.9	0.86	0.87
SimLin	0.08	0.15	0.33	0.41	875.4	824.5	0.84	0.85
EL Embeddings	0.08	**0.17**	**0.44**	**0.62**	**451.29**	**394.04**	**0.92**	**0.93**
Onto2Vec	0.08	0.15	0.35	0.48	641.1	587.9	0.79	0.80
OPA2Vec	0.06	0.13	0.39	0.58	523.3	466.6	0.87	0.88
Random walk	0.06	0.13	0.31	0.40	612.6	587.4	0.87	0.88
Node2Vec	0.07	0.15	0.36	0.46	589.1	522.4	0.87	0.88

Table 5: Prediction performance for yeast protein–protein interactions.

Method	Raw Hits@10	Filtered Hits@10	Raw Hits@10	Filtered Hits@100	Raw Mean Rank	Filtered Mean Rank	Raw AUC	Filtered AUC
TransE	**0.05**	**0.11**	0.24	0.29	3960.4	3890.6	0.78	0.79
SimResnik	**0.05**	0.09	0.25	0.30	1933.6	1864.4	0.88	0.89
SimLin	0.04	0.08	0.20	0.23	2287.9	2218.7	0.86	0.87
EL Embeddings	0.01	0.02	0.22	0.26	**1679.72**	**1637.65**	**0.90**	**0.90**
Onto2Vec	**0.05**	0.08	0.24	0.31	2434.6	2391.2	0.77	0.77
OPA2Vec	0.03	0.07	0.23	0.26	1809.7	1767.6	0.86	0.88
Random walk	0.04	0.10	**0.28**	**0.34**	1942.6	1958.6	0.85	0.86
Node2Vec	0.03	0.07	0.22	0.28	1860.5	1813.1	0.86	0.87

Table 6: Prediction performance for human protein–protein interactions.

The Jupyter Notebook *data.ipynb* in our repository provides source code to generate the datasets, the splits, and the input files for the different embedding methods.

We then generated ontology embeddings using EL Embeddings, Onto2Vec, OPA2Vec, and used the generated graph to produce embeddings through random walks, biased random walks (Node2Vec), and TransE. We then use these embeddings, as well as two semantic similarity measures (Resnik's and Lin's), to predict protein–protein interactions. For embeddings based on random walks, Onto2Vec, and OPA2Vec, we use cosine similarity to compute the pairwise similarity of all pairs of proteins in our dataset, including the proteins in the training, test, and validation sets; for TransE embeddings and EL Embeddings we use the prediction function for `interacts-with` edges and compute a prediction score for all pairs of proteins in our dataset; and we use the semantic similarity measures to compute the similarity between all pairs of proteins.

In the evaluation, for each protein p we rank all other proteins p_i based on their similarity to p. We then consider positives as pairs (p, p_k) which are PPIs included in our test set, and we report hits (recall) at ranks 10 and 100, mean rank at which the PPIs are found, and the ROCAUC (using micro-averages per protein). Results are separated in *Raw* and *Filtered*; *Raw* results evaluate all pairs of proteins while *Filtered* results evaluate all pairs of proteins except the pairs that are included in the training or validation sets. *Filtered* results are usually better since training pairs are not considered in the evaluation. We made Jupyter Notebooks available for all our experiments, and Table 5 summarizes the results for yeast and Table 6 for human; all results in these tables can be reproduced using the Jupyter Notebooks.

Overall, while our results are by no means a comprehensive evaluation and are limited to the task of predicting PPIs, we can obtain some information from our experiments. Traditional semantic similarity measures, in particular Resnik's measure [52], performs well across many evaluations, in particular in recall at the first ranks, and often has better performance than ontology embedding methods; this property is also of importance in other applications where Resnik's measure performs better than most other measures [30, 57]. Moreover, exploiting more of the axioms generally yields better results as can be seen when comparing EL Embeddings with most other methods. Furthermore, exploiting longer, or more indirect, relations, either through random walks or through utilizing the semantics, usually improves results over methods that are based on local properties or simply adjacency.

One experiment we did not perform here is to use the embeddings as part of a supervised machine learning model to predict the associations; such an approach has the potential to drastically improve the predictive performance results [30, 31], depending on the chosen machine learning model [23]. While we do not report on the use of embeddings for supervised learning here, we include an example of using the ontology embeddings in a Siamese neural network within the executable notebooks we provide.

7 Limitations and future work

Machine learning using ontologies involves a set of emerging techniques that have their roots in computer science and major applications in the life sciences where a large amount of ontologies have been developed and are applied to characterize data. Currently, several methods that allow background knowledge to be used by machine learning models are based on knowledge graphs and graph embeddings, and while these methods can be very successful, they lack the ability to utilize the model-theoretic semantics underlying ontologies. Ontologies, and representation artifacts based on similar formalisms, have the ability to represent more complex forms of knowledge, including using quantifiers, intersection, negation, and the ability to represent inconsistent knowledge. Strong negation, for example, is crucial in constraining search and cannot be substituted with the limited form of negation that is sometimes applied in knowledge graphs (i.e., the closed world assumption). However, while ontologies are able to express strong negation and other complex facts or rules, most ontology embedding methods are not yet able to adequately utilize them. Most syntactic and graph-based approaches do not interpret negation as constraints, or use any of the semantics associated with it, and can therefore not use negation to restrict search; and while model-based embeddings can utilize negation as part of the embedding they do not interact with the similarity measures or machine learning models that utilize them.

Several approaches aim to systematically integrate symbolic representations and machine learning are not yet widely applied to life science data and knowledge. Neuro-symbolic systems and neuro-symbolic integration [82, 83] provide a framework in which machine learning is integrated with symbolic representations; in the neuro-symbolic cycle, deductive inference is applied on the symbolic representations; embeddings project these representations into some space where they can be combined with data and where machine learning and optimization methods can be applied; and a knowledge extraction process maps the results back into the symbolic space. How to implement either of these projections is an active research area several of which we have reviewed here, and neuro-symbolic systems will put them together into a single framework. There is also recent interest in implementing the entire neuro-symbolic cycle, for example in vision [84]; however, with the rich set of formalized knowledge bases and the large amounts of data produced in the life sciences, we expect these systems to have major impact on how AI is applied in biology and biomedicine in the future.

Approaches to improve learning with ontologies while preserving and exploiting their semantics do not only include investigating embeddings into vector spaces (which, arguably, are mainly inspired by the needs of modern machine learning systems) but also approaches based on formal languages and logic, including Markov logic [85] and probabilistic inference [86]. Similarly, for extracting knowledge from data, new paradigms such as 'reinforcement learning as inference' [87] are increasingly being applied to generate explanations and representations that can be verified for consistency with background knowledge [88–90].

One main limitation of all the approaches we discussed here is their inability to consider quantitative information or data. In all cases, ontologies are used to model qualitative information and then possibly combined with other quantitative information after an embedding is generated; methods that can jointly learn on ontologies and quantitative information mapped to them include graph neural networks which will likely see increasing adoption in the coming years.

Funding

This work was supported by funding from King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. URF/1/3454-01-01, URF/1/3790-01-01, FCC/1/1976-04, FCC/1/1976-06, FCC/1/1976-17, FCC/1/1976-18, FCC/1/1976-23, FCC/1/1976-25, FCC/1/1976-26, and URF/1/3450-01.

References

Min, S., Lee, B., and Yoon, S. (2016). "Deep learning in bioinformatics". In: *Briefings in Bioinformatics* 18.5, pp. 851–869.

Feigenbaum, E. A. (1977). "The art of artificial intelligence – Themes and case studies of knowledge engineering". In: *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*. Vol. 2. CAMBRIDGE, MASSACHUSETTS: MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

Gkoutos, G. V., Green, E. C., Mallon, A. M., et al. (2004). "Building mouse phenotype ontologies." In: *Pac Symp Biocomput*, pp. 178–189.

Schindelman, G., Fernandes, J., Bastiani, C., et al. (2011). "Worm Phenotype Ontology: integrating phenotype data within and beyond the C. elegans community". In: *BMC Bioinformatics* 12.1, p. 32.

Deans, A. R., Lewis, S. E., Huala, E., et al. (2015). "Finding our way through phenotypes". In: *PLoS Biol.* 13.1, e1002033.

Oellrich, A., Collier, N., Groza, T., et al. (2016). "The digital revolution in phenotyping". In: *Brief. Bioinformatics* 17.5, pp. 819–830.

Robinson, P. N., Köhler, S., Bauer, S., et al. (2008). "The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease." In: *Am J Hum Genet* 83.5, pp. 610–615.

Koehler, S., Carmody, L., Vasilevsky, N., et al. (2019). "Expansion of the Human Phenotype Ontology (HPO) knowledge base and resources". In: *Nucleic Acids Res.* 47.D1, pp. D1018–D1027.

Gkoutos, G. V., Schofield, P. N., and Hoehndorf, R. (2017). "The anatomy of phenotype ontologies: principles, properties and applications". In: *Briefings in Bioinformatics*.

Mungall, C., Gkoutos, G., Smith, C., et al. (2010). "Integrating phenotype ontologies across multiple species". In: *Genome Biol* 11.1, R2+.

Grau, B., Horrocks, I., Motik, B., et al. (2008). "OWL 2: The next step for OWL". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 6.4, pp. 309–322.

Ashburner, M., Ball, C. A., Blake, J. A., et al. (2000). "Gene ontology: tool for the unification of biology". In: *Nature Genetics* 25.1, pp. 25–29.

Horridge, M., Bechhofer, S., and Noppens, O. (2007). "Igniting the OWL 1.1 Touch Paper: The OWL API". In: *Proceedings of OWLEd 2007: Third International Workshop on OWL Experiences and Directions.*

Noy, N. F., Sintek, M., Decker, S., et al. (2001). "Creating Semantic Web Contents with Protege-2000". In: *IEEE Intelligent Systems* 16.2, pp. 60–71.

Kazakov, Y., Krötzsch, M., and Simancik, F. (2014). "The Incredible ELK". English. In: *Journal of Automated Reasoning* 53.1, pp. 1–61.

Motik, B., Shearer, R., and Horrocks, I. (2009). "Hypertableau Reasoning for Description Logics". In: *Journal of Artificial Intelligence Research* 36, pp. 165–228.

Sirin, E. and Parsia, B. (2004). "Pellet: An OWL DL Reasoner". In: *Proceedings of the 2004 International Workshop on Description Logics, DL2004, Whistler, British Columbia, Canada, Jun 6-8.* Ed. by V. Haarslev and R. Möller. Vol. 104. CEUR Workshop Proceedings. Aachen, Germany: CEUR-WS.org.

Rodriguez-Garcia, M. A. and Hoehndorf, R. (2018). "Inferring ontology graph structures using OWL reasoning". In: *BMC Bioinformatics* 19.1, p. 7.

Harispe, S. et al. (2014). "The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies". In: *Bioinformatics* 30.5, pp. 740–742.

Zhu, G. and Iglesias, C. A. (2017). "Computing Semantic Similarity of Concepts in Knowledge Graphs". In: *IEEE Transactions on Knowledge and Data Engineering* 29.1, pp. 72–85.

Couto, F. M. and Lamurias, A. (2019). "Semantic Similarity Definition". In: *Encyclopedia of Bioinformatics and Computational Biology.* Ed. by S. Ranganathan, M. Gribskov, K. Nakai, et al. Oxford: Academic Press, pp. 870–876.

Holter, O. M., Myklebust, E. B., Chen, J., et al. (2019). "Embedding OWL ontologies with OWL2Vec". In: *CEUR Workshop Proceedings.* Vol. 2456, pp. 33–36.

Chen, J., Althagafi, A., and Hoehndorf, R. (2020). "Predicting candidate genes from phenotypes, functions, and anatomical site of expression". In: *bioRxiv.*

Alshahrani, M., Khan, M. A., Maddouri, O., et al. (2017). "Neuro-symbolic representation learning on biological knowledge graphs". In: *Bioinformatics* 33.17, pp. 2723–2730.

Ristoski, P. and Paulheim, H. (2016). "RDF2Vec: RDF Graph Embeddings for Data Mining". In: *International Semantic Web Conference.*

Grover, A. and Leskovec, J. (2016). "Node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 855–864.

Ali, M., Jabeen, H., Hoyt, C. T., et al. (2019). "The KEEN Universe: An Ecosystem for Knowledge Graph Embeddings with a Focus on Reproducibility and Transferability". In: *Proceedings of the International Semantic Web Conference (ISWC) 2019.*

Ali, M., Hoyt, C. T., Domingo-Fern?ndez, D., et al. (2019). "BioKEEN: a library for learning and evaluating biological knowledge graph embeddings". In: *Bioinformatics* 35.18, pp. 3538–3540.

Fey, M. and Lenssen, J. E. (2019). "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds.*

Smaili, F. Z., Gao, X., and Hoehndorf, R. (2018). "Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations". In: *Bioinformatics* 34.13, pp. i52–i60.

— (2019). "Opa2vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction". In: *Bioinformatics* 35.12, pp. 2133–2140.

Kulmanov, M., Liu-Wei, W., Yan, Y., et al. (2019). "EL Embeddings: Geometric construction of models for the Description Logic EL++". In: *Proceedings of IJCAI 2019*. IJCAI.

Kulmanov, M., Khan, M. A., and Hoehndorf, R. (2018). "DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier". In: *Bioinformatics* 34.4, pp. 660–668.

Sureyya Rifaioglu, A., Doğan, T., Jesus Martin, M., et al. (2019). "DEEPred: Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks". In: *Scientific Reports* 9.1, p. 7344.

Wang, J., Zhang, J., Cai, Y., et al. (2019). "DeepMiR2GO: Inferring Functions of Human MicroRNAs Using a Deep Multi-Label Classification Model". In: *International journal of molecular sciences* 20.23, p. 6046.

Gruber, T. R. (1993). "Towards Principles for the Design of Ontologies Used for Knowledge Sharing". In: *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Ed. by N. Guarino and R. Poli. Deventer, The Netherlands: Kluwer Academic Publishers.

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). "The Semantic Web". In: *Scientific American* 284.5, pp. 28–37.

Baader, F. (2003). *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press.

Smith, B., Ceusters, W., Klagges, B., et al. (2005). "Relations in biomedical ontologies." In: *Genome Biol* 6.5, R46.

Beckett, D. (2004). *RDF/XML Syntax Specification (Revised)*. W3C Recommendation. World Wide Web Consortium (W3C).

Santana da Silva, F., Jansen, L., Freitas, F., et al. (2017). "Ontological interpretation of biomedical database content". In: *J Biomed Semantics* 8.1, p. 24.

Hoehndorf, R., Mencel, L., Gkoutos, G. V., et al. (2016). "Large-Scale Reasoning over Functions in Biomedical Ontologies". In: *Formal Ontology in Information Systems*. Vol. 283. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 299–312.

Zhang, S.-B. and Tang, Q.-R. (2016). "Protein–protein interaction inference based on semantic similarity of Gene Ontology terms". In: *Journal of Theoretical Biology* 401, pp. 30–37.

Schlicker, A. and Albrecht, M. (2009). "FunSimMat update: new features for exploring functional similarity". In: *Nucleic Acids Research*.

Smedley, D., Oellrich, A., Köhler, S., et al. (2013). "PhenoDigm: analyzing curated annotations to associate animal models with human diseases". In: *Database* 2013.

Köhler, S., Schulz, M. H., Krawitz, P., et al. (2009). "Clinical Diagnostics in Human Genetics with Semantic Similarity Searches in Ontologies". In: *The American Journal of Human Genetics* 85.4, pp. 457–464.

Robinson, P. N., Köhler, S., Oellrich, A., et al. (2014). "Improved exome prioritization of disease genes through cross-species phenotype comparison". In: *Genome Res* 24.2, pp. 340–348.

Lord, P. W., Stevens, R. D., Brass, A., et al. (2003). "Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation". In: *Bioinformatics* 19.10, pp. 1275–1283.

Radivojac, P., Clark, W. T., Oron, T. R., et al. (2013). "A large-scale evaluation of computational protein function prediction". In: *Nat Meth* 10.3, pp. 221–227.

Rada, R., Mili, H., Bicknell, E., et al. (1989). "Development and application of a metric on semantic nets". In: *IEEE Transactions on Systems, Man, and Cybernetics* 19.1, pp. 17–30.

Harispe, S. et al. (2015). "Semantic Similarity from Natural Language and Ontology Analysis". In: *Synthesis Lectures on Human Language Technologies* 8.1, pp. 1–254.

Resnik, P. (1995). "Using Information Content to Evaluate Semantic Similarity in a Taxonomy". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., pp. 448–453.

Lin, D. (1998). "An Information-Theoretic Definition of Similarity". In: *In Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, pp. 296–304.

Jiang, J. J. and Conrath, D. W. (1997). "Semantic similarity based on corpus statistics and lexical taxonomy". In: *Proc of 10th International Conference on Research in Computational Linguistics, ROCLING'97*.

Pesquita, C., Faria, D., Bastos, H., et al. (2007). "Evaluating GO-based semantic similarity measures". In: *PROCEEDINGS OF THE 10TH ANNUAL BIO-ONTOLOGIES MEETING (BIOONTOLOGIES*, pp. 37–40.

Pesquita, C., Faria, D., Falcao, A. O., et al. (2009). "Semantic Similarity in Biomedical Ontologies". In: *PLoS Comput Biol* 5.7, e1000443.

Kulmanov, M. and Hoehndorf, R. (2017). "Evaluating the effect of annotation size on measures of semantic similarity". In: *Journal of Biomedical Semantics* 8.1, p. 7.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: ACM, pp. 701–710.

Mikolov, T., Sutskever, I., Chen, K., et al. (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119.

Levy, O. and Goldberg, Y. (2014). "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, et al. Curran Associates, Inc., pp. 2177–2185.

Koehler, S., Bauer, S., Horn, D., et al. (2008). "Walking the interactome for prioritization of candidate disease genes". In: *Am. J. Hum. Genet.* 82.4, pp. 949–958.

Smedley, D., Koehler, S., Czeschik, J. C., et al. (2014). "Walking the interactome for candidate prioritization in exome sequencing studies of Mendelian diseases". In: *Bioinformatics* 30.22, pp. 3215–3222.

ben-Avraham, D. and Havlin, S. (2000). *Diffusion and Reaction in Fractals and Disordered Systems*. Cambridge, UK: Cambridge University Press.

Bordes, A., Usunier, N., Garcia-Duran, A., et al. (2013). "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, et al. Curran Associates, Inc., pp. 2787–2795.

Wang, Z., Zhang, J., Feng, J., et al. (2014). "Knowledge Graph Embedding by Translating on Hyperplanes". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. Québec City, Québec, Canada: AAAI Press, pp. 1112–1119.

Wang, Q., Mao, Z., Wang, B., et al. (2017). "Knowledge Graph Embedding: A Survey of Approaches and Applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12, pp. 2724–2743.

Nickel, M., Murphy, K., Tresp, V., et al. (2016). "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proceedings of the IEEE* 104, pp. 11–33.

Ren, H., Hu, W., and Leskovec, J. (2020). "Query2box: Reasoning over Knowledge Graphs in Vector Space using Box Embeddings". In: *Proceedings of ICLR 2020*.

Maaten, L. van der and Hinton, G. (2008). "Visualizing High-Dimensional Data Using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

Silla Jr., C. N. and Freitas, A. A. (2011). “A Survey of Hierarchical Classification Across Different Application Domains”. In: *Data Min. Knowl. Discov.* 22.1-2, pp. 31–72.

Radivojac, P. and Clark, W. T. (2013). “Information-theoretic evaluation of predicted ontological annotations”. In: *Bioinformatics* 29.13, pp. i53–i61.

Notaro, M., Schubach, M., Robinson, P. N., et al. (2017). “Prediction of Human Phenotype Ontology terms by means of hierarchical ensemble methods”. In: *BMC Bioinformatics* 18.1, p. 449.

SOKOLOV, A. and BEN-HUR, A. (2010). “HIERARCHICAL CLASSIFICATION OF GENE ONTOLOGY TERMS USING THE GOstruct METHOD”. In: *Journal of Bioinformatics and Computational Biology* 08.02, pp. 357–376.

Kahanda, I., Funk, C., Verspoor, K., et al. (2015). “PHENOstruct: Prediction of human phenotype ontology terms using heterogeneous data sources [version 1; referees: 2 approved]”. In: *F1000Research* 4.259.

Feng, S., Fu, P., and Zheng, W. (2018). “A hierarchical multi-label classification method based on neural networks for gene function prediction”. In: *Biotechnology & Biotechnological Equipment* 32.6, pp. 1613–1621.

Wang, H., Dou, D., and Lowd, D. (2016). “Ontology-Based Deep Restricted Boltzmann Machine”. In: *Proceedings, Part I, 27th International Conference on Database and Expert Systems Applications - Volume 9827*. DEXA 2016. Porto, Portugal: Springer-Verlag, pp. 431–445.

Whetzel, P. L., Noy, N. F., Shah, N. H., et al. (2011). “BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications”. In: *Nucleic Acids Research* 39.suppl2, W541–W545.

Gottlieb, A., Stein, G. Y., Ruppin, E., et al. (2011). “PREDICT: a method for inferring novel drug indications with application to personalized medicine”. In: *Mol Syst Biol* 7, p. 496.

Campillos, M., Kuhn, M., Gavin, A.-C. C., et al. (2008). “Drug target identification using side-effect similarity.” In: *Science* 321.5886, pp. 263–266.

Hoehndorf, R., Schofield, P. N., and Gkoutos, G. V. (2011). “PhenomeNET: a whole-phenome approach to disease gene discovery”. In: *Nucleic Acids Res* 39.18, e119.

Szklarczyk, D., Gable, A. L., Lyon, D., et al. (2018). “STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets”. In: *Nucleic Acids Research* 47.D1, pp. D607–D613.

Garcez, A. d’Avila, Besold, T., Raedt, L. de, et al. (2015). *Neural-Symbolic Learning and Reasoning: Contributions and Challenges.*

Besold, T. R., Garcez, A. S. d’Avila, Bader, S., et al. (2017). “Neural-Symbolic Learning and Reasoning: A Survey and Interpretation”. In: *CoRR* abs/1711.03902.

Mao, J., Gan, C., Kohli, P., et al. (2019). “The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision”. In: *International Conference on Learning Representations.*

Richardson, M. and Domingos, P. (2006). “Markov logic networks”. In: *Machine Learning* 62, pp. 107–136.

Goertzel, B. (2008). *Probabilistic logic networks : a comprehensive conceptual, mathematical and computational framework for uncertain inference.* New York, London: Springer.

Levine, S. (2018). “Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review”. In: *CoRR* abs/1805.00909.

Saxton, D., Grefenstette, E., Hill, F., et al. (2019). “Analysing Mathematical Reasoning Abilities of Neural Models”. In: *CoRR* abs/1904.01557.

Evans, R., Saxton, D., Amos, D., et al. (2018). “Can Neural Networks Understand Logical Entailment?” In: *CoRR* abs/1802.08535.

Evans, R. and Grefenstette, E. (2017). "Learning Explanatory Rules from Noisy Data". In: *CoRR* abs/1711.04574.