OWL2Vec*: embedding of OWL ontologies

Jiaoyan Chen[1] · Pan Hu[1] · Ernesto Jimenez-Ruiz[2,3] · Ole Magnus Holter[3] · Denvar Antonyrajah[4] · Ian Horrocks[1]

Abstract

Semantic embedding of knowledge graphs has been widely studied and used for prediction and statistical analysis tasks across various domains such as Natural Language Processing and the Semantic Web. However, less attention has been paid to developing robust methods for embedding OWL (Web Ontology Language) ontologies, which contain richer semantic information than plain knowledge graphs, and have been widely adopted in domains such as bioinformatics. In this paper, we propose a random walk and word embedding based ontology embedding method named OWL2Vec*, which encodes the semantics of an OWL ontology by taking into account its graph structure, lexical information and logical constructors. Our empirical evaluation with three real world datasets suggests that OWL2Vec* benefits from these three different aspects of an ontology in class membership prediction and class subsumption prediction tasks. Furthermore, OWL2Vec* often significantly outperforms the state-of-the-art methods in our experiments.

Keywords Ontology · Ontology embedding · Word embedding · Web ontology language · OWL2Vec* · Ontology completion

1 Introduction

In recent years, the semantic embedding of knowledge graphs (KGs) has been widely investigated (Wang et al., 2017). The objective of such embeddings is to represent in a vector space KG components such as entities and relations in a way that captures the structure of the graph. Various kinds of KG embedding algorithms have been proposed and

✉ Jiaoyan Chen
  jiaoyan.chen@cs.ox.ac.uk

[1] Department of Computer Science, University of Oxford, Oxford, UK

[2] City, University of London, London, UK

[3] Department of Informatics, University of Oslo, Oslo, Norway

[4] Samsung Research, Staines, UK

successfully applied to KG refinement (*e.g.*, link prediction (Rossi et al., 2020) and entity alignment Sun et al. 2020), recommendation systems (Ristoski et al., 2019), zero-shot learning (Chen et al., 2020c; Wang et al., 2018), interaction prediction in bioinformatics (Smaili et al., 2018; Myklebust et al., 2019), and so on. However, most of these algorithms focus on creating embeddings for multi-relational graphs composed of RDF (Resource Description Framework) triples such as ⟨*England*,[1] *isPartOf*, *UK*⟩ and ⟨*UK*, *hasCapital*, *London*⟩.[2] They do not deal with OWL ontologies (or ontological schemas in OWL) which include not only graph structures,[3] but also logic constructors such as class disjointness, existential and universal quantification (*e.g.*, *a country must have at least one city as its capital*), and meta data such as the synonyms, definitions and comments of a class. OWL ontologies have been widely used in many domains such as bioinformatics, the Semantic Web and Linked Data (Myklebust et al., 2019; Horrocks, 2008). They are capable of expressing complex domain knowledge and managing large scale domain vocabularies, and can often improve the quality and usability of the KG (Paulheim & Gangemi, 2015; Chen et al., 2020a).

Inspired by the success of KG embeddings, more recently there has been a growing interest in embedding simple ontological schemas consisting, *e.g.*, of hierarchical classes, and property domain and range (Hao et al., 2019; Moon et al., 2017; Alshargi et al., 2018; Guan et al., 2019); however, these methods rely on having a large number of facts (*i.e.*, an ABox), and do not support more expressive OWL ontologies which contain some widely used logic constructors such as the class disjointness and the existential quantification mentioned above. Embeddings for OWL ontologies have started to receive some attention recently. Kulmanov et al. (2019) and Garg et al. (2019) proposed to model the semantics of the logic constructor by geometric learning, but their models only support some of the logic constructors from the description logics (DLs) $\mathcal{EL}^{++}$ (which is closely related to OWL EL – a fragment of OWL) and $\mathcal{ALC}$, respectively. Moreover, both methods consider only the logical and graph structure of an ontology, and ignore its lexical information that widely exists in the meta data (*e.g.*, *rdfs:label* and *rdfs:comment* triples). OPA2Vec (Smaili et al., 2018) considers the ontology's lexical information by learning a word embedding model which encodes statistical correlations between items in a corpus. However, it treats each axiom as a sentence and fails to explore and utilize the semantic relationships between axioms. OWL2Vec* (Holter et al., 2019), which is our very preliminary work before OWL2Vec*, captures the semantics of OWL ontologies by exploring the neighborhoods of classes. This was shown to be quite effective, but it does not fully exploit the graph structure, the lexical information, or the logical semantics available in OWL ontologies.

In this work we have extended OWL2Vec* in order to provide a more general and robust OWL ontology embedding framework which we call OWL2Vec* . OWL2Vec* exploits an OWL (or OWL 2) ontology by walking over its graph forms and generates a corpus of three documents that capture different aspects of the semantics of the ontology: *(i)* the graph structure and the logic constructors, *(ii)* the lexical information (*e.g.*, entity names, comments and definitions), and *(iii)* a combination of the lexical information, graph structure and logical constructors. Finally, OWL2Vec* uses a word

---

[1] https://www.w3.org/RDF/

[2] https://www.w3.org/OWL/

[3] In this paper, an ontology's graph structure includes the relation between instances (*e.g.*, *isPartOf*) as in RDF KGs, the subsumption relation between classes (*i.e.*, the class hierarchy defined by *rdfs:subClassOf*), the membership relation between instances and classes, and the relation between instances and literals.

embedding model to create embeddings of both entities and words from the generated corpus. Note that the OWL2Vec* framework is compatible with different word embedding methods and their different settings, although the current implementation adopts *Word2Vec* (Mikolov et al., 2013b) and its skip-gram architecture.

We have evaluated OWL2Vec* in two case studies — class membership prediction and class subsumption prediction, using three large scale real world ontologies — a healthy lifestyle ontology named HeLis (Dragoni et al., 2018), a food ontology named FoodOn (Dooley et al., 2018) and the Gene Ontology (GO) (G.O. 2008). In the case studies we empirically analyze the impact of *(i)* different document and embedding settings which correspond to combinations of the semantics of the graph structure, lexical information and logic constructors, *(ii)* different graph structure exploration settings (*e.g.*, the transformation methods from an OWL ontology to an RDF graph, and the graph walking strategies), *(iii)* ontology entailment reasoning, and *(iv)* word embedding pre-training. The results suggest that OWL2Vec* can achieve significantly better performance than the baselines including the state-of-the-art ontology embeddings (Kulmanov et al., 2019; Garg et al., 2019; Smaili et al., 2018; Holter et al., 2019), some classic KG embeddings such as RDF2Vec (Ristoski and Paulheim 2016), TransE (Bordes et al., 2013) and DistMult (Yang et al., 2014), and two supervised Transformer (Vaswani et al., 2017) classifiers based on the textual context. We also calculated the Euclidean distance between entities and visualized the embeddings of some example entities to analyze different embedding methods.

Briefly this study can be summarized as follows.

- This work is among the first that aim at embedding all kinds of semantics of OWL ontologies including the graph structure, the literals and the logical constructors. A general framework named OWL2Vec* together with different strategies for addressing different OWL semantics has been developed. OWL2Vec* has the potential to *(i)* enable statistical or machine learning tasks over massive ontologies, thus assisting their curation and boosting their application, *(ii)* facilitate the integration of symbolic and sub-symbolic systems into new neural-symbolic solutions.
- The work has evaluated OWL2Vec* in two important ontology completion case studies (class membership prediction and class subsumption prediction) on three real world ontologies, where OWL2Vec* outperforms both state-of-the art ontology embedding and classic KG embedding methods. We have also conducted extensive ablation studies to verify the adopted strategies as well as visualization analysis to facilitate interpretation.

The remainder of the paper is organized as follows. The next section introduces the preliminaries including both background and related work. Section 3 introduces the technical details of OWL2Vec* as well as the case studies. Section 4 presents the experiments and the evaluation results. The last section concludes and discusses future work.

2 Preliminaries

2.1 OWL ontologies

Our OWL2Vec* embedding targets OWL ontologies (Bechhofer et al., 2004), which are based on the $\mathcal{SROIQ}$ description logic (DL) (Baader et al., 2017). Consider a signature $\Sigma = (\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$, where $\mathcal{N}_C$, $\mathcal{N}_R$ and $\mathcal{N}_I$ are pairwise disjoint sets of, respectively, atomic concepts, atomic roles and individuals. Complex concepts and roles can be composed using DL constructors such as conjunction (*e.g.*, $C \sqcap D$), disjunction (*e.g.*, $C \sqcup D$), existential restriction (*e.g.*, $\exists r.C$) and universal restrictions (*e.g.*, $\forall r.C$) where C and D are concepts, and r is a role. An OWL ontology comprises a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The TBox is a set of axioms such as General Concept Inclusion (GCI) axioms (*e.g.*, $C \sqsubseteq D$), Role Inclusion (RI) axioms (*e.g.*, $r \sqsubseteq s$) and Inverse Role axioms (*e.g.*, $s \equiv r^-$), where C and D are concepts, r and s are roles, and r^- denotes the inverse of r. The ABox is a set of assertions such as concept assertions (*e.g.*, $C(a)$), role assertions (*e.g.*, $r(a, b)$) and individual equality and inequality assertions (*e.g.*, $a \equiv b$ and $a \not\equiv b$), where C is a concept, r is a role, a and b are individuals. It is worth noting that the terminological part can also be divided into a TBox and an RBox, where the RBox models the interdependencies between the roles such as the RI.

In OWL, the aforementioned concept, role and individual are modeled as *class*, *object property* and *instance*, respectively. To avoid confusion, we will only use the terms of class, object property and instance in the remainder of the paper. We will also use the term *property* which can refer to not only object property, but also *data property* and *annotation property*. Meanwhile, for convenience, we will also use a general term *entity* to refer to a class, a property or an instance. Note that an object property models the relationship between two instances; a data property models the relationship between an instance and a literal value (*e.g.*, number or text); and an annotation property is used to represent a (non-logical) relationship between an entity and an annotation (*e.g.*, comment or label). Each entity is uniquely represented by an Internationalized Resource Identifier (IRI).[4] These IRIs may be lexically 'meaningful' (*e.g.*, *vc:AlcoholicBeverages* in Fig. 1a) or consist of internal IDs that do not carry useful lexical information (*e.g.*, *obo:FOODON_00002809* in Fig. 1b); in either case the intended meaning may also be indicated via annotations (see below). To enhance the reading, we will typically show together with the ID-based IRI of an entity a readable label from its available annotations.

In OWL, a GCI axiom $C \sqsubseteq D$ corresponds to a subsumption relation between the class C and the class D, while a concept assertion $C(a)$ corresponds to a membership relation between the instance a and the class C. Meanwhile, in OWL, complex concepts, complex roles, axioms and role assertions can be serialised as (sets of) RDF triples, each of which is a tuple composed of a subject, a predicate and an object. For the predicate, these triples use a combination of bespoke object properties (*e.g.*, *vc:hasNutrient*), and built-in properties by RDF, RDFS and OWL (*e.g.*, *rdfs:subClassOf*,[5] *rdf:type* and *owl:someValuesFrom*). In Fig. 1, for example, the relationship between the two instances *vc:FOOD-4001 (blonde beer)* and *vc:VitaminC_100* is represented by a triple using

---

[4] An entity can also be represented by an Uniform Resource Identifier (URI). IRIs extend URIs by using the Universal Character Set, where URIs were limited to ASCII, with far fewer characters.

[5] https://www.w3.org/TR/rdf-schema/

(a) The HeLis Ontology

(b) The FoodOn Ontology
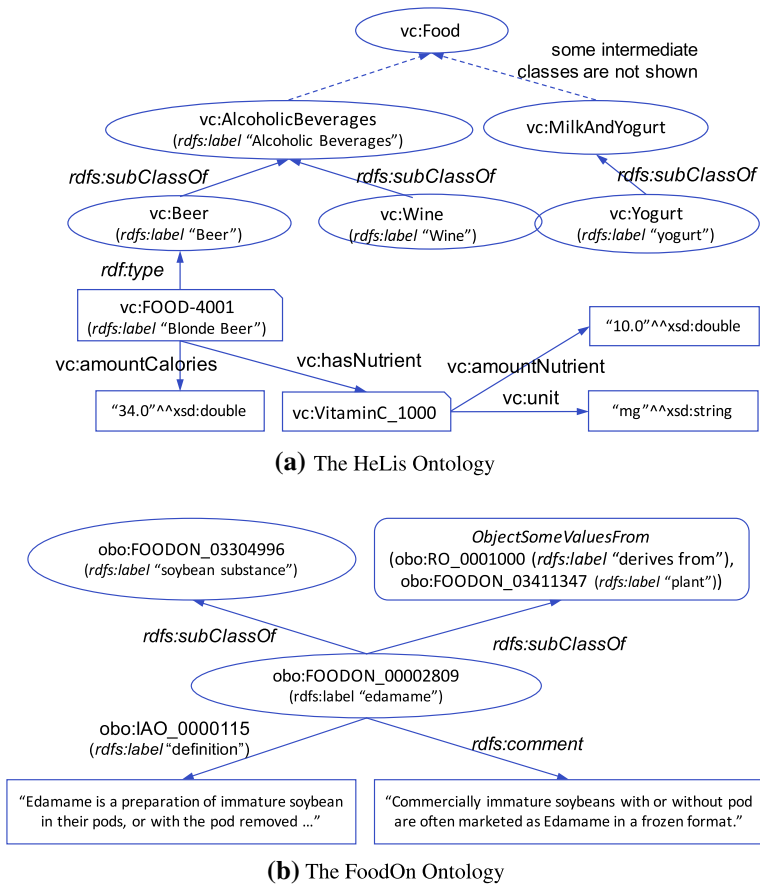
Fig. 1 Fragments of the ontologies. Note *vc* is the prefix associated to the IRI namespace of http://www.fbk. eu/ontologies/virtualcoach#, while *obo*, *oboInOwl*, *xsd*, *rdf*, *rdfs* and *owl* are prefixes of standard vocabularies

the object property *vc:hasNutrient*, while the existential restriction involving the class *obo:FOODON_00002809 (edamame)* and the object property *obo:RO_0001000 (derives from)* is represented by triples using three OWL built-in properties, *i.e.*, *owl:Restriction*, *owl:onProperty* and *owl:someValuesFrom*. The object of an RDF triple of an OWL assertion can be a literal value; for example, the calories amount of *vc:FOOD-4001 (blonde beer)* is represented by a triple using the bespoke data property *vc:amountCalories* and the literal value 34.0 of type *xsd:double*.

In addition to axioms and assertions with formal logic-based semantics, an ontology often contains metadata information in the form of annotation axioms. These annotations can also be represented by RDF triples using annotation properties as the predicates; *e.g.*, the class *obo:FOODON_00002809 (edamame)* is annotated using *rdfs:label* to specify

a name string, using *rdfs:comment* to specify a description, and using *obo:IAO-0000115 (definition)* — a bespoke annotation property to specify a natural language "definition".

Knowledge graph (KG) refers to structured knowledge resources which are often expressed as a set of RDF triples (Hogan et al., 2020). Many KGs only contain instances and facts which are equivalent to an OWL ontology ABox. Some other KGs such as DBpedia (Auer et al., 2007) are also enhanced with an schema which is equivalent to the TBox of an OWL ontology. Thus, a KG can often be understood as an ontology.

2.2 Semantic embedding

Semantic embedding refers to a series of representation learning (or feature learning) techniques that encode the semantics of data such as sequences and graphs into vectors, such that they can be utilized by downstream machine learning prediction and statistical analysis tasks (Bengio et al., 2013). Word embedding or sequence feature learning models such as Feed-Forward Neural Networks, Recurrent Neural Networks and Transformers are widely used for semantic embedding, and they have shown good performance in embedding the context (*e.g.*, item co-occurrence) in sequences (Mikolov et al., 2013a; Peters et al., 2018; Devlin et al., 2019). Two classic auto-encoding architectures for learning representations of sequential items are continuous skip-gram and continuous Bag-of-Words (CBOW) (Mikolov et al., 2013b, a). The former aims at predicting the surroundings of an item, while the latter aims at predicting an item based on its surroundings. *Word2Vec* is a well known group of sequence feature learning techniques for learning word embeddings from a large corpus, and was initially developed by a team at Google; it can be configured to use either skip-gram or CBOW architectures (Mikolov et al., 2013b, a).

Semantic embedding has also been extended to KGs composed of role assertions (Wang et al., 2017). The entities and relations (object properties) are represented in a vector space while retaining their relative relationships (semantics), and the resulting vectors are then applied to downstream tasks including link prediction (Rossi et al., 2020), entity alignment (Sun et al., 2020), and erroneous fact detection and correction (Chen et al., 2020a). One paradigm for learning KG representations is computing the embeddings in an *end-to-end* manner, iteratively adjusting the vectors using an optimization algorithm to minimize the overall loss across all the triples, where the loss is usually calculated by scoring the truth/falsity of each triple (positive and negative samples). Algorithms based on this technique include translation based models such as TransE (Bordes et al., 2013) and TransR (Lin et al., 2015) and latent factor models such as DistMult (Yang et al., 2014).

Another paradigm is to first explicitly explore the neighborhoods of entities and relations in the graph, and then learn the embeddings using a word embedding model. Two representative algorithms based on this paradigm are node2vec (Grover & Leskovec, 2016) and Deep Graph Kernels (Yanardag and Vishwanathan 2015). The former extracts random graph walks and creates skip-gram or CBOW models as the corpus for training, while the latter uses graph kernels such as Weisfeiler-Lehman (WL) sub-graph kernels as the corpus. However, both embedding algorithms were originally developed for undirected graphs, and thus may have limited performance when directly applied to KGs. RDF2Vec addresses this issue by extending the idea of the above two algorithms to directed labeled RDF graphs,

and has been shown to learn effective embeddings for large scale KGs such as DBpedia (Ristoski & Paulheim, 2016; Ristoski et al., 2019). Recent studies have explored the usage of new word embedding or sequence feature learning models for learning embeddings; one example is the KG embedding and link prediction method named RW-LMLM which combines the random walk algorithm with Transformer (Wang et al., 2019).

Our OWL2Vec* technique belongs to the word embedding paradigm, but we focus on OWL ontologies instead of typical KGs, with the goal of preserving the semantics not only of the graph structure, but also of the lexical information and the logical constructors. Note that the graph of an ontology, which includes hierarchical categorization structure, differs from the multi-relation graph composed of role (relation) assertions of a typical KG; furthermore, according to our literal review on ontology embedding (cf. Sect. 2.3) and the latest survey (Kulmanov et al., 2020), there are currently no existing KG embedding methods that jointly explore the ontology's lexical information and logical constructors.
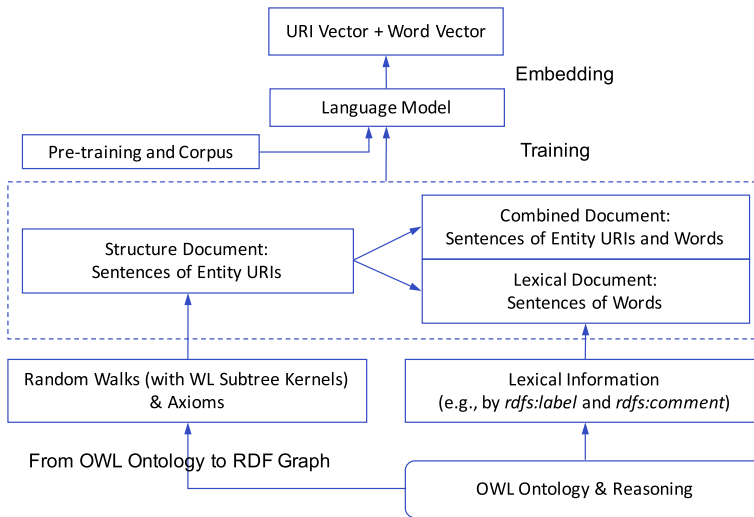
2.3 Ontology embedding

The use of machine learning prediction and statistical analysis with ontologies is receiving wider attention, and some approaches to embedding the semantics of OWL ontologies can already be found in the literature. Unlike typical KGs, OWL ontologies include not only graph structure but also logical constructors, and entities are often augmented with richer lexical information specified using *rdfs:label*, *rdfs:comment* and many other bespoke or built-in annotation properties. The objective of OWL ontology embedding in this study is to represent each OWL named entity (class, instance or property) by a vector, such that the inter-entity relationships indicated by the above information are kept in the vector space, and the performance of the downstream tasks, where the input vectors can be understood as learned features, is maximized.

EL Embedding (Kulmanov et al., 2019) and Quantum Embedding (Garg et al., 2019) are two OWL ontology embedding algorithms of the end-to-end paradigm. They construct specific score functions and loss functions for logical axioms from $\mathcal{EL}^{++}$ and $\mathcal{ALC}$, respectively, by transforming logical relations into geometric relations. This encodes the semantics of the logical constructors, but ignores the additional semantics provided by the lexical information of the ontology. Moreover, although the graph structure is explored by considering class subsumption and class membership axioms, the exploration is incomplete as it uses only *rdfs:subClassOf* and *rdf:type* edges, and ignores edges involving other relations.

Onto2Vec (Smaili et al., 2018) and OPA2Vec (Smaili et al., 2018) are two ontology embedding algorithms of the word embedding paradigm using a model of either the skip-gram architecture or the CBOW architecture. Onto2Vec uses the axioms of an ontology as the corpus for training, while OPA2Vec complements the corpus of Onto2Vec with the lexical information provided by, *e.g.*, *rdfs:comment*. Both adopt the deductive closure of an ontology with entailment reasoning. They have been evaluated with the Gene Ontology for predicting protein-protein interaction (*i.e.*, a domain-specific relationship between classes), which is quite different from the class membership prediction and the class subsumption prediction in this study. Both methods treat each axiom as a sentence, which means that they cannot explore the correlation between axioms. This makes it hard to fully explore the graph structure and the logical relation between axioms, and may also lead to the problem of corpus shortage for small to medium scale ontologies. OWL2Vec* deals with the above issues of OPA2Vec and Onto2Vec by complementing their axiom corpus with a corpus generated by walking over RDF graphs that are transformed from the OWL ontology with

Fig. 2 The overall framework of OWL2Vec*

its graph structure and logical constructors considered. In addition, to fully utilize the lexical information, OWL2Vec* creates embeddings for not only the ontology entities as the current KG/ontology embedding methods but also for the words in the lexical information.

3 Methodology

Figure 2 presents the overall framework of OWL2Vec*, which mainly consists of two core steps: *(i)* corpus extraction from the ontology, and *(ii)* word embedding model training with the corpus. The corpus includes a structure document, a lexical document, and a combined document. The first two documents aim at exploring the ontology's graph structure, logical constructors and lexical information, where ontology entailment reasoning can be enabled, while the third document aims at preserving the correlation between entities (IRIs) and their lexical labels (words). Note that the latter two documents are constructed using the first document as the backbone while taking into account the lexical information available from the ontology. See Table 2 for sentence examples of each document. Briefly, given an input ontology $\mathcal{O}$ and the target entities E of $\mathcal{O}$ for embedding, OWL2Vec* outputs a vector for each entity e in E, denoted as $\boldsymbol{e} \in \mathbb{R}^d$, where d is the (configurable) embedding dimension. Note that E can be all the entities in $\mathcal{O}$ or just a part needed for a specific application. With the OWL2Vec* embeddings, we apply them in two downstream case studies — class membership prediction and class subsumption prediction. For class membership prediction we set E to all the named classes and instances; for class subsumption prediction we set E to all the named classes.

Table 1 Projection rules, based on Soylu et al. (2018), Holter et al. (2019), used in the second strategy to generate an RDF graph

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

$\Box$ is one of: $\geq, \leq, =, \exists, \forall$. A, B, B_i and C_i are atomic concepts (classes), s_i, r and r' are roles (object properties), r^- is the inverse of a relation r, a and b are individuals (instances), $\top$ is the top concept (defined by *owl:Thing*)

3.1 From OWL ontology to RDF graph

OWL2Vec* incorporates two strategies to turn the original OWL ontology $\mathcal{O}$ into a graph $\mathcal{G}$ that is composed of RDF triples. The first strategy is the transformation according to OWL to RDF Graph Mapping which is originally defined by the W3C[6] to store and exchange OWL ontologies as RDF triples. Some simple axioms such as membership and subsumption axioms for atomic entities, data and annotation properties associated to atomic entities, and relational assertions between atomic instances can be directly transformed into RDF triples by introducing some built-in properties or using the bespoke properties in the axioms (*e.g.*, ⟨*vc:FOOD-4001 (Blonde Beer), rdf:type, vc:Beer*⟩, ⟨*vc:FOOD-4001, rdfs:label*, "Blonde Beer"⟩ and ⟨*vc:FOOD-4001, vc:hasNutrient, vc:VitaminC_1000*⟩). Axioms involving complex class expressions need to be transformed into multiple triples and often rely on blank nodes. For example, the existential restriction of the class *obo:FOODON_00002809 (edamame)* in Fig. 1b, *i.e.*, *ObjectSomeValuesFrom(obo:RO_0001000 (derives from), obo:FOODON_03411347 (plant))* is transformed into four RDF triples, *i.e.*, ⟨*obo:FOODON_00002809, rdfs:subClassOf, _:x*⟩, ⟨*_:x, owl:someValuesFrom, obo:FOODON_03411347* ⟩, ⟨*_:x, rdf:type, owl:Restriction*⟩ and ⟨*_:x, owl:onProperty, obo:RO_0001000* ⟩, where *_:x* denotes a blank node. In this example, one additional node *_:x* and one additional edge *rdfs:subClassOf* are inserted between *obo:FOODON_00002809* and *obo:FOODON_03411347*.

The second strategy is based on projection rules proposed in Soylu et al., (2018), Holter et al., (2019), as shown in Table 1, where every RDF triple ⟨*X, r, Y*⟩ in the projection (the third column) is justified by one or more axioms in the ontology (the first

---

[6] https://www.w3.org/TR/owl2-mapping-to-rdf/

and second columns). As in the first strategy, a simple relational assertion between two atomic entities (the final row in Table 1), or a simple data or annotation property associated to an atomic entity, is directly transformed into one single triple. While those complex logical constructors (the first six rows in Table 1), unlike the first strategy, are approximated. For example, the above mentioned existential restriction of the class *obo:FOODON_00002809* would be represented with ⟨*obo:FOODON_00002809*, *obo:RO_0001000*, *obo:FOODON_03411347*⟩. This avoids the use of blank nodes in the RDF graph, which may act as noise towards the correlation between entities when the embeddings are learned; but, the exact logical relationships are not kept in the resulting RDF graph. Moreover, the projection of membership and subsumption axioms (the seventh and eighth rows in Table 1) has two settings. In the first setting, the two involved atomic entities are transformed into one triple with the predicate of *rdf:type* or *rdfs:subClassOf*. In the second setting, in addition to the above triple, one more triple which uses the inverse of *rdf:type* or *rdfs:subClassOf* is added. This enables a bidirectional walk between two entities with a subsumption or membership relationship on the transformed RDF graph, and would impact the corpus and the embeddings. In the remainder of this paper, we by default refer to the first setting when we mention projection rules, and we refer to the second setting by the term of projection rules with inverse or by using the suffix "(+R)".

Both ontology to RDF graph transformation strategies can incorporate an OWL entailment reasoner to compute the TBox classification and ABox realization before $\mathcal{O}$ is transformed into an RDF graph $\mathcal{G}$. Such reasoning grounds the axioms of logical constructors and leads to explicit representation of some hidden knowledge. For example, in Fig. 1a, we can infer a hidden triple ⟨*vc:FOOD-4001 (blonde beer)*, *rdf:type*, *vc:AlcoholicBeverages*⟩ from ⟨*vc:FOOD-4001*, *rdf:type*, *vc:Beer*⟩ and ⟨*vc:Beer*, *rdfs:subClassOf*, *vc:AlcoholicBeverages*⟩. When the reasoning is enabled, such inferred hidden triples will be included in the transformed RDF graph $\mathcal{G}$. In our experiments we use the HermiT OWL reasoner (Glimm et al., 2014), and we evaluate the impact of enabling or disabling reasoning (cf. the second paragraph in Sect. 4.3.3 and Table 6).

3.2 Structure document

The structure document aims at capturing both the graph structure and the logical constructors of the ontology. With the RDF graph $\mathcal{G}$, one option is computing random walks for each target entity in E with the RDF graph $\mathcal{G}$. Each walk, which is a sequence of entity IRIs, acts as a sentence of the structure document. Ex1 and Ex2 in Table 2 are two walk examples both starting from the class *vc:FOOD-4001 (blonde beer)*. To implement the random walk algorithm, we first transform the RDF graph $\mathcal{G}$ into a directed single relation graph $\mathcal{G}'$; for each RDF triple ⟨X, r, Y⟩ in $\mathcal{G}$, the subject X, the object Y and the relation r are transformed into three vertices, two edges are added from the vertex of X to the vertex of r and from the vertex of r to the vertex of Y respectively. Given one starting vertex, we fairly and randomly select the next vertex from all its connected vertices, and iterate this "step" operation for a specific number of times to perform "walking".

OWL2Vec* also allows the usage of the Weisfeiler Lehman (WL) kernel (Shervashidze et al., 2011) which encodes the structure of a sub-graph into a unique identity and thus enables the representation and incorporation of the sub-graph in a walk. For one vertex in the transformed single relation graph $\mathcal{G}'$, there is an associated sub-graph (neighbourhood) starting from this vertex, and we simply call this sub-graph's WL kernel (identity) as

Table 2 Sentence examples that are extracted from the ontology fragments in Fig. 1

Document	Example name and source	Sentence(s)
Structure	Ex1: random walk of depth tree starting from *vc:FOOD-4001* (*blonde beer*)	(*vc:FOOD-4001, vc:hasNutrient, vc:VitaminC_100, vc:amountNutrient*)
	Ex2: random walk of depth four starting from *vc:FOOD-4001* (*blonde beer*)	(*vc:FOOD-4001, rdf:type, vc:Beer, rdfs:subClassOf, vc:AlcoholicBeverages*)
	Ex3: the walk in Ex2 with the WL sub-graph kernel enabled	(*vc:FOOD-4001, rdf:type, kernel_id1_md5, rdfs:subClassOf, kernel_id2_md5*)
	Ex4: the existential restriction of *obo:FOODON_00002809* (*edamame*)	(*obo:FOODON_00002809, subClassOf, obo:RO_0001000, some, obo:FOODON_03411347*)
Literal	Ex5: replacing the IRIs in Ex1 by their labels or names	("blonde", "beer", "has", "nutrient", "vitamin", "c", "amount", "nutrient")
	Ex6: replacing the IRIs in Ex3 by their labels or names	("blonde", "beer", "type", kernel_id1_md5, "subclassof", kernel_id2_md5)
	Ex7: the *obo:IAO_0000115* (*definition*) annotation of *obo:FOODON_00002809* (*edamame*)	("edamame", "edamame", "is", "a", "preparation", "of", "immature", "soybean", "in", "their", "pods" …)
Combined	Ex8: keeping the IRI of one item in the Ex1 sentence (random strategy)	(*vc:FOOD-4001*, "has", "nutrient", "vitamin", "c", "amount", "nutrient")
	Ex9: keeping the IRI of one item in the Ex1 sentence (traversing strategy)	(*vc:FOOD-4001*, "has", "nutrient", "vitamin", "c", "amount", "nutrient"), …, ("blonde", "beer", "has", "nutrient", "vitamin", "c", *vc:amountNutrient*)

this vertex's WL kernel. In our implementation, we first extract the original random walks. For each random walk, we then keep the IRIs of the starting vertex and the vertices that are obtained from the relations, but replace the IRIs of the none-starting vertices that are obtained from the subjects or objects with their WL kernels. Ex3 in Table 2 is an example of enabling the WL sub-graph kernel for the walk of Ex2. Note that when calculating a vertex's WL kernel, the size of its sub-graph, *i.e.*, the depth from this vertex to the farthest vertex in the sub-graph can be set. We generate and adopt all the walks, with the sub-graph size ranging from 0 to a maximum size — a hyper-parameter that is set to 4 by default. Specially, the WL kernel enabled random walk with the sub-graph size of 0 is equivalent to the original random walk.

To capture the logical constructors, OWL2Vec* extracts all the axioms of the ontology and complements the sentences of the structure document. In our implementation, each ontology axiom is transformed into a sequence following the OWL Manchester Syntax[7], where the original built-in terms such as "subClassOf" and "some" are kept. Ex4 in Table 2 is an example of such Manchester Syntax sentence according to the axiom of the existential restriction of the class *obo:FOODON_00002809 (edamame)*. In comparison with the random walk over the projected RDF graph, which generates the sentence of (*obo:FOODON_00002809*, *obo:RO_0001000*, *obo:FOODON_03411347*) for the same axiom in Ex4, the Manchester Syntax sentence indicates the logical relationship between the terms by the buit-in terms; while in comparison with the random walk over the graph transformed by W3C OWL to RDF Graph Mapping, the Manchester syntax sentence is shorter and avoids the blank nodes.

3.3 Lexical document

The lexical document includes two kinds of word sentences. The first kind are generated from the entity IRI sentences in the structure document, while the second are extracted from the relevant lexical annotation axioms in the ontology. For the first kind, given an entity IRI sentence, each of its entities is replaced by its English label defined by *rdfs:label*. Note that the label is parsed and transformed into lowercase tokens, and those tokens with no letter characters are filtered out, before it replaces the entity IRI. It is possible that some entities have no English annotations by *rdfs:label*, such as the class *vc:MilkAndYogurt* and the instance *vc:VitaminC_1000* in Fig. 1a. In this case, we prefer to use the name part of the IRI and parse it into words assuming that the name follows the camel case (*e.g.*, *vc:MilkAndYogurt* is parsed into "milk", "and" and "yogurt"). One sentence example of this kind is Ex5 in Table 2, which is generated by replacing the IRIs of the Ex1 sentence by their words. Specially, some IRIs have neither English labels or meaningful IRI names, and when the WL sub-graph kernel is enabled, there are also kernel identities in the structure sentence. We keep these original IRIs and identities in the word sentences (cf. Ex6 in Table 2).

The second kind of word sentences are extracted from the textual annotations. They include two kinds: annotations by bespoke annotation properties such as *obo:IAO_0000115 (definition)*, *obo:IAO_0010000 (has axiom label)* and *oboInOwl:hasSynonym*[6], and annotations by built-in annotation properties such as *rdfs:comment* and *rdfs:seeAlso*. In our current OWL2Vec* implementation, we consider all the annotation properties of an ontology

---

except for *rdfs:label*. The annotations by *rdfs:label* are ignored in generating word sentences of the second kind because they are already considered in the word sentences of the first kind (*e.g.*, Ex5). More specifically, for each annotation axiom, OWL2Vec* replaces the subject entity by its English label or IRI name as in transforming the IRI sentence, and keeps the lowercase word tokens parsed from the annotation value. One example of such word sentence is Ex7 in Table 2 which is based on the annotation by *obo:IAO_0000115 (definition)* to the class *obo:FOODON_00002809 (edamame)*. It would enable the model to learn the correlation of "edamame" to other words in the relevant background such as "soybean" and "pods".

3.4 Combined document

OWL2Vec* further extracts a combined document from the structure document and the entity annotations, so as to preserve the correlation between entities (IRIs) and words in the lexical information. To this end, we developed two strategies to deal with each IRI sentence in the structure document. One strategy is to randomly select an entity in an IRI sentence, keep the IRI of this entity, and replace the other entities of this sentence by their lowercase word tokens extracted from their labels or IRI names as in the creation of the lexical document. One example is Ex8 in Table 2, where the IRI of *vc:FOOD-4001 (blonde beer)* of the IRI sentence of Ex1 is kept while the other IRIs are replaced by their corresponding words. The other strategy is traversing all the entities in a IRI sentence. For each entity, it generates a combined sentence by keeping the IRI of this entity, and replacing the others by their lowercase word tokens as in the random strategy. Thus for one IRI sentence, it generates m combined sentences where m is the number of entities of the IRI sentence. Ex9 in Table 2 is an example of the combined sentences based on the second strategy over the IRI sentence in Ex1.

The combined document aims at capturing the correlation between IRIs and words, such as *vc:FOOD-4001 (blonde beer)* and "nutrient" in Ex7. On the one hand this would benefit the embeddings of the IRIs with the semantics of words. This is especially useful in some contexts where only IRI vectors are available. For example, some entities have neither English labels or meaningful IRI name, and only IRI vectors can be used for these entities. On the other hand, the association with IRIs would incorporate some semantics of the graph structure into the words' embeddings. Again there are some contexts where only words are analyzed. One example is when OWL2Vec* is used as an ontology (domain) tailored word embedding model for the classification of external text of this specific domain. Meanwhile, this may also add noise to the correlation between words (*e.g.*, *vc:hasNutrient* between "beer" and "vitamin" in Ex9) and negatively impact the words' embeddings. The impact of the combined document and its two strategies is analyzed in our evaluation (cf. Sect. 4.3.1).

3.5 Embeddings

OWL2Vec* first merges the structure document, the lexical document and the combined document as one document, and then uses this document to train a *Word2Vec* model with the skip-gram architecture. The training is ended when the loss trends to be stable. The hyper-parameter of the minimum count of words is set to 1 such that each word or entity (IRI) is encoded as long as it appears in the documents at least once. Specially, we can pre-train the *Word2Vec* model by a large and general corpus such as a dump of Wikipedia articles. This brings some prior correlations between words, especially between a word's

synonyms and between a word's variants, which enables the downstream machine learning tasks to identify their semantic equality or similarity w.r.t. the corpus. However, such prior correlations may also be noisy and play a negative role in a domain specific task (cf. the evaluation in Sect. 4.3.4). Note that *Word2Vec* has been selected because it is one of the most widely used word embedding algorithms. It has already been successfully applied in KG embedding in a combination with random walk; one typical example is RDF2Vec (Ristoski and Paulheim 2016; Ristoski et al., 2019). With the adoption of a mature embedding technique, in this study we can focus on extending semantic embedding from a KG to an ontology which expresses a much wider range of semantics, by developing suitable corpus extraction methods. OWL2Vec* is uncoupled to *Word2Vec*, and is thus compatible with other word embedding or sequence feature learning methods such as the contextual model BERT which has shown its superiority according to some recent studies (Miaschi and Dell'Orletta 2020). We leave the selection, evaluation or even development of more suitable language embedding models to our future work.

With the trained word embedding model, OWL2Vec* calculates the embedding of each target entity e in E. Its embedding $\boldsymbol{e}$ is the concatenation of $V_{iri}(e)$ and $V_{word}(e)$, where $V_{iri}(e)$ is the vector of the IRI of e, and $V_{word}(e)$ is some summarization of the vectors of all the lowercase word tokens of e. In our evaluation we simply adopt the averaging operator for $V_{word}(e)$, which usually works quite well for different data and tasks. As predictive information of different words' embeddings lie in different dimensions, the averaging operation would not lead to a loss of predictive information, especially when a classifier is further stacked after the embeddings for downstream applications (cf. Sect. 3.6). Note some more complicated weighting strategies such as using TF-IDF (term frequency–inverse document frequency) (Rajaraman and Ullman, 2011) to calculate the importance of each token can also be considered (cf. Arora et al. (2019) for more methods). As in the case of constructing lexical sentences from IRI sentences, the word tokens of e are extracted from its English label if such a label exists, or from its IRI name otherwise. Due to the concatenation, the embedding size of $\boldsymbol{e}$, *i.e.*, d, is twice the original embedding size. $V_{iri}(e)$ and $V_{word}(e)$ can also be independently used. A comparison of their performance can be found in Sect. 4.3.1.

3.6 Case studies

We applied OWL2Vec* in ontology completion which first trains a prediction model from known relations (axioms) and then predicts those plausible relations.[8] It includes two tasks: class membership prediction and class subsumption prediction, where the embedding of an entity can be understood as the features automatically learned from its neighbourhood, relevant axioms and lexical information without any supervision. In the remainder of this sub-section we first introduce the prediction task details with the membership case and then present the difference with respect to the subsumption case.

Given a head entity e_1 and a tail entity e_2, where e_1 is an instance and e_2 is a class, the membership prediction task aims at training a model to predict the plausibility that e_1 is a member of e_2 (*i.e.*, $e_2(e_1)$). The input is the concatenation of the embeddings of e_1 and e_2, *i.e.*, $\boldsymbol{x} = [\boldsymbol{e_1}, \boldsymbol{e_2}]$, while the output is a score y in [0, 1], where a higher y indicates a more

---

[8] Our ontology completion task is different from ontology reasoning. Our goal is not to infer relations that logically follows from the given input, but to try to discover plausible relations that complement the original ontology. (Most) plausible relations may not be inferred, and our evaluation focuses exactly on those plausible relations that cannot be inferred.

plausible membership relation. For the prediction model, some (non-linear) binary classifiers such as Random Forest (RF) and Multi-Layer Perception (MLP) can be adopted (cf. the evaluation of classifiers in Sect. 4.4).

In training, positive training samples are those declared membership axioms. They are directly extracted from the ontology. While negative samples are constructed by corrupting each positive sample. Namely, for each positive sample (e_1, e_2), one negative sample (e_1, e_2') is generated, where e_2' is a random class of the ontology and e_1 is not a member of e_2' even after entailment reasoning. In prediction, given a head entity (*i.e.*, the target), a candidate set of classes are selected (*e.g.*, all the classes except for the top class *owl:Thing*, or a subset after filtering via some heuristic rules), each candidate is predicted with a normalized score by the trained classifier, which indicates the degree of the candidate to be correct, and the candidates are then ranked according to their scores where the top is the most likely class of the instance. Class subsumption prediction is similar to class membership prediction, except that e_1 and e_2 are both classes, the goal is to predict whether e_1 is subsumed by e_2 (*i.e.*, $e_1 \sqsubseteq e_2$), and the head entity e_1 itself is excluded from the candidate classes.

4 Evaluation

4.1 Experimental setting

We evaluated OWL2Vec* on class membership prediction with the HeLis[9] ontology (Dragoni et al., 2018), and on class subsumption prediction with the FoodOn[10] ontology (Dooley et al., 2018) and the Gene ontology (GO)[11]. HeLis captures general knowledge about both food and healthy lifestyles, FoodOn captures more detailed knowledge about food, and GO is a major bioinformatics initiative to unify the representation of gene and gene product attributes. Their DL expressivities are $\mathcal{ALCHIQ} {\Leftarrow} \mathcal{D} {\Rightarrow}$, $\mathcal{SRIQ}$ and $\mathcal{SRI}$ respectively. Some statistics of the two ontologies are shown in Table 3. Due to different knowledge representations, HeLis has a large number of membership axioms but a very small number of subsumption axioms, while FoodOn and GO have only subsumptions axioms. This is the reason why we evaluated membership prediction on HeLis, but subsumption prediction on FoodOn and Go. Data and codes are available at https://github.com/KRR-Oxford/OWL2Vec*-Star.

The experiment on membership and subsumption prediction follows the following setting: all the explicitly declared class membership axioms (or class subsumption axioms) are randomly divided into three sets for training (70%), validation (10%) and testing (20%), respectively. For each axiom in the validation/testing set, the head entity (*i.e.*, an instance for membership prediction and a class for subsumption prediction) is the target whose class is to be predicted from all the candidates and compared against the tail entity (as the ground truth class) in evaluation. All the candidates are ranked according to the predicted score which indicates the likelihood of being the head entity's class. We calculate the following widely adopted metrics: Hits@1, Hits@5, Hits@10 and MRR (Mean Reciprocal

---

[9]　HeLis project: https://horus-ai.fbk.eu/helis/

[10]　FoodOn project: https://foodon.org/

[11]　GO was accessed on August 05, 2020 via http://www.geneontology.org/ontol ogy/

Table 3 Statistics of the HeLis ontology, the FoodOn ontology and the GO ontology

	Instances#	Classes #	Axioms #	Memb. #	Subs. #	Label Avg Size	Entities with No Label	Anno. Avg Size	Anno. # per Entity
HeLis	20, 318	277	172, 213	20, 318	261	1.71	0.05%	0.13	0.14
FoodOn	359	28, 182	241, 581	0	29, 778	2.92	0.32%	6.70	3.99
GO	0	44, 244	513, 306	0	72, 601	4.26	0.15%	4.26	9.09

Memb. and Subs. are short for membership axioms and subsumption axioms, respectively. Membership and subsumption in Table 3 denote the declared membership and subsumption axioms with named classes alone, *i.e.*, those involving composed classes and those inferred are not counted. # denotes the count. Size refers to the number of word tokens. Label refers to the English label by *rdfs:label* of the instance or the class, or the meaningful IRI name if there is no such an English label. Anno. refers to the literals by annotation properties (excluding *rdfs:label*). Entity here refers to instance and class

Rank). The first three measure the recall of the ground truths within the top 1, 5 and 10 ranking positions, respectively, while the fourth averages the reciprocals of the ranking positions of the ground truths. The higher the metrics, the better the performance.

The performance of OWL2Vec⋆ is reported with the following settings. For the embedding model, its dimension is set to 100 if no pre-training is adopted, and otherwise set to be consistent with the pre-trained model; the window size is set to 5; the minimum count of words is set to 1; the iteration number of training is set to 10, which is based on the observation of the loss. The *Word2Vec* pre-training (with a dimension of 200) uses the latest English Wikipedia article dump[12], as in many other *Word2Vec* relevant studies such as (Chen et al., 2019). Other corpus or pre-trained models can also be used, and we can further select a corpus that is specific to the domain of the ontology. Such extensive evaluation will be considered in the future work. Random Forest (RF) is adopted as the basic binary classifier and the WL sub-graph kernel is enabled by default in the random walk unless stated otherwise. Other hyper-parameters such as the walking depth and the transformation from OWL ontology to RDF graph, as well as the hyper-parameters of the baselines are adjusted through the validation set — the setting that leads to the highest MRR is adopted.

The evaluation is organized as follows. We first compare OWL2Vec⋆ with the baselines, then analyze the impact of different settings including the documents, the IRI and word embeddings, the settings for generating the structure document (walking type, walking depth and the transformation from OWL ontology to RDF graph), the usage of reasoning and pre-training. We next analyze the effectiveness of OWL2Vec⋆ towards different classifiers including RF, MLP, Logistic Regression (LR) and Support Vector Classifier (SVC), all of which are implemented by scikit-learn (Pedregosa et al., 2011), and finally analyze the embeddings via visualization and comparing the Euclidean distances. The selected embedding baselines include *(i)* four well-known knowledge graph embedding methods, *i.e.*, RDF2Vec, TransE, TransR and DistMult, *(ii)* four state-of-the-art ontology embedding methods, *i.e.*, Onto2Vec, OPA2Vec, EL Embedding and Quantum Embedding,[13] *(iii)* the original OWL2Vec* which is equivalent to OWL2Vec⋆ using the IRI embedding, structure document and ontology projection rules, *and (iv)* the pre-trained *Word2Vec* model. The embeddings of these baselines are applied to the two tasks in the same way as OWL2Vec⋆, with the Random Forest classifier. Note that RDF2Vec, TransE, TransR and DistMult are trained with the RDF graph $\mathcal{G}$ transformed from the original ontology using OWL to RDF Graph Mapping without entailment reasoning, while the pre-trained *Word2Vec* calculates the average word vector of an entity according to its label (or its IRI name if the label does not exist) as in OWL2Vec⋆.

The surface form of the textual information (*i.e.*, naming conventions followed in the ontologies) may also play a role in our prediction tasks. For example, the instance "Blonde Beer" includes the word "Beer", which is the label of its membership class. Thus we further compared our OWL2Vec⋆ plus RF solution to the supervised Transformer classifier (Vaswani et al., 2017) which embeds the head and tail entities' contextual text. The Transformer classifier has two versions: *label* which considers the English labels and IRI names of the two entities, and *all text* which considers all of the two entities' textual labels and annotations. The label, IRI name and textual annotation are pre-processed in the same way as in OWL2Vec⋆, and they are orderly concatenated into one sequence as the input of the Transformer.

---

[12] https://dumps.wikimedia.org/enwiki/

[13] For EL (resp. Quantum) Embedding, HeLis and FoodOn are first transformed into DL $\mathcal{EL}^{++}$ (resp. DL $\mathcal{ALC}$) by removing logical axioms outside the supported expressivity.

Table 4 Overall results of OWL2Vec* and the baselines

Method	HeLis			
	MRR	Hits@1	Hits@5	Hits@10
Transformer (*label*)	0.657	0.515	0.824	0.897
Transformer (*all text*)	0.599	0.390	0.870	0.912
RDF2Vec	0.345	0.219	0.460	0.655
TransE	0.181	0.09	0.232	0.355
TransR	0.298	0.184	0.391	0.559
DistMult	0.253	0.166	0.304	0.437
Quantum Embeding	0.159	0.132	0.163	0.190
Onto2Vec	0.211	0.108	0.268	0.397
OPA2Vec	0.237	0.146	0.286	0.408
OWL2Vec*	0.335	0.215	0.397	0.601
Pre-trained *Word2Vec*	0.899	0.877	0.923	0.933
OWL2Vec*	**0.953**	**0.932**	**0.978**	**0.987**

(a) Membership Prediction

Method	FoodOn				GO			
	MRR	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10
Transformer (*label*)	0.016	0.005	0.027	0.046	0.009	0.001	0.009	0.018
Transformer (*all text*)	0.022	0.011	0.032	0.050	0.014	0.008	0.017	0.019
RDF2Vec	0.078	0.053	0.097	0.119	0.043	0.017	0.057	0.087
TransE	0.029	0.011	0.044	0.065	0.015	0.005	0.018	0.030
TransR	0.072	0.044	0.093	0.130	0.048	0.016	0.076	0.113
DistMult	0.076	0.045	0.099	0.134	0.046	0.018	0.68	0.097

Table 4 (continued)

Method	FoodOn				GO			
	MRR	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10
EL Embeding	0.040	0.014	0.067	0.099	0.018	0.005	0.021	0.036
Onto2Vec	0.034	0.014	0.047	0.064	0.024	0.008	0.031	0.053
OPA2Vec	0.093	0.058	0.117	0.156	0.075	0.032	0.106	0.157
OWL2Vec*	0.091	0.052	0.121	0.152	0.031	0.012	0.040	0.067
Pre-trained *Word-2Vec*	0.136	0.089	0.175	0.227	0.123	0.055	0.177	0.260
OWL2Vec*	**0.213**	**0.143**	**0.287**	**0.357**	**0.170**	**0.076**	**0.258**	**0.376**

(b) Subsumption Prediction

For RDF2Vec we use the implementation of pyRDF2Vec[14]; for TransE, TransR and DistMult we use the implementation of OpenKE[15]; for Onto2Vec and OPA2Vec we implement them as special cases of OWL2Vec⋆; for EL Embedding and Quantum Embedding we use the codes attached in their original papers (Holter et al., 2019) and (Garg et al., 2019), respectively. The Transformer classifier is implemented by Tensorflow[16] with one token and position embedding layer, and one Transformer block that contains two attention heads. All the results are generated locally with repetitions.

4.2 Comparison with baselines

Table 4 reports the performance of OWL2Vec⋆ and the baselines with their optimum settings. The performance of OWL2Vec⋆ with different settings can be found in Sect. 4.3. In Table 4 we can observe that OWL2Vec⋆ outperforms all the baselines. Note all these comparisons have statistical significance with the p-value being $\ll 0.05$ in the two-tailed test. Among all these ontology embedding and KG embedding baselines which directly calculate the IRI's vector without considering the word vector, OPA2Vec achieves the best performance on FoodOn and GO for subsumption prediction; while the KG embedding method RDF2Vec performs the best on HeLis for class membership prediction. In contrast, the two logic embedding methods Quantum Embedding and EL Embedding, and TransE perform poorly on all the three ontologies. Our preliminary work OWL2Vec* achieves promising results on HeLis (close to RDF2Vec) and FoodOn (close to OPA2Vec), but performs poorly on GO. OWL2Vec⋆ outperforms both KG embedding methods and ontology embedding methods; for example, consider the Hits@1 of OWL2Vec⋆, it is 325.6% higher than RDF2Vec on HeLis, 146.6% higher than OPA2Vec on FoodOn, and 126.7% higher than OPA2Vec on GO.

Meanwhile, OWL2Vec⋆ outperforms the pre-trained *Word2Vec*, with 6.0%, 56.6% and 38.2% higher MRR on HeLis, FoodOn and GO, respectively. It is interesting to see that the pre-trained *Word2Vec* using entity labels or IRI names achieves good performance, outperforming ontology and KG embedding baselines such as RDF2Vec and OPA2Vec. This means that the textual information plays a very important role in embedding real world ontologies, especially for membership prediction and subsumption prediction as the names of the instances and classes with a membership or subsumption relationship often use some common words or words with relevant meanings (*e.g.*, synonyms or word variants). This observation on the importance of the textual information is consistent with our following ablation study on the usage of word embedding, IRI embedding and both (cf. V_{iri}, V_{word} and $V_{iri,word}$ in Table 5). A key difference between OWL2Vec⋆ and *Word2Vec* is that the word embedding model of OWL2Vec⋆ is trained by an ontology tailored corpus underpinned by its graph structure and logical axioms.

In Table 4, we can also observe that both Transformer classifiers (*i.e.*, *label* and *all text*) perform worse than the RF classifier using the pre-trained *Word2Vec* or OWL2Vec⋆ embeddings. On HeLis, they are effective with some promising results which are better than the KG and ontology embedding baselines; while on FoodOn and GO, they are very ineffective, with much worse performance than all the other methods. This means that the

---

[14] https://github.com/IBCNServices/pyRDF2Vec
[15] https://github.com/thunlp/OpenKE
[16] https://www.tensorflow.org/

Table 5 The results of OWL2Vec* under different document (D) and embedding (V) settings. Subscripts: s (resp. l) denotes the structure (resp. lexical) document; rc (resp. tc) denotes the combined document with the random (resp. traversal) strategy; iri (resp. $word$) denotes the IRI (resp. word) embedding

HeLis

Setting	MRR	Hits@1	Hits@5	Hits@10
$D_s + V_{iri}$	0.353	0.226	0.470	0.668
$D_{s,l} + V_{iri}$	0.448	0.295	0.623	0.814
$D_{s,l} + V_{word}$	0.938	0.920	0.961	0.974
$D_{s,l} + V_{iri,word}$	0.952	**0.934**	0.974	0.984
$D_{s,l,rc} + V_{iri}$	0.446	0.299	0.618	0.799
$D_{s,l,rc} + V_{word}$	0.945	0.926	0.970	0.979
$D_{s,l,rc} + V_{iri,word}$	0.951	0.932	**0.975**	**0.987**
$D_{s,l,tc} + V_{iri}$	0.505	0.355	0.695	0.854
$D_{s,l,tc} + V_{word}$	0.943	0.923	0.969	0.976
$D_{s,l,tc} + V_{iri,word}$	**0.953**	0.932	**0.975**	**0.987**

(a) Membership Prediction

Setting	FoodOn				GO			
	MRR	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10
$D_s + V_{iri}$	0.154	0.102	0.210	0.150	0.095	0.044	0.144	0.195
$D_{s,l} + V_{iri}$	0.183	0.120	0.249	0.305	0.116	0.048	0.185	0.252
$D_{s,l} + V_{word}$	**0.213**	**0.143**	**0.287**	**0.357**	**0.170**	**0.076**	**0.258**	**0.376**
$D_{s,l} + V_{iri,word}$	0.203	0.133	0.273	0.338	0.137	0.068	0.204	0.319
$D_{s,l,rc} + V_{iri}$	0.188	0.125	0.249	0.310	0.115	0.050	0.164	0.249
$D_{s,l,rc} + V_{word}$	0.196	0.127	0.246	0.329	0.155	0.066	0.237	0.348
$D_{s,l,rc} + V_{iri,word}$	0.201	0.134	0.270	0.330	0.139	0.060	0.206	0.297
$D_{s,l,tc} + V_{iri}$	0.172	0.107	0.234	0.297	0.101	0.046	0.154	0.209
$D_{s,l,tc} + V_{word}$	0.194	0.130	0.254	0.314	0.150	0.061	0.230	0.343

Table 5 (continued)

Setting	FoodOn				GO			
	MRR	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10
$D_{s,I,c} + V_{iri,word}$	0.202	0.127	0.278	0.349	0.139	0.061	0.204	0.300

(b) Subsumption Prediction

surface form of the entities' textual contexts (token sequences), with feature learning by Transformer, brings very little predictive information on FoodOn and GO and partial predictive information on HeLis. This result is consistent with our observation on the ontologies' naming mechanisms for the instances and subclasses, and in turn confirms that the semantics from the large Wikipedia corpus encoded in the *Word2Vec* embeddings, and the semantics of the ontology graph and logical constructors encoded in the OWL2Vec* embeddings play a very important role in these prediction tasks.

Note that the performance of membership prediction with HeLis is much higher than that of the subsumption prediction with FoodOn and Go. This is because the former has much less candidate classes (cf. Table 3) and is thus less challenging. Meanwhile the entity name and label's surface form (*i.e.*, the class member's naming mechanism) of HeLis makes additional contribution to its membership prediction, as analyzed above.

4.3 Analysis of OWL2Vec* settings

4.3.1 Lexical information

According to Table 5 we can find that the lexical document D_l leads to a significant improvement of performance when it is merged with the structure document D_s (*i.e.*, $D_{s,l}$). Consider MRR: $D_{s,l}$ outperforms D_s by 26.9% on HeLis, by 18.8% on FoodOn and by 22.1% on GO when the IRI embedding (V_{iri}) is used, and by 169.7%, 31.8% and 44.2% respectively when both IRI embedding and word embedding ($V_{iri,word}$) are used.

Unlike the lexical document, the combined documents ($D_{s,l,rc}$ and $D_{s,l,tc}$), which also rely on the lexical information of the ontology, lead to a limited positive impact. For class membership prediction, the best performance of $D_{s,l,rc}$ (MRR: 0.951) and the best performance of $D_{s,l,tc}$ (MRR: 0.953) are both very close to the best performance as $D_{s,l}$ (MRR: 0.952), while for class subsumption prediction on FoodOn and GO, they are both worse than the best performance of $D_{s,l}$. We can also find that the combined document has a negative impact when the word embedding (V_{word}) is used alone on FoodOn and GO. On FoodOn, D_{rc} (D_{tc} resp.) reduces the MRR from 0.213 to 0.196 (0.194 resp.), while on GO, D_{rc} (D_{tc} resp.) reduces the MRR from 0.170 to 0.155 (0.150 resp.). This is because the combined sentences build the correlation between words and IRIs, which benefits the IRI embeddings, but brings noise to the correlation between words and harms the word embeddings.

Besides the lexical document, the word embedding (V_{word}) which also benefits from the utilization of the lexical information of the ontology shows a very strong positive impact. On the one hand, as discussed in Sect. 4.2, the two methods that use the word embedding, *i.e.*, OWL2Vec* and the pre-trained *Word2Vec*, both dramatically outperform the remaining methods. On the other hand, as shown in Table 5, the best performance on HeLis comes from $V_{iri,word}$, while the best performance on FoodOn and GO comes from V_{word}. The improvement of $V_{iri,word}$ and V_{word} over V_{iri} is quite significant; for example, when the lexical and structure documents ($D_{s,l}$) are used, the Hits@1 of $V_{iri,word}$ is 0.934, 0.133 and 0.068 on HeLis, FoodOn and GO, respectively, while the corresponding Hits@1 of V_{iri} is 0.295, 0.120 and 0.048, respectively.

Regarding the IRI embedding, on the one hand it can alone outperform the baseline embeddings in Table 4 except for the pre-trained *Word2Vec*. On the other hand, the impact of the IRI embedding when it is concatenated with the word embedding varies from task to task. It has a positive impact on class membership prediction with HeLis; for example,

(a) Membership Prediction (HeLis)

(b) Subsumption Prediction (FoodOn)

(c) Subsumption Prediction (GO)
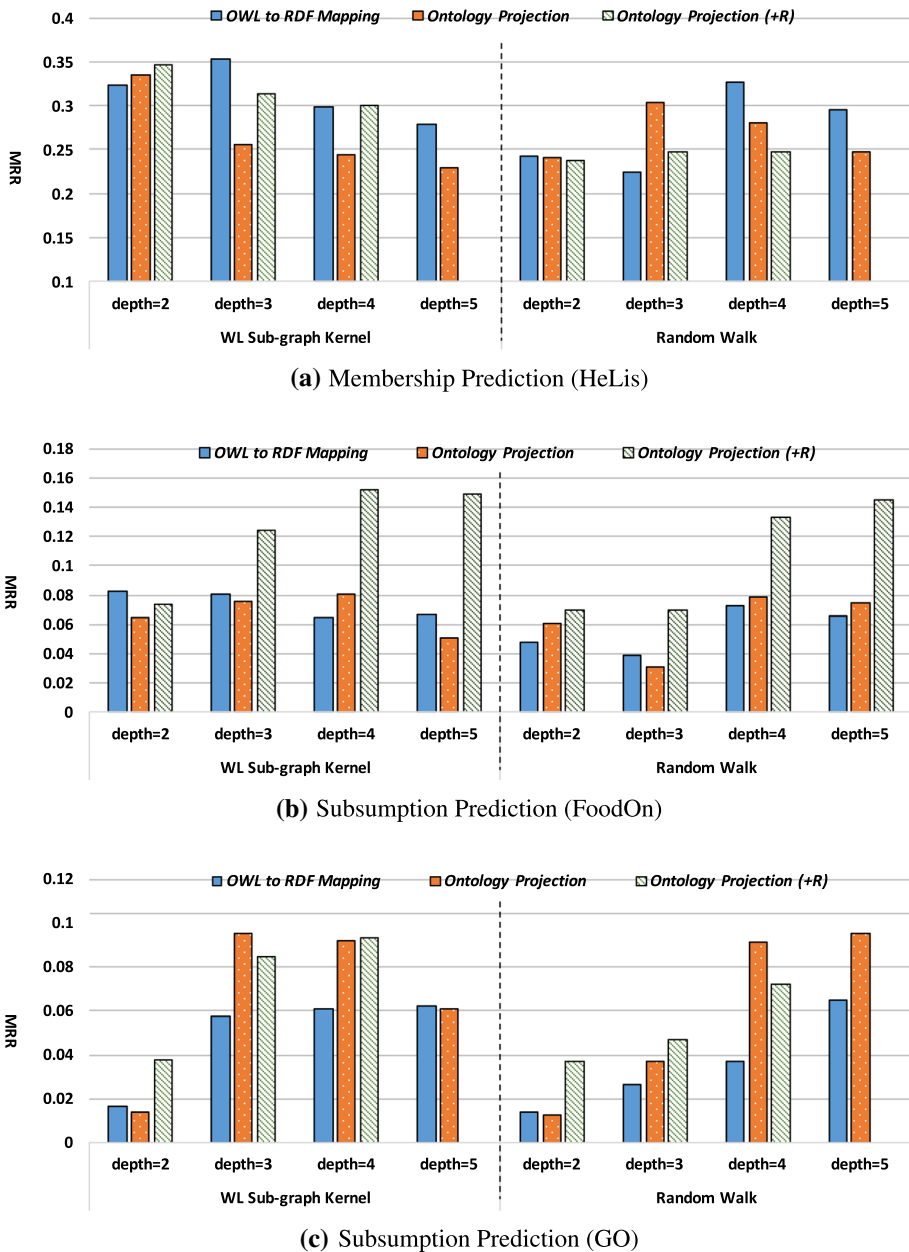
Fig. 3 Comparison of structure documents by different graph structure exploration settings, with the MRR results of OWL2Vec* ($D_s + V_{iri}$) reported

when trained by the structure document and lexical document ($D_{s,l}$), the MRR of $V_{iri,word}$ is 1.5% higher than V_{word}. However, on class subsumption prediction with FoodOn and GO, the IRI embedding shows a negative impact, *i.e.*, $V_{iri,word}$ is often close to or worse than V_{word}. This may be due to the fact that the lexical semantics plays a dominant role in these

prediction tasks, and the word sentences from which the word embeddings are learned have already used the structure sentences as the backbone. Meanwhile, we simply concatenate the two vectors as the input of a basic classifier without any mechanisms to better integrate the two embeddings (inputs).

4.3.2 Graph structure

Figure 3 shows the performance of the IRI embedding of OWL2Vec* when it is trained using structure documents extracted under different ontology graph structure exploration settings. We first compare the two solutions that generate the RDF graph $\mathcal{G}$: *(i)* the OWL to RDF Graph Mapping defined by W3C, which may lead to redundant blank nodes and longer paths for some complex axioms, but keeps the complete semantics; *(ii)* the ontology projection rules which lead to a more compact graph but approximate most complex axioms, *i.e.*, some logical relationships like the type of class restrictions are missing in the projected RDF graph; and *(iii)* the ontology projection rules with inverse triples for the membership and subsumption axioms. Please see Sect. 3.1 for more details. Note that some results of the solution *(iii)* with the depth of 5 are missing as the walking strategy did not stop after several hours. On HeLis, the solution *(i)* has higher MRR when the WL sub-graph kernel is enabled and the depth is set to ≥ 3, or when the original random walk is adopted and the depth is set to ≥ 4. Its best MRR value (*i.e.*, 0.353) is higher than the other solutions (*i.e.*, 0.335 and 0.346). On FoodOn, the solution *(iii)* has much higher MRR when the walking depth is ≥ 4 (as high as 0.152 with the WL sub-graph kernel enabled) than the best of the solution *(i)* and *(ii)* (0.083 and 0.081 respectively). On GO, the solution *(ii)* performs well when the walking depth is set to 3 or 4, and the WL sub-graph kernel is enabled, or when the walking depth is ≥ 4 with pure random walk. It also leads to the best MRR, *i.e.*, 0.095. Therefore, the performance of these three ontology to RDF graph transformation methods varies from ontology to ontology; in OWL2Vec*, the OWL to RDF Graph Mapping is adopted on HeLis, the projection rules with inverse are adopted on FoodOn and the projection rules are adopted on GO.

With Figure 3 we can also compare different walking strategies and walking depths used in extracting IRI sentences from the RDF graph $\mathcal{G}$. We have two main observations. First, the walking depth is important for random walk with or without the WL sub-graph kernel. In general, to achieve the best performance, random walk with WL sub-graph kernel needs a smaller walking depth. Consider the OWL to RDF Mapping: the optimal walking depth is 3 on HeLis and 2 on FoodOn for random walk with the WL sub-graph kernel, but is 4 for raw random walk. Consider the ontology GO with the projection rules: the best performance for random walk with the WL sub-graph kernel often lies in the depth of 3, while the best performance for raw random walk lies in the depth of 5. Second, the top MRR with the WL sub-graph kernel enabled is higher than the top MRR with raw random walk on HeLis and FoodOn, and is the same on GO. Both observations are as expected because enabling the WL sub-graph kernel incorporates the structure information of the sub-graphs of partial entities of a random walk.

4.3.3 Logical constructors

On the one hand, the performance of the baselines in Table 4 which adopt the logical structure alone, including EL Embedding, Quantum Embedding and Onto2Vec, is relatively poor in comparison with the other methods. On the other hand, the logical structure has a

Table 6 Performance (MRR) *with* and *without* entailment reasoning

	Setting	Onto2Vec	OPA2Vec	OWL2Vec* $(D_s + V_{iri})$	OWL2Vec* $(D_{s,l} + V_{word})$
HeLis	*with*	0.211	0.237	0.340	0.938
	without	0.221	0.226	0.353	0.935
FoodOn	*with*	0.034	0.087	0.152	0.188
	without	0.019	0.088	0.154	0.205
GO	*with*	0.024	0.075	0.092	0.167
	without	0.034	0.087	0.095	0.170

Regarding OWL2Vec*, the ontology is transformed into an RDF graph with the approach that achieves the best performance, *i.e.*, W3C OWL to RDF Graph Mapping on HeLis, projection rules (+R) for FoodOn, and projection rules for GO

Table 7 Performance of the classifiers of Random Forest (RF), Multi-Layer Perception (MLP), Support Vector Classifier (SVC) and Logistic Regression (LR), using OWL2Vec* $(D_{s,l} + V_{word})$

	RF		MLP		SVC		LR	
	MRR	Hits@1	MRR	Hits@1	MRR	Hits@1	MRR	Hits@1
HeLis	0.938	0.920	0.935	0.916	0.816	0.752	0.135	0.052
FoodOn	0.213	0.143	0.219	0.120	0.063	0.033	0.012	0.009
GO	0.170	0.076	0.152	0.064	0.138	0.071	0.011	0.008

positive impact when it works together with the graph structure in OWL2Vec*. On HeLis, in comparison with RDF2Vec, the difference of OWL2Vec* with the setting of the structure document and the IRI embedding (*i.e.*, $D_s + V_{iri}$) is that it additionally uses sentences from Manchester Syntax axioms; while on FoodOn and GO, the difference also includes that OWL2Vec* $(D_s + V_{iri})$ uses the projection rules (with inverse) while RDF2Vec uses OWL to RDF Graph Mapping. By comparing the results of RDF2Vec in Table 4 and the results of OWL2Vec* $(D_s + V_{iri})$ in Table 5, we can find the impact of adding these axiom sentences is positive on HeLis; the latter (*i.e.*, using axiom sentences) has 2.3% higher MRR and 3.2% higher Hits@1. On FoodOn and GO, the improvement of OWL2Vec* $(D_s + V_{iri})$ over RDF2Vec is very significant, with 97.4% higher MRR and 92.5% higher Hits@1 on FoodOn, and 120.9% higher MRR and 164.7% higher Hits@1 on GO. This improvement is partially due to the usage of the Manchester Syntax axioms, and is partially due to the projection rules (with inverse).

We also analyzed the impact of using reasoning (provided by the OWL 2 reasoner HermiT) before the ontology is transformed into an RDF graph. The results on Onto2Vec, OPA2Vec, OWL2Vec* $(D_s + V_{iri})$ and OWL2Vec* $(D_{s,l} + V_{word})$ are shown in Table 6. We can see that reasoning has a limited impact in the conducted experiments; the MRR results *with* and *without* reasoning are quite close for OPA2Vec and OWL2Vec* with both settings of $D_s + V_{iri}$ and $D_{s,l} + V_{word}$. Note that OWL2Vec* in Table 6 uses W3C OWL to RDF Graph Mapping on HeLis, projection rules with inverse on FoodOn and projection rules on GO. The impact of reasoning is limited with all the three transformation approaches. The impact of reasoning for Onto2Vec is more significant especially on
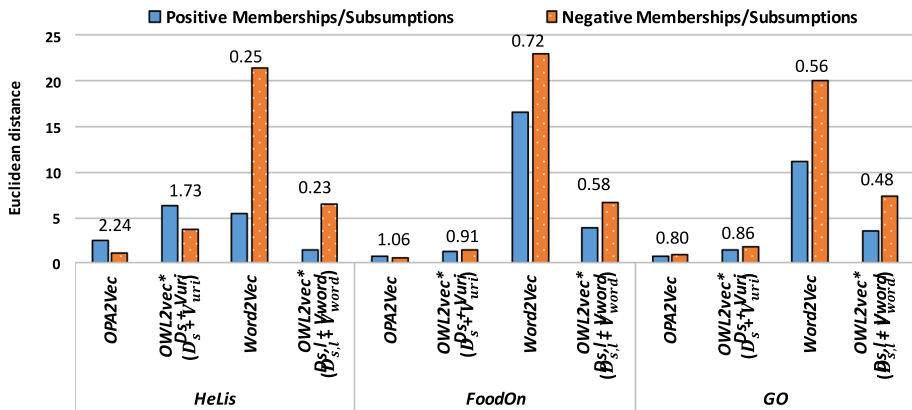
Fig. 4 The average Euclidean distance between the class and its instance (resp. subclass) for the positive and negative memberships (resp. subsumptions) used in classifier training. The number above every pair of positive and negative bars is their ratio

FoodOn and GO. That may be because it uses the axiom sentences that are more likely to be impacted by entailment reasoning. OPA2Vec uses the sentences of literal annotations which are much less impacted by entailment reasoning. OWL2Vec* uses multiple kinds of sentences and are thus more robust. Meanwhile, the impact of reasoning on Onto2Vec varies from ontology to ontology; for example, it is positive for FoodOn but negative for GO.

4.3.4 Pre-training

With the setting of $D_{s,l} + V_{word}$, the MRR of OWL2Vec* decreases to 0.933, 0.136 and 0.147 on HeLis, FoodOn and GO respectively, while its Hits@1 decreases to 0.913, 0.091 and 0.069 respectively. On the one hand, using a pre-trained *Word2Vec* does not increase but decreases the performance of OWL2Vec* (cf. Table 5 for the corresponding results without pre-training). That may be because the pre-trained *Word2Vec* is short of prior correlations involving entity IRIs, and its usage also leads to less compact embeddings with their dimension increases from 100 to 200. On the other hand, OWL2Vec* with pre-training still outperforms the original *Word2Vec* whose results are shown in Table 4. This in turn verifies that the embeddings learned from the generated documents underpinned by the graph structure and the logical structure are tailored to the specific characteristics of the given ontology and these embeddings are more effective for the prediction tasks of these ontologies.

4.4 Classifiers

Table 7 presents the results of four different binary classifiers that use the OWL2Vec* embeddings as input. Note that we used the setting of $D_{s,l} + V_{word}$ because it achieves the best performance on subsumption prediction for FoodOn and GO, and very competitive performance on membership prediction for HeLis in our above evaluation. We can find that MLP (with single hidden layer) has a quite competitive performance as RF, especially on HeLis and FoodOn. The performance of OWL2Vec* with MLP is also better than all the baselines in Table 4 on each ontology. SVC can work for HeLis and GO, but it has lower
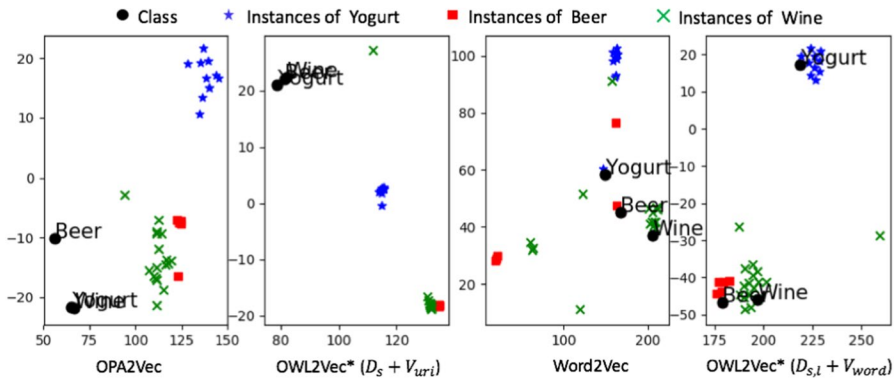
(a) Three classes of HeLis – "Yogurt", "Beer" and "Wine", and their instances (10, 6 and 17 respectively).
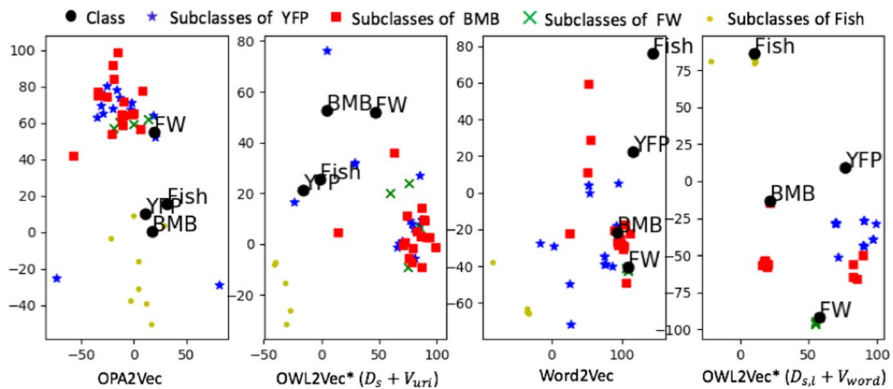
(b) Four classes of FoodOn – "Yogurt Food Product" (YFP), "Barley Malt Beverage" (BMB), "Fruit Wine" (FW) and "Fish", and their subclasses (14, 17, 5 and 8 respectively).

Fig. 5 Embedding visualization via t-SNE

MRR and Hits@1 than RF and MLP. On the other hand, we find LR, the only linear classifier, has very poor performance on all the three ontologies. This indicates that the embeddings learned by OWL2Vec* need a non-linear classifiers to achieve good performance in membership prediction and subsumption prediction.

4.5 Interpretation and visualization

To show that the learned embeddings (*i.e.*, input features of the classifier for membership and subsumption prediction) are discriminative and effective, we analyze the Euclidean distance between the embeddings of the two entities in a membership or subsumption axiom. We calculate the average distance for the true axioms that are extracted from the ontology, and the false axioms that are constructed by corrupting each true axiom (*i.e.*, the same way as negative sampling in the case studies), using the embeddings learned by OPA2Vec, the pre-trained *Word2Vec*, and OWL2Vec* with two settings. The results are shown in Fig. 4. Note that the difference of the Euclidean distance between the entities in the positive axioms and the entities in the negative axioms is sufficient to indicate the discrimination of

the features, but it is not necessary. We can find that *Word2Vec* and OWL2Vec⋆ with $D_{s,l}$ + V_{word} (*i.e.*, using the structure document, the lexical document and the word embedding) have quite discriminative average distances for all the three ontologies. Namely, the positive axioms lead to much shorter average distance than the negative axioms. This is consistent with their final good performance shown above. Specially, for OPA2Vec and OWL2Vec⋆ with D_s + V_{iri} (*i.e.*, using the structure document and the IRI embedding) on HeLis, we can find the distance is also discriminative. However, in contrast, the positive axioms has longer average distance than the negative axioms. This is because the instance usually lies in one end of a sequence where it co-occurs with its class (*i.e.*, a walk of WL sub-tree kernel of depth 3 for OWL2Vec⋆, or a membership axiom for OPA2Vec), and thus its distance of co-occurrence to its class becomes larger than to a random class.

We also visualize the embeddings of some example classes/instances via t-SNE (Maaten and Hinton 2008) in order to obtain further insights about the quality of the computed embeddings. In Fig. 5a (for HeLis) we can find two characteristics for the embeddings learned by OWL2Vec⋆ with $D_{s,l}$ and V_{word}: (1) the instances of each class are clustered into a compact cluster, and (2) these instances are very close to their corresponding class. Both characteristics are promising: they confirm that the embeddings are discriminative and explain why the embeddings enable a very good performance in membership prediction (*e.g.*, Hits@5 is as high as 0.978). For the embeddings learned by OPA2Vec and OWL2Vec⋆ with D_s and V_{iri}, they have the first characteristic as well, but the distance of an instance to its class is often longer than its distance to some other class, which is consistent with the average Euclidean distance analyzed above. Such embeddings can still benefit membership prediction under the standard supervised learning setting adopted in our evaluation, where some instances of one class are used for training while the other instances of this class, which are close to the training instances in the embedding space, are for testing. However, the generalization will be dramatically impacted, especially under a zero-shot learning setting where the instances of a new class, which have never appeared in the training samples, are used for testing.

In Fig. 5b (for FoodOn) we can observe similar characteristics for the embeddings learned by OWL2Vec⋆ with $D_{s,l}$ and V_{word}. Namely, for each class, its subclasses are mostly quite close to each other (*i.e.*, being clustered into one cluster), and their distances to this class are mostly shorter than their distance to any other class. However, the two characteristics are not as significant as in HeLis, especially for the class "Barley Malt Beverage" and its subclasses, indicating that embedding FoodOn, which has more axioms and entities (cf. Table 3), is more challenging. On the other hand, the two characteristics of OWL2Vec⋆ with $D_{s,l}$ and V_{word} are more significant than those of the other three methods — *Word2Vec*, OPA2Vec and OWL2Vec⋆ with D_s and V_{iri}, which demonstrates its better performance on subsumption prediction. For example, in comparison with *Word2Vec* which has the second best performance, OWL2Vec⋆ with $D_{s,l}$ and V_{word} shortens the distance between "Fish" and its subclasses, and makes the subclasses of "Yogurt Food Product" closer to each other.

5 Discussion and outlook

In this paper we have presented OWL2Vec⋆, a robust semantic embedding framework for OWL ontologies. OWL2Vec⋆ extracts documents from the ontology that capture its graph structure, axioms of logical constructors, as well as its lexical information, and then learns a word embedding model for both entity embeddings and word embeddings. We applied

OWL2Vec* to class membership prediction and class subsumption prediction with three real world ontologies, namely HeLis, FoodOn and GO, and we empirically analysed different semantics and techniques such as entailment reasoning and ontology to RDF graph transformation. The evaluation demonstrates that on these tasks OWL2Vec* can significantly outperform state-of-the-art methods.

Ontology Text Understanding. Our experiments suggest that lexical information plays a very important role in both class membership prediction and class subsumption prediction. In real world ontologies such as HeLis, FoodOn and GO, entity names often reflect, in natural language, their relationships to surrounding entities; in HeLis, for example, the instance *vc:FOOD-700637* (*Soy Milk*) is an instance of the class *vc:SoyProducts*. In addition, ontologies often contain a large number of entity annotations ranging from short phrases to long textual descriptions. In FoodOn, for example, 169, 630 out of 241, 581 axioms are annotations. However, patterns within the textual information in the ontologies, which is underpinned by the graph and logical structure, are quite different from normal natural language text (cf. Sect. 4.3.4). To further improve ontology embedding in the future, we need to develop new word embedding architectures and training methods that are tailored to the kinds of textual information typically present in state-of-the-art ontologies.

Ontology Completion via Prediction. In this study OWL2Vec* has been used to complete an ontology by discovering plausible axioms. We adopted a typical supervised learning setting to model a common scenario in ontology completion, where satisfactory results have been achieved; in class membership prediction, the classes of 93.2% of the test instances can be recalled. In some real world cases, however, there is often a bias between the axioms for training and the axioms for prediction. For example, consider the case of membership prediction for a new class defined on the fly without any known instances (*i.e.*, zero-shot learning scenario discussed Sect. 4.5). This leads to sample shortage in training and becomes much more challenging—the above metric drops to 65.6% for OWL2Vec* and less than 10% for other KG embedding and ontology embedding methods in Table 4. In future work we plan to develop more robust ontology embeddings with higher generalization for dealing with such cases. Meanwhile, we will consider using OWL2Vec* embeddings and machine learning to address other ontology completion challenges. One study we are working on is predicting the cross-ontology class mapping for ontology integration and curation (Chen et al., 2020b; Horrocks et al., 2020); another potentially meaningful study is approximating the output of the standard reasoning tasks over expressive ontologies, which have exponential or even higher time complexity by traditional logical reasoners.

Applications. In addition to the evaluated membership and subsumption prediction tasks, OWL2Vec* can be applied to assist a wide range of ontology design and quality assurance (QA) problems (Horrocks et al., 2020). A typical QA task is ontology alignment as presented in our ongoing work (Chen et al., 2020b), where we use OWL2Vec* to embed the classes of two to-be-aligned ontologies as their features for mapping prediction. Note through the ontology mappings, we can further use the cross-ontology information to augment subsumption and membership prediction; for example, the missing subsumption relationship between *obo:FOODON_03305289 (Soybean Milk)* and *obo:FOODON_00002266 (Soybean Food Product)* in FoodOn (where *obo:FOODON_03305289* is only categorized as *obo:FOODON_00003202 (Beverage)*) can be discovered by mapping them to their HeLiS counterparts *vc:SoyMilk* and *vc:SoyProducts* whose subsumption relationship is defined. Both OWL2Vec* and (Chen et al., 2020b) are in cooperation with Samsung Research UK, aiming at building and curating a high quality food ontology which is

beneficial to artificial intelligence and information systems in domains such as personal health and agriculture. The entity clustering by OWL2Vec* can also contribute to ontology design by *e.g.*, discovering potential classes that have not been defined, as well as ontology QA by *e.g.*, entity resolution. OWL2Vec*, in collaboration with ZB MED - Information Centre for Life Sciences, also aims at being applied to identify clusters in an ontology and assign these clusters as topics (*i.e.*, a set of ontology classes) to a corpus of documents to enhance the results of an information retrieval task (Ritchie et al., 2021). In addition, OWL2Vec*, as an ontology tailored word embedding model, could replace the original word embedding models to increase performance in some domain specific tasks such as biomedical text analysis (Hao et al., 2020). This is also a promising direction worth studying.

References

Alshargi, F., Shekarpour, S., Soru, T., Sheth, A.P. (2018). Metrics for evaluating quality of embeddings for ontological concepts.

Arora, S., Liang, Y., Ma, T. (2019). A simple but tough-to-beat baseline for sentence embeddings. In: ICLR.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A Nucleus for a web of open data. *The semantic web* (pp. 722–735). Springer.

Baader, F., Horrocks, I., Lutz, C., & Sattler, U. (2017). *Introduction to Description Logic*. Cambridge University Press.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A., et al., (2004). OWL web ontology language reference. W3C recommendation *10*(02)

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence, 35*(8), 1798–1828.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In: NeurIPS. pp. 2787–2795.

Chen, J., Chen, X., Horrocks, I., Myklebust, B., & Jimenez-Ruiz, E. (2020). Correcting knowledge base assertions. *Proceedings of The Web Conference, 2020,* 1537–1547.

Chen, J., Jiménez-Ruiz, E., & Horrocks, I. (2019). Canonicalizing knowledge base literals. *International semantic web conference* (pp. 110–127). Springer.

Chen, J., Jimenez-Ruiz, E., Horrocks, I., Antonyrajah, D., Hadian, A., Lee, J. (2020). Augmenting ontology alignment by semantic embedding and distant supervision. In: European semantic web conference (ESWC).

Chen, J., Lecue, F., Geng, Y., Pan, J.Z., Chen, H. (2020). Ontology-guided Semantic Composition for Zero-Shot Learning. In: KR.

G. O. & Consortium. (2008). The gene ontology project in 2008. *Nucleic Acids Research, 36*(Supp 1), D440–D444.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL. pp. 4171–4186.

Dooley, D. M., Griffiths, E. J., Gosal, G. S., Buttigieg, P. L., Hoehndorf, R., Lange, M. C., et al., (2018). FoodOn: A harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food, 2*(1), 1–10.

Dragoni, M., Bailoni, T., Maimone, R., & Eccher, C. (2018). HeLis: An ontology for supporting healthy lifestyles. *International semantic web conference* (pp. 53–69). Berlin: Springer.

Garg, D., Ikbal, S., Srivastava, S.K., Vishwakarma, H., Karanam, H., Subramaniam, L.V. (2019). Quantum embedding of knowledge for reasoning. In: NeurIPS. pp. 5595–5605.

Glimm, B., Horrocks, I., Motik, B., Stoilos, G., & Wang, Z. (2014). HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning, 53*(3), 245–269.

Grover, A., Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In: KDD. pp. 855–864.

Guan, N., Song, D., & Liao, L. (2019). Knowledge graph embedding with concepts. *Knowledge-Based Systems, 164,* 38–44.

Hao, B., Zhu, H., Paschalidis, I. (2020). Enhancing clinical bert embedding using a biomedical knowledge base. In: Proceedings of the 28th international conference on computational linguistics. pp. 657–661

Hao, J., Chen, M., Yu, W., Sun, Y., Wang, W. (2019). Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In: KDD

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J.E.L., Kirrane, S., Neumaier, S., Polleres, A., et al., (2020). Knowledge graphs. arXiv preprint arXiv:2003.02320

Holter, O.M., Myklebust, E.B., Chen, J., Jimenez-Ruiz, E. (2019). Embedding OWL ontologies with OWL2Vec*. In: International semantic web conference (Posters and Demos).

Horrocks, I. (2008). Ontologies and the Semantic Web. *Communications ACM, 51*(12), 58–67.

Horrocks, I., Chen, J., Jaehun, L. (2020). Tool support for ontology design and quality assurance. In: ICBO 2020 integrated food ontology workshop (IFOW).

Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R. (2019). EL Embeddings: Geometric construction of models for the description logic EL++. In: IJCAI.

Kulmanov, M., Smaili, F. Z., Gao, X., & Hoehndorf, R. (2020). Semantic similarity and machine learning with ontologies. *Briefings in*. (**bioinformatics**).

Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In: AAAI.

Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9,* 2579–2605.

Miaschi, A., Dell'Orletta, F. (2020). Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In Proceedings of the 5th workshop on representation learning for NLP. pp. 110–119.

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In: NeurIPS. pp. 3111–3119.

Moon, C., Jones, P., Samatova, N.F. (2017). Learning entity type embeddings for knowledge graph completion. In: CIKM. pp. 2215–2218.

Myklebust, E. B., Jimenez-Ruiz, E., Chen, J., Wolf, R., & Tollefsen, K. E. (2019). Knowledge graph embedding for ecotoxicological effect prediction. *ISWC* (pp. 490–506). Springer.

Paulheim, H., & Gangemi, A. (2015). Serving DBpedia with DOLCE-more than just adding a cherry on top. *International semantic web conference* (pp. 180–196). Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al., (2011). Scikit-learn: Machine learning in python. *The Journal of machine Learning research, 12,* 2825–2830.

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018). Deep contextualized word representations. In Proceedings of NAACL-HLT. pp. 2227–2237.

Rajaraman, A., Ullman, J.D. (2011). Data mining, p. 1–17. Cambridge University Press https://doi.org/10.1017/CBO9781139058452.002

Ristoski, P., & Paulheim, H. (2016). RDF2Vec: RDF graph embeddings for data mining. *ISWC* (pp. 498–514). Springer.

Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., & Paulheim, H. (2019). RDF2Vec: RDF graph embeddings and their applications. *Semantic Web, 10*(4), 721–752.

Ritchie, A., Chen, J., Castro, L.J., Rebholz-Schuhmann, D., Jiménez-Ruiz, E. (2021). Ontology Clustering with OWL2Vec**. In: ESWC workshop DeepOntoNLP.

Rossi, A., Firmani, D., Matinata, A., Merialdo, P., Barbosa, D. (2020). Knowledge graph embedding for link prediction: A comparative analysis. arXiv preprint arXiv:2002.00819.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research, 12*(9).

Smaili, F. Z., Gao, X., & Hoehndorf, R. (2018). Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics, 34*(13), i52–i60.

Smaili, F. Z., Gao, X., & Hoehndorf, R. (2018). OPA2Vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction. *Bioinformatics, 35*(12).

Soylu, A., Kharlamov, E., Zheleznyakov, D., Jiménez-Ruiz, E., Giese, M., Skjæveland, M. G., et al., (2018). OptiqueVQS: A visual query system over ontologies for industry. *Semantic Web, 9*(5), 627–660.

Sun, Z., Zhang, Q., Hu, W., Wang, C., Chen, M., Akrami, F., Li, C. (2020). A Benchmarking Study of Embedding-based Entity Alignment for Knowledge Graphs. arXiv preprint.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I. (2017). Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008

Wang, C., Yan, M., Yi, C., & Sha, Y. (2019). Capturing semantic and syntactic information for link prediction in knowledge graphs. *International semantic web conference* (pp. 664–679). Springer.

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering, 29*(12), 2724–2743.

Wang, X., Ye, Y., Gupta, A. (2018). Zero-shot recognition via semantic embeddings and knowledge graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6857–6866.

Yanardag, P., Vishwanathan, S. (2015). Deep graph kernels. In: 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374.

Yang, B., Yih, W.t., He, X., Gao, J., Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575