# Ontology Completion Using Graph Convolutional Networks

Na Li[1], Zied Bouraoui[2], and Steven Schockaert[3]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
dg1733007@smail.nju.edu.cn
[2] CRIL - CNRS & University of Artois, France
zied.bouraoui@cril.fr
[3] Cardiff University, UK
SchockaertS1@Cardiff.ac.uk

**Abstract.** Many methods have been proposed to automatically extend knowledge bases, but the vast majority of these methods focus on finding plausible missing facts, and knowledge graph triples in particular. In this paper, we instead focus on automatically extending ontologies that are encoded as a set of existential rules. In particular, our aim is to find rules that are plausible, but which cannot be deduced from the given ontology. To this end, we propose a graph-based representation of rule bases. Nodes of the considered graphs correspond to predicates, and they are annotated with vectors encoding our prior knowledge about the meaning of these predicates. The vectors may be obtained from external resources such as word embeddings or they could be estimated from the rule base itself. Edges connect predicates that co-occur in the same rule and their annotations reflect the types of rules in which the predicates co-occur. We then use a neural network model based on Graph Convolutional Networks (GCNs) to refine the initial vector representation of the predicates, to obtain a representation which is predictive of which rules are plausible. We present experimental results that demonstrate the strong performance of this method.

**Keywords:** knowledge base completion · rule induction · graph convolutional networks · commonsense reasoning.

## 1 Introduction

Many approaches have been proposed in recent years for the problem of finding plausible missing facts in knowledge graphs, typically by learning vector space representations of the entities and relations that are predictive of plausible triples [8,27,33,43,48]. Beyond knowledge graphs, however, ontologies also play an important role on the Web [19]. For the ease of presentation, in this paper we will consider ontologies which are encoded as sets of existential rules [4], although our model would be straightforward to adapt to other formalisms such as description logics [2]. Similar to knowledge graphs, existing ontologies are often incomplete, hence there is a need for methods that can automatically predict plausible missing rules for a given ontology. For some ontologies, where we have a large database of facts (often called an ABox), plausible rules can be

learned similarly to how rules are learned in inductive logic programming and statistical relational learning [11,31,40,44]. However, for many commonly used ontologies, such a database of facts is not available. In this paper, we therefore address the challenge of predicting plausible missing rules based only on the rules that are in a given ontology (along with word embeddings in some variants of our proposed model).

This problem has thus far hardly received any attention, with the exception of [10]. The main underlying idea behind the approach from [10], which we will build on in this paper, is that ontologies often contain large sets of rules which only differ in one predicate. As a simple example, consider the following rules

$$\textit{Beer}(x) \rightarrow R(x)$$
$$\textit{Gin}(x) \rightarrow R(x)$$

Without knowing what the predicate $R$ represents, we can infer that the following rule is also valid:

$$\textit{Wine}(x) \rightarrow R(x)$$

This is intuitively because almost all natural properties which beer and gin have in common are also satisfied by wine. To formalize this intuition, [10] considered the notion of rule templates.

A rule template $\rho$ is a second-order predicate, which corresponds to a rule in which one predicate occurrence has been replaced by a placeholder. For example, in the above example, we can consider a template $\rho$ such that $\rho(P)$ holds if the rule $P(x) \rightarrow R(x)$ is valid, meaning that we would expect this rule to be entailed by the ontology if the ontology were complete. Given such a template $\rho$, we can consider the set of all instances $P_1, ..., P_n$ such that the corresponding rules $\rho(P_1), ..., \rho(P_n)$ are entailed by the given ontology. The main strategy for finding plausible rules proposed in [10] then essentially consists in finding predicates $P$ which are similar to $P_1, ..., P_n$. More precisely, the predicates are represented as vectors and it is assumed that each template $\rho$ can be modelled as a Gaussian distribution over the considered vector space, i.e. the probability that $\rho(P)$ is a valid rule is considered to be proportional to $\mathcal{G}_\rho(\mathbf{p})$, with $\mathbf{p}$ the vector representation of $P$ and $\mathcal{G}_\rho$ the Gaussian distribution modelling $\rho$. In addition to the templates described above, which are called *unary templates*, [10] also considered *binary templates*, which correspond to rules in which two predicate occurrences have been replaced by a placeholder. While unary templates enable a strategy known as interpolation, using binary templates leads to a form of analogical reasoning, both of which are well-established commonsense reasoning principles.

A critical aspect of this strategy for ontology completion is how the vector representation of the predicates is obtained. The approach from [10] relies on the combination of two types of vectors : (i) the word vector of the predicate name, obtained from a standard pre-trained word embedding [29]; (ii) a vector representation which is learned from the ontology itself, using a variant of the AnalogySpace method [41]. However, there are important limitations with this strategy. For instance, it is not clear why the predicates that satisfy a given template should follow a Gaussian distribution in the considered vector space. Moreover, the way in which the predicate representations are constructed

does not maximally take advantage of the available information. In particular, the approach based on the AnalogySpace method only relies on the known instances of the unary templates, i.e. binary templates are completely ignored for constructing the vector representations of the predicates. This is clearly sub-optimal, as knowing that $\rho(P, R)$ is a valid rule, for a given binary template $\rho$, intuitively tells us something about the semantic relationship between the predicates $P$ and $R$, which should in turn allow us to improve our representation of $P$ and $R$.

In this paper, we introduce a new method for predicting plausible rules which addresses both concerns. Our model is based on Graph Convolutional Networks (GCNs), a popular neural network architecture for graph-structured data [13,23,38]. We start from a graph-based representation of the rule base, in which the nodes correspond to predicates. Each node is annotated with a vector representation of the corresponding predicate. In this paper, we will use the vector representations from [10] for this purpose. Crucially, however, rather than using these vectors directly for making predictions as in [10], in our case they are merely used for initializing the GCN. Edges are annotated with the binary templates that are satisfied by the corresponding pair of predicates. We then propose a GCN model, which iteratively refines the vector encoding of the nodes, taking advantage of the edge annotations based on the binary templates. The resulting node vectors are then used to predict which predicates satisfy the different unary templates and which pairs of predicates satisfy the different binary templates, and thus to predict which rules are plausible. Note in particular, that our aim is to *learn* a vector representation of the predicates which is predictive of plausible rules, rather than relying on assumptions about a given vector representation. Our experimental results confirm that this approach is able to substantially outperform the method from [10].

## 2 Related Work

Within the area of knowledge base completion, we can broadly distinguish between two classes of methods: methods focused on finding plausible facts and methods focused on finding plausible rules.

*Predicting Facts.* In the last few years, there has been a large amount of work on finding missing triples in knowledge graphs. A popular strategy for this task is to rely on knowledge graph embedding, which aims to identify plausible triples by representing entities as vectors in a low-dimensional vector space and learning a parametrized scoring function for each relation. For instance, in the influential TransE model, relations are modelled as translations between the embeddings of entities [7], that is, $\mathbf{e}_h + \mathbf{e}_r \approx \mathbf{e}_t$, if (*h*, *r*, *t*) holds. Some other well-known approaches make use of bilinear scoring functions. For example, in [48] the authors propose to learn entity embeddings such that $\mathbf{e}_h^{\mathrm{T}} \mathbf{R}_r \mathbf{e}_t$ is higher for correct triples $(h, r, t)$ than for incorrect triples. Here $\mathbf{e}_h$ and $\mathbf{e}_t$ are the embeddings of the entities $h$ and $t$, and $\mathbf{R}_r$ is a diagonal matrix representing the relation $r$. The ComplEx model [43] is an extension of [48] in the complex space. A different strategy consists in learning latent soft clusters of predicates to predict missing facts in relational data, for example by using Markov logic network [24] or by applying neural network models [36,40]. Several rule-based approaches have also been proposed, where

observed regularities in the given knowledge graph are summarized as a weighted set of rules, which is then used to derive plausible missing facts. For instance, a soft inference procedure was proposed in [26] to infer different relations by tuning the weights associated with random walks that follow different paths through the graph. [16] proposed a novel method with iterative guidance from soft rules with various confidence levels extracted automatically from the knowledge graph. The aforementioned strategies all rely on exploiting statistical regularities in the given knowledge graph. There are also several ways in which external knowledge can be used to predict missing facts. One possibility is to rely on information extraction from text corpora [1,25]. In this setting, one can distinguish between methods based on a generic question answering system [45] and methods which use the given knowledge bases as a distant supervision signal [30,34]. Apart from directly relying on text corpora, some approaches have instead relied on pre-trained entity embeddings, which can be learned from open-domain resources such as Wikipedia, WikiData or BabelNet [12,20]. For instance [9] focused on finding missing instances of concepts in the context of ontologies, by modelling these concepts as Gaussians in a given vector space. This problem of ABox induction was also considered in [6], which instead relied on kernels for structured data to capture similarities between entities. A similar problem was also considered in [32], which relied on features that were directly derived from Wikipedia. Finally, various approaches have also been proposed to combine the two main aforementioned strategies, for example by incorporating textual descriptions of entities when learning knowledge graph embeddings [21,46,47,49], or by incorporating relation extraction methods [35,42].

*Predicting Rules.* The problem of learning rules, in the context of ontologies, has been approached from two different angles. First, we can identify methods that induce rules based on the given (relational) facts, e.g. based on ideas from the field of Statistical Relational Learning. For example, [11] proposed a system inspired by inductive logic programming, while [44] introduced statistical schema induction to mine association rule from RDF data and then generate ontologies. More recently, [40] used so-called Lifted Relational Neural Networks to learn rules in an implicit way. In [31], meta-rules were found automatically by meta-interpretive learning. Some other methods, e.g. [3], used Formal Concept Analysis. What all these approaches have in common is that a sufficiently large database is required to be able to learn rules from a given ontology, which is however, not the case for the majority of available ontologies on the Web. The second class of methods is concerned with predicting rules directly from the ontology itself, which did not receive much attention yet. From a purely theoretical side, this problem has been studied in a propositional setting in [39], where methods based on interpolation and extrapolation of rules were proposed. However, the implementation of these methods requires some background knowledge (e.g. a betweenness relation is required to apply interpolation), which is not often available. In [5], a method that implements a kind of similarity based reasoning using Markov logic has been proposed in order to find plausible rules. The idea of similarity based reasoning has been also pursued in logic programming to extend the unification mechanism [28,37]. As already mentioned in Section 1, [10] recently proposed a method that relies on the notion of rule templates and the estimation of Gaussian distributions over predicate embeddings to make predictions.

*Graph Convolutional Networks.* In this paper, we use a variant of Graph Convolutional Networks (GCNs) to learn a vector representation of the predicates that occur in our rule base which is suitable for predicting plausible rules. GCNs are a generalization of Convolutional Neural Networks (CNNs). Whereas the latter require data with a regular structure, such as images or sequences, GCNs allow for irregular graph-structured data. GCNs can learn to extract features from the given node representations, and compose these features to construct highly expressive node vectors. These node vectors can then be used in a wide variety of graph-related tasks, such as graph classification [13] and graph generation [15]. Recently, researchers have applied GCNs to find missing facts in knowledge bases [17,38]. For example, [17] use GCNs for the standard triple classification and out-of-knowledge-base entity problems. Schlichtkrull et al. [38] model multi-relational data using GCNs for entity classification and link prediction. However, to our knowledge, this paper is to first to use GCNs for rule base completion.

## 3 A GCN Model for Rule Induction

Let $\mathcal{R}$ be a rule base, i.e. a set of rules. Our aim is to find additional rules that intuitively appear to be plausible, even if they cannot be deduced from $\mathcal{R}$. Throughout our description, we will assume that $\mathcal{R}$ contains existential rules [4], i.e. rules of the following form:

$$r_1(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y} \,.\, s_1(\mathbf{z_1}) \wedge ... \wedge s_m(\mathbf{z_m}) \tag{1}$$

where $\mathbf{x_1}, ..., \mathbf{x_n}, \mathbf{y}, \mathbf{z_1}, ..., \mathbf{z_m}$ are tuples of variables. We consider existential rules because they are an expressive and well-studied framework for representing ontologies. However, because our method treats these rules as purely syntactic objects, it is in fact not tied to any particular logical framework or semantics. We could readily apply the same method to description logics, for instance.

### 3.1 Graph Representation of the Rule Base

Before we introduce our proposed method in Section 3.2, we now first describe how the rule base $\mathcal{R}$ can be encoded as a graph.

*Rule Templates.* As mentioned in Section 1, our graph encoding of the rule base will rely on the notion of rule templates from [10]. Rule templates are second-order predicates, which correspond to a rule in which one (for unary templates) or two (for binary templates) occurrences of a predicate have been replaced by a placeholder. For a unary template $\rho$ and a predicate $P$, we write $\rho(P)$ to denote the rule that is obtained by instantiating the placeholder with $P$, and similar for binary templates. We say that $P$ satisfies $\rho$ if $\rho(P)$ is a valid rule in the considered domain. If $\mathcal{R}$ were complete, then $P$ would satisfy $\rho$ iff $\mathcal{R}$ entails $\rho(P)$. In general, however, the rule base $\mathcal{R}$ is incomplete, which means that it only partially specifies which predicates satisfy the template $\rho$. In particular, suppose that $P_1, ..., P_n$ are all the predicates for which $\rho(P_i)$ can be deduced from the given rule base $\mathcal{R}$. Then $P_1, ..., P_n$ are the only predicates which are

known to satisfy the template $\rho$. The problem we consider below is to identify additional predicates $P$ which are likely to satisfy $\rho$, or equivalently, identify rules of the form $\rho(P)$ which are valid in the considered domain but missing from the given rule base. However, rather than considering this problem for a single template, we consider all the possible templates that occur in $\mathcal{R}$.

Let $\theta$ be an existential rule of the form (1). Then $\theta$ is associated with the following unary templates:

$$\begin{aligned}\rho_1(\star) &= \star(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge s_m(\mathbf{z_m}) \\ &... \\ \rho_{n+m}(\star) &= r_1(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge \star(\mathbf{z_m})\end{aligned}$$

as well as the following binary templates:

$$\begin{aligned}\rho_{1,2}(\star, \bullet) &= \star(\mathbf{x_1}) \wedge \bullet(\mathbf{x_2}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge s_m(\mathbf{z_m}) \\ &... \\ \rho_{1,n+m}(\star, \bullet) &= \star(\mathbf{x_1}) \wedge r_2(\mathbf{x_2}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge \bullet(\mathbf{z_m}) \\ &... \\ \rho_{n+m-1,n+m}(\star, \bullet) &= r_1(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge \star(\mathbf{z_{m-1}}) \wedge \bullet(\mathbf{z_m})\end{aligned}$$

In addition to these templates, we also consider typed templates. In particular, assume that the predicates are organized in a taxonomy and let $\rho$ be a rule template that was obtained by replacing the predicate $P$ in the rule $\theta$ by a placeholder. Let $Q$ be a parent of $P$ in the taxonomy (i.e. we have that $\mathcal{R}$ contains the rule $P(x) \rightarrow Q(x)$). Then the corresponding typed version of $\rho$, denoted by $\rho^Q$, is satisfied by those predicates $P'$ that satisfy $\rho$ and that also have $Q$ as a direct parent.

We denote respectively by $\mathcal{L}_1(\theta)$ and $\mathcal{L}_2(\theta)$ the set of all unary and binary templates that can be obtained from the rule $\theta$ (including both typed and untyped templates). We also let $\mathcal{L}_1(\mathcal{R}) = \bigcup_{\theta \in \mathcal{R}} \mathcal{L}_1(\theta)$ and $\mathcal{L}_2(\mathcal{R}) = \bigcup_{\theta \in \mathcal{R}} \mathcal{L}_2(\theta)$ be respectively the set of all unary and binary templates that can be obtained from the set of rules $\mathcal{R}$.

*Graph Representation.* We encode the rule base $\mathcal{R}$ as a graph $\mathcal{G}_\mathcal{R} = (\mathcal{P}_\mathcal{R}, \mathcal{E})$ where $\mathcal{P}_\mathcal{R}$ is a set of all predicates that occur in $\mathcal{R}$ and $\mathcal{E}$ contains all pairs of predicates $(P, Q)$ that co-occur in at least one rule in $\mathcal{R}$. To capture the knowledge encoded in the rule base (as well as potentially some external knowledge), we use two labelling functions. The node labelling function $\eta$ maps each predicate $P$ from $\mathcal{P}_\mathcal{R}$ onto a real valued vector $\eta(P) \in \mathbb{R}^d$. This vector can be viewed as the input encoding of the predicate $P$ and can be defined in different ways (see below). The edge labelling function $\xi$ maps each pair of predicates $(P, Q)$ from $\mathcal{E}$ onto a binary vector $\xi(P, Q) \in \{0, 1\}^m$, where $m = |\mathcal{L}_2(\mathcal{R})|$. In particular, let $\rho_1, ..., \rho_m$ be an enumeration of all binary templates from $\mathcal{L}_2(\mathcal{R})$. The $i^{th}$ coordinate of the vector $\xi(P, Q)$ is 1 iff the rule $\rho_i(P, Q)$ occurs in $\mathcal{R}$.

*Node Vectors.* To construct the input encoding $\eta(P)$ of predicate $P$, we will either use a vector $\eta_w(P)$ derived from the name of predicate $P$ using a standard pre-trained word embedding, or a vector $\eta_t(P)$ that encodes which of the unary templates from

$\mathcal{L}_1(\mathcal{R})$ are satisfied by $P$. Specifically, to obtain the vector $\eta_w(P)$, we first tokenize the predicate name using a small set of simple heuristics, based on standard ontology naming conventions[4]. For example, the predicate name *RedWine* gives the following list of words: $(red, wine)$. Let $(w_1, ..., w_n)$ be the list of words thus obtained, then the vector representation $\eta_w(P)$ of $P$ is simply obtained by averaging the vector representations of these words. Namely, $\eta_w(P) = \frac{1}{n}(\mathbf{w_1} + ... + \mathbf{w_n})$, where $\mathbf{w_i}$ denotes for the vector representation of word $w_i$ in the word embedding. Even though this averaging strategy may seem naive, it is known to be surprisingly effective for capturing the meaning of phrases and sentences [18].

The vector $\eta_t(P)$, encoding knowledge about $P$ derived from the unary templates, is constructed as follows. First, we consider a binary vector $\eta_t^B(P) \in \{0,1\}^k$ with $k = |\mathcal{L}_1(\mathcal{R})|$, whose $i^{th}$ coordinate is 1 iff the $i^{th}$ unary template, in some arbitrary but fixed enumeration of the unary templates, is satisfied by $P$. In other words, $\eta_t^B$ is thus the counterpart of $\xi$ for unary templates. We then define $\eta_t(P) \in \mathbb{R}^l$ as the a low-dimensional approximation of $\eta_t^B$, obtained using singular value decomposition (SVD) as in [10]. In particular, let $X$ be a matrix with one row for each predicate, where the row corresponding to $P$ is given by the vector $\eta_t^B(P_i)$. Let $X = U\Sigma V^T$ be the singular value decomposition of $X$. Then $\eta_t(P)$ is given by the first $l$ columns of the row corresponding to $P$ in the matrix $U\Sigma$. This use of the singular value decomposition is a well-known technique to compress the information encoded in the vectors $\eta_t^B(P)$ into a lower-dimensional representation. Note that the vectors $\eta_t(P)$ and $\eta_t(Q)$ will be similar if the sets of unary templates satisfied by $P$ and $Q$ are similar.

### 3.2 GCN Model

*Background.* Graph Convolutional Networks (GCNs) produce node-level embeddings of graphs, by iteratively exchanging the current vector representations of the nodes along the edges of the graph. GCNs are thus essentially message-passing models. Let us write $\mathbf{h}_i^{(0)}$ for the initial vector representation of node $n_i$. A GCN iteratively refines this representation based on the following propagation rule [14]:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} f\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right) \tag{2}$$

where $\mathcal{N}_i$ is the neighborhood of $n_i$, i.e. the set of nodes that are incident with $n_i$. Furthermore, $f(\cdot,\cdot)$ is a transformation function, which is used to combine the current representation of $n_i$ with the current representation of a given neighbor $n_j$. Both linear and non-linear transformations can be used for this purpose, but we will restrict ourselves to linear transformations in this paper. These transformed representations are intuitively viewed as messages which are sent from the neighbors of $n_i$. These messages are then aggregated (using a summation) after which a non-linear activation function $\sigma$ is used. We will use the ReLU function for this purpose, defined by $\sigma(x) = \max(0, x)$.

[4] http://wiki.opensemanticframework.org/index.php/Ontology_Best_Practices

*Model Description.* The standard formulation in (2) does not take into account edge labels, which play an important role in our setting as they encode the nature of the relationship between the (predicates corresponding to the) two nodes. Let us write $\mathcal{N}_P^{\rho_i}$ for the set of all nodes $Q$ that are connected with $P$ in our graph for which $(P, Q)$ is an instance of the binary template $\rho_i$, i.e. $(P, Q) \in \mathcal{E}_\mathcal{R}$ and the $i^{th}$ component of $\xi(P, Q)$ is 1.

We specifically consider the following variant:

$$\mathbf{h}_P^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_P^{(l)} + \sum_{\rho \in \mathcal{L}_2(\mathcal{R})} \sum_{Q \in \mathcal{N}_P^\rho} \frac{1}{|\mathcal{N}_P^\rho|}\mathbf{W}_\rho^{(l)}\mathbf{h}_Q^{(l)}\right) \tag{3}$$

where we write $\mathbf{h}_P^{(l)}$ for the embeddings of the (node corresponding to) predicate $P$. In the input layer, $\mathbf{h}_P^{(0)}$ is the vector representation of the node $P$ given by the label $\eta(P)$. The matrix $\mathbf{W}_\rho^{(l)}$ encodes a template-specific linear transformation, which together with the node transformation $\mathbf{W}_0^{(l)}$ defines the $l$-th layer of our model.

We now describe how the GCN model can be used to predict plausible instances of the considered unary and binary templates. Note that each such a prediction corresponds to the prediction of a plausible rule, as mentioned in Section 3.1.

*Unary Template Prediction.* We treat the problem of predicting plausible instances of unary templates as a multi-label node classification problem. To this end, we add an output layer to the GCN which has one neuron for each unary template and each predicate, i.e. for each predicate-template combination we make a prediction about whether the template applies to that predicate. We use a sigmoid activation function for this output layer and we use the cross-entropy loss function to train the model:

$$J = -\sum_{\rho \in \mathcal{L}_1(\mathcal{R})} \sum_{Q \in \mathcal{P}_\mathcal{R}} y_Q^\rho \log(p_Q^\rho) + (1 - y_Q^\rho)\log(1 - p_Q^\rho)$$

where $p_Q^\rho \in [0, 1]$ is the model's prediction that predicate $Q$ satisfies template $\rho$ and $y_Q^\rho \in \{0, 1\}$ is the corresponding ground truth, i.e. $y_Q^\rho = 1$ iff $\rho(Q)$ can be entailed from $\mathcal{R}$. Note that when training this model, we thus implicitly assume that the rule base $\mathcal{R}$ is complete. However, the capacity of the GCN model is not sufficient to perfectly satisfy this training objective, which means that it will make some mistakes and predict some rules which are, in fact, not entailed by $\mathcal{R}$. These "mistakes" then correspond to the rules which we view to be plausible. Indeed, the reason why the GCN model predicts such a rule $\rho(P)$ is it is not able to separate $P$ from the predicates that are known to satisfy $\rho$, which suggests that $P$ is semantically similar to such predicates, and thus that $\rho(P)$ should be considered as plausible.

For the ease of presentation, in the formulation of the loss function above, we assumed that all templates are untyped. For typed templates, rather than considering all predicates $Q \in \mathcal{P}_\mathcal{R}$, we only consider those of the correct type. Furthermore, in the experiments, we add the following regularization term to the loss function, which we empirically found to be helpful:

$$J_{reg} = \sum_{\rho \in \mathcal{L}_2(\mathcal{R})} \sum_{(Q,S) \in \mathcal{N}_P^\rho} \|\mathbf{h}_Q - \mathbf{h}_S\|_2^2$$

where we write $\mathbf{h}_P$ for the embedding of predicate $P$ in the final layer. Note that this regularization is thus only applied to the final embeddings, i.e. the layer before the classification layer, instead of all layers.

The intuitive justification is that predicates which often co-occur in the same rule are likely to be semantically related. This is particularly useful because the majority of the rules in a typical ontology are basic subsumption rules of the form $P(x) \rightarrow Q(x)$. In some cases, we do not have much information about the parent concept $Q$ (e.g. because $Q$ is an abstract concept), in which case the regularization term will encourage its representation to be close to the average of the representations of its children. Conversely, it may also be the case that we instead do not have much information about $P$ (e.g. because it is too specialized), in which case the regularization term would encourage the representation of $P$ to stay close to the representation of its parent.

*Binary Template Prediction.* We view the problem of predicting binary template instances as a link prediction problem. For each pair of predicates $(P, Q)$ from $\mathcal{P}_\mathcal{R}$ and each binary template $\rho \in \mathcal{L}_2(\mathcal{R})$, the task is to predict whether $(P, Q)$ satisfies $\rho$. To this end, we need a scoring function for each template $\rho$ such that $s_\rho(P, Q)$ is high for valid pairs $(P, Q)$ and low for other pairs. In principle, any of the scoring functions that have been proposed for knowledge graph embedding could be used for this purpose. In our experiments, we will use the following bilinear scoring function [48]:

$$s(P, \rho, Q) = \mathbf{h}_P^T \mathbf{R}_\rho \mathbf{h}_Q,$$

where $\mathbf{h}_P$ and $\mathbf{h}_Q$ are the final-layer vector representations of the predicates, as before. Furthermore, $\mathbf{R}_\rho$ is a diagonal matrix which corresponds to the representation that is learned for the binary template $\rho$. Note that while this scoring function is symmetric, this symmetry is broken in practice when using typed binary templates. This is because the only situation in which both the rules $\rho(P, Q)$ and $\rho(Q, P)$ would be considered is when they are of the same type (i.e. they have the same parent), which is almost never the case. In order to train the model, we sample negative examples by randomly corrupting one of the predicates in positive examples. We apply a sigmoid function to the scoring function and then again train the model using a cross-entropy loss.

## 4 Model Evaluation

In this section, we experimentally evaluate our method[5], comparing it against the method from [10] as our baseline.

*Methodology.* The datasets we consider are constructed from the OWL version of the following ontologies: SUMO[6], which is a large open domain ontology, as well as Wine[7],

[5] Implementation and data are available at https://github.com/bzdt/GCN-based-Ontology-Completion.git

[6] http://www.adampease.org/OP/

[7] https://www.w3.org/TR/2003/PR-owl-guide-20031215/wine

Table 1: Parameter settings for *GCN* models.

| | Wine | | Economy | | Olympics | | Transport | | SUMO | |
|---|---|---|---|---|---|---|---|---|---|---|
| | UT | BT | UT | BT | UT | BT | UT | BT | UT | BT |
| $GCN_{mf}$ lr | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 |
| $GCN_{mf}$ hid | 32 | 32 | 64 | 32 | 32 | 32 | 64 | 32 | 32 | 32 |
| $GCN_{mf}$ ly | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 5 | 5 |
| $GCN_{mf}$ l2 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 |
| $GCN_{we}$ lr | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 |
| $GCN_{we}$ hid | 32 | 32 | 64 | 32 | 32 | 32 | 64 | 32 | 32 | 32 |
| $GCN_{we}$ ly | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 5 | 5 |
| $GCN_{we}$ l2 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 |
| $GCN_{cm}$ lr | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 |
| $GCN_{cm}$ hid | 32 | 32 | 64 | 32 | 32 | 32 | 64 | 32 | 32 | 32 |
| $GCN_{cm}$ ly | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 5 | 6 |
| $GCN_{cm}$ l2 | 0.1 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 |
| $GCN_{con}$ lr | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 |
| $GCN_{con}$ hid | 32 | 128 | 64 | 32 | 64 | 64 | 32 | 64 | 32 | 32 |
| $GCN_{con}$ ly | 3 | 4 | 3 | 4 | 4 | 4 | 3 | 4 | 5 | 6 |
| $GCN_{con}$ l2 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0.1 |

Economy[8], Transport[9] and Olympics[10], which are smaller domain-specific ontologies. These OWL ontologies were then converted into existential rules (where we simply omitted those OWL axioms that cannot be expressed in this way). In the experiments, we use a standard pre-trained 300-dimensional word embedding learned using Skip-gram on the 100B words Google News corpus[11].

To evaluate the performance of our model, we split the considered rule bases into training and test sets. We use 10-fold cross validation for the small ontologies, while for the larger SUMO ontology, we use a fixed $2/3$ split for training and $1/3$ for testing. After splitting each rule base, we applied Pellet Reasoner[12] to determine all rules that can be derived from each training split. Subsequently, we removed from the corresponding test split all rules that could be derived from the training split. The derived rules are kept in the training split, i.e. we apply our model to the deductive closure of the rules in the training data.

Clearly, because it is based on rule templates, our GCN model can only predict rules that correspond to instances of rule templates which occur in the training data. Our evaluation therefore focuses on predicting, for all of the unary (resp. binary) templates found in the training data, which predicates (resp. pairs of predicates) are likely to satisfy them, beyond those instances that are already found in the training data. Furthermore, we can only make predictions about predicates that occur in the training data, so any predicates that only appear in the test split are also ignored.

[8] http://reliant.teknowledge.com/DAML/Economy.owl
[9] http://reliant.teknowledge.com/DAML/Transportation.owl
[10] http://swat.cse.lehigh.edu/resources/onto/olympics.owl
[11] https://code.google.com/archive/p/word2vec/
[12] https://github.com/stardog-union/pellet

For evaluation purposes, we assume that a prediction is correct iff the corresponding rule can be derived from the given ontology (i.e. training and test split). This is clearly a simplifying assumption, given that our starting point is that some valid rules are actually missing. As a result, the reported evaluation scores should be viewed as a lower approximation of the performance of the methods (given that some predictions which are assessed to be false may actually be correct rules that were missing in the original ontology). Importantly, however, this still allows us to compare the relative performance of different methods. This evaluation strategy follows common practice in the context of knowledge base completion (e.g. the standard benchmarks for knowledge graph completion also rely on this simplifying assumption).

We choose the number of layers according to the size of the ontology. For small ontologies (e.g. Wine), a limited number of layers is preferable to avoid overfitting, while for larger ontologies (e.g. SUMO), it makes sense to use more layers as more training data is available for these cases. Specifically, for unary template prediction, we use a model consisting of 3 GCN layers for the small datasets (which includes the output layer), and 5 GCN layers for SUMO. For the first two layers we use a ReLU activation function, while sigmoid is used for the output layer. Regardless of the number of GCN layers, sigmoid is always used for the last layer and ReLU for the other layers. For the binary template prediction, we use 2 GCN layers with ReLU activation for the small datasets, and 3 GCN layers for SUMO. This is followed by a scoring layer and a fully connected layer using sigmoid. Crucially, to avoid overfitting and encourage the model to generalize beyond the given instances of the templates, we apply dropout (dropout rate = 0.5) to the hidden layers. We also use L2-norm regularization, which encourages the model to focus on the most informative binary templates only when aggregating the messages (noting that the model would converge to $\mathbf{W}_{\rho}^{(l)} = 0$ if template $\rho$ were not informative). We have implemented the model in the Deep Graph Library (DGL) [13], using the Adam optimizer [22] for training. We considered four variants of the GCN model:

- $\text{GCN}_{\text{mf}}$ uses the $\eta_t(P)$ vector based on SVD decomposition as initial representation of $P$.
- $\text{GCN}_{\text{we}}$ uses the predicate representations $\eta_w(P)$ obtained from the word embedding as input vectors.
- $\text{GCN}_{\text{cm}}$ combines the two independent models, i.e. it trains models for both $\eta_t(P)$ and $\eta_w(P)$ independently, and combines their predictions. We calculate the performance measures of the union set of the rules predicted using both models. For a given rule, as long as one of the two models predicts it correctly, it is considered a correct prediction.
- $\text{GCN}_{\text{con}}$ combines the two representations, i.e. it uses the concatenation of $\eta_t(P)$ and $\eta_w(P)$ as the input encoding of $P$.

As our baseline, we consider the model from [10], which we will refer to as BRI. We consider four variants of this model, being direct counterparts to the four variants of our model:

- $\text{BRI}_{\text{mf}}$ uses the $\eta_t(P)$ vector to represent predicates.

[13] https://docs.dgl.ai

Table 2: Results of the rule induction experiments.

| | | Wine | | Economy | | Olympics | | Transport | | SUMO | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UT | BT | UT | BT | UT | BT | UT | BT | UT | BT |
| $BRI_{mf}$ | Pr | 0.030 | 0.583 | 0.091 | 0.992 | 0.150 | 0.286 | 0.000 | 0.600 | 0.534 | 1.000 |
| $BRI_{mf}$ | Rec | 0.045 | 0.180 | 0.070 | 0.309 | 0.108 | 0.191 | 0.000 | 0.173 | 0.201 | 0.072 |
| $BRI_{mf}$ | F1 | 0.032 | 0.259 | 0.075 | 0.445 | 0.114 | 0.214 | 0.000 | 0.251 | 0.292 | 0.134 |
| $BRI_{we}$ | Pr | 0.118 | 0.400 | 0.093 | 0.993 | 0.307 | 0.286 | 0.000 | 1.000 | 0.791 | 0.969 |
| $BRI_{we}$ | Rec | 0.311 | 0.073 | 0.294 | 0.599 | 0.225 | 0.238 | 0.000 | 0.464 | 0.287 | 0.328 |
| $BRI_{we}$ | F1 | 0.159 | 0.124 | 0.138 | 0.742 | 0.234 | 0.257 | 0.000 | 0.609 | 0.421 | 0.490 |
| $BRI_{cm}$ | Pr | 0.118 | 0.700 | 0.089 | **0.992** | 0.407 | 0.250 | 0.000 | **1.000** | 0.802 | 0.971 |
| $BRI_{cm}$ | Rec | 0.331 | 0.234 | 0.297 | **0.627** | 0.325 | 0.250 | 0.000 | **0.538** | 0.316 | 0.348 |
| $BRI_{cm}$ | F1 | 0.162 | 0.330 | 0.135 | **0.765** | 0.334 | 0.250 | 0.000 | **0.667** | 0.453 | 0.513 |
| $BRI_{con}$ | Pr | 0.094 | 0.600 | 0.072 | 0.855 | 0.200 | 0.286 | 0.000 | 0.367 | 0.250 | 0.750 |
| $BRI_{con}$ | Rec | 0.102 | 0.288 | 0.101 | 0.267 | 0.050 | 0.191 | 0.000 | 0.132 | 0.002 | 0.015 |
| $BRI_{con}$ | F1 | 0.085 | 0.364 | 0.083 | 0.387 | 0.079 | 0.214 | 0.000 | 0.187 | 0.005 | 0.030 |
| $GCN_{mf}$ | Pr | 0.489 | 0.475 | **0.750** | 0.733 | 0.278 | 0.286 | 0.010 | 0.400 | 0.543 | 0.732 |
| $GCN_{mf}$ | Rec | 0.349 | 0.244 | **0.153** | 0.180 | 0.292 | 0.286 | 0.018 | 0.077 | 0.421 | 0.409 |
| $GCN_{mf}$ | F1 | 0.334 | 0.313 | **0.243** | 0.269 | 0.273 | 0.286 | 0.013 | 0.125 | 0.474 | 0.524 |
| $GCN_{we}$ | Pr | 0.645 | 0.900 | 0.172 | 0.911 | 0.350 | 0.429 | 0.020 | 0.850 | 0.719 | 0.836 |
| $GCN_{we}$ | Rec | 0.259 | 0.356 | 0.218 | 0.526 | 0.392 | 0.429 | 0.033 | 0.267 | 0.396 | 0.493 |
| $GCN_{we}$ | F1 | 0.355 | 0.488 | 0.183 | 0.651 | 0.328 | 0.429 | 0.021 | 0.387 | 0.510 | 0.620 |
| $GCN_{cm}$ | Pr | 0.465 | 0.875 | 0.175 | 0.891 | **0.465** | 0.429 | 0.118 | 0.850 | **0.778** | **0.884** |
| $GCN_{cm}$ | Rec | 0.382 | 0.444 | 0.232 | 0.591 | **0.533** | 0.429 | 0.033 | 0.313 | **0.437** | **0.516** |
| $GCN_{cm}$ | F1 | 0.353 | 0.563 | 0.189 | 0.688 | **0.463** | 0.429 | 0.036 | 0.434 | **0.559** | **0.651** |
| $GCN_{con}$ | Pr | **0.416** | **0.900** | 0.163 | 0.912 | 0.233 | **0.857** | **0.371** | 0.454 | 0.692 | 0.812 |
| $GCN_{con}$ | Rec | **0.356** | **0.476** | 0.245 | 0.585 | 0.267 | **0.762** | **0.044** | 0.139 | 0.374 | 0.485 |
| $GCN_{con}$ | F1 | **0.356** | **0.607** | 0.191 | 0.698 | 0.242 | **0.786** | **0.077** | 0.201 | 0.485 | 0.607 |

- $BRI_{we}$ uses the representation $\eta_w(P)$ based on word vectors.
- $BRI_{cm}$ combines the predictions of the $BRI_{mf}$ and $BRI_{we}$ models.
- $BRI_{con}$ uses the concatenation of $\eta_t(P)$ and $\eta_w(P)$.

To tune the parameters of the models, we randomly select 10% of the training data as a validation set. The parameters to be tuned include the learning rate (chosen from $\{0.1, 0.01, 0.001\}$) for Adam optimization, the number of units in the hidden layers (chosen from $\{16, 32, 64, 128, 256\}$), the dimensionality of the input encodings of the predicates in cases where we use the SVD based method (chosen from $\{20, 30, 40, 50, 100\}$) and the threshold for classification and the hyperparameter for L2 regularization. Table 1 reports the different settings that were selected. The trade-off hyperparameters of the regularzier $J_{reg}$ for unary template prediction are 0.01 for the Economy and Transport ontologies and 0.1 for Wine and Olympics ontologies. We use the same parameters for each fold. For instance, for the Wine ontology, the number of units is 32 and we use a 40-dimensional input encoding of the predicates. The hyperparameter for L2 is set to 0 for the unary template prediction and to 0.1 for binary template prediction respectively.

*Quantitative Evaluation.* Table 2 reports the performance of the different models in terms of precision (Pr), recall (Rec) and F1 score. Note that both unary template predictions and binary template predictions are the multi-label classification tasks. However, what matters in the prediction is not how many nodes or links are classified correctly, but how successful the models are at predicting missing rules. Therefore, the precision, recall and F1 scores are computed w.r.t. the number of correctly predicted rules instead. In all cases, we use micro-averaging to calculate the overall precision, recall and F1 scores.

The results in Table 2 show that the GCN model is indeed able to outperform the BRI model from [10]. The table separately shows the performance of models which only rely on unary templates (UT) for predicting plausible rules and models which only rely on binary templates (BT). As can be seen, for UT, the GCN models consistently, and often substantially, outperform the BRI counterparts, which demonstrates that the GCN models are able to improve the representation of the predicates by propagating and incorporating the information received from related predicates. In the case of the BT results, the GCN models perform best on the Wine, Olympics and SUMO ontologies, but they perform less well on the Economy and Transport ontologies. This can be explained by the fact that the number of examples we have for each binary template in these cases is much lower, which can result in overfitting on the training data. In contrast, for SUMO, which is by far the largest ontology, the outperformance of our model is consistent and very substantial. Finally, when comparing the $\text{GCN}_{\text{mf}}$ and $\text{GCN}_{\text{we}}$ variants, we clearly see that using word embeddings to initialize the node vectors leads to the best results, although both models are outperformed by the concatenation based model $\text{GCN}_{\text{con}}$ or the combined model $\text{GCN}_{\text{cm}}$. Comparing the performance of $\text{GCN}_{\text{cm}}$ and $\text{GCN}_{\text{con}}$, we can see that the concatenation model $\text{GCN}_{\text{con}}$ generally performs better. Interestingly, the difference in performance between $\text{GCN}_{\text{cm}}$ and $\text{GCN}_{\text{con}}$ is more mixed.

*Qualitative Analysis.* We illustrate the performance of the GCN model by discussing some examples of predicted rules. As an example from the UT setting, our model was able to correctly predict the following rule from the Wine ontology:

$$\textit{DryRedWine}(x) \rightarrow \textit{TableWine}(x)$$

by using the template $\rho(\star) = \star(x) \rightarrow \textit{TableWine}(x)$. The instances of this template that were given in the training data are *RedTableWine*, *DryWhiteWine* and *Burgundy*. Based on these instances, the BRI model was not able to predict that *DryRedWine* is also a plausible instance. The GCN models, however, were able to exploit edges (i.e. binary templates) corresponding to the following rules:

$$\begin{aligned}
\textit{Merlot}(x) &\rightarrow \textit{DryRedWine}(x)\\
\textit{Merlot}(x) &\rightarrow \textit{RedTableWine}(x)\\
\textit{DryRedWine}(x) &\rightarrow \textit{DryWine}(x)\\
\textit{DryWhiteWine}(x) &\rightarrow \textit{DryWine}(x)\\
\textit{Burgundy}(x) &\rightarrow \textit{DryWine}(x)
\end{aligned}$$

As an example from the BT setting, the GCN model was able to correctly predict the following rule from the Olympics ontology:

$$\textit{WomansTeam}(x) \rightarrow \exists y\,.\,\textit{hasMember}(x, y) \wedge \textit{Woman}(y)$$

based on the following rules from the training data:

$$\textit{MensTeam}(x) \rightarrow \exists y\,.\,\textit{hasMember}(x, y) \wedge \textit{Man}(y)$$
$$\textit{MixedTeam}(x) \rightarrow \exists y\,.\,\textit{hasMember}(x, y) \wedge \textit{Woman}(y)$$

This illustrates the ability of models based on binary templates to perform analogical reasoning. Note that this rule cannot be predicted in the setting where only unary templates are used.

From a practical perspective, an important question is whether our model is able to find rules which are missing from the existing ontologies, rather than merely identifying held-out rules (as we did in the experiments above). Here we present some examples of rules that were predicted by our model, but which cannot be deduced from the full ontologies. These predictions are based on a GCN model that was trained on the full ontologies. Some of the rules we obtained are as follows:

$$\textit{Cycle}(x) \rightarrow \textit{LandVehicle}(x)$$
$$\textit{AgriculturalProduct}(x) \rightarrow \textit{Product}(x) \wedge \textit{Exporting}(x)$$
$$\textit{CargoShip}(x) \rightarrow \textit{Ship}(x) \wedge \textit{DryBulkCargo}(x)$$

As can be seen, these rules intuitively make sense, which suggests that our approach could indeed be useful to suggest missing rules in a given ontology. Since there exists rule $\textit{Bicycle}(x) \rightarrow \textit{Cycle}(x)$ in the Transport ontology, which makes $\textit{Cycle}(x) \rightarrow \textit{LandVehicle}(x)$ plausible. $\textit{AgriculturalProduct}(x) \rightarrow \textit{Product}(x) \wedge \textit{Exporting}(x)$ is plausible, here "Exporting", according to the Economy ontology, is employed in international trade, because of the rules $\textit{Exporting}(x) \rightarrow \textit{ChangeOfPossession}(x)$ and $\textit{Exporting}(x) \rightarrow \textit{FinancialTransaction}(x)$.

## 5 Conclusion

In this paper, we proposed a method for predicting plausible missing rules from a given ontology (or rule base) based on Graph Convolutional Networks (GCNs). To this end, we introduced an encoding of the ontology as a graph. We then introduced a GCN model that can take advantage of this graph encoding to predict rules in a more faithful way than existing methods. This is essentially due to the fact that the GCN model is able to derive structural features from the rule base, to learn much richer representations of predicates than those that are used in existing approaches.

The problem considered in this paper is not yet as mature as related topics such as knowledge graph completion, and accordingly there are still several important and interesting avenues for future work. One natural extension of our current approach would be to use a joint prediction framework, which would ensure that the collection of rules

predicted by the model is consistent with the given rule base. Essentially, such an approach would be able to use the requirement that the set of rules needs to be logically consistent as a kind of additional supervision signal. More generally, there is a clear benefit in developing methods that can integrate induction (in the sense of predicting plausible rules) and deduction in a tighter way. In terms of the technical details of our GCN model, one area that could be improved is that the parameters which are learned for each of the binary templates are currently independent from each other, which can lead to overfitting, given the small number of instances of many templates. As a possible alternative, the edge labels could be replaced by a low rank approximation of the current binary vectors.

## Acknowledgements

Steven Schockaert was supported by ERC Starting Grant 637277. Zied Bouraoui was supported by CNRS PEPS INS2I MODERN.

## References

1. Alfarone, D., Davis, J.: Unsupervised learning of an IS-A taxonomy from a limited domain-specific corpus. In: Proc. IJCAI. pp. 1434–1441 (2015)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)
3. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Proc. IJCAI. vol. 7, pp. 230–235 (2007)
4. Baget, J., Leclère, M., Mugnier, M., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. **175**(9-10), 1620–1654 (2011). https://doi.org/10.1016/j.artint.2011.03.002, http://dx.doi.org/10.1016/j.artint.2011.03.002
5. Beltagy, I., Chau, C., Boleda, G., Garrette, D., Erk, K., Mooney, R.: Montague meets Markov: Deep semantics with probabilistic logical form. In: Proceedings of *SEM13. pp. 11–21 (2013)
6. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Proceedings of the 6th International Semantic Web Conference. pp. 58–71 (2007)
7. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proc. NIPS, pp. 2787–2795 (2013)
8. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
9. Bouraoui, Z., Jameel, S., Schockaert, S.: Inductive reasoning about ontologies using conceptual spaces. In: Proc. AAAI. pp. 4364–4370 (2017)
10. Bouraoui, Z., Schockaert, S.: Automated rule base completion as bayesian concept induction. In: Proceedings of the Thirty-third AAAI Conference on Artificial Intelligence, January 27 – February 1, 2019, Honolulu, Hawaii, USA. (2019)
11. Bühmann, L., Lehmann, J., Westphal, P.: Dl-learner—a framework for inductive learning on the semantic web. Journal of Web Semantics **39**, 15–24 (2016)
12. Camacho-Collados, J., Pilehvar, M.T., Navigli, R.: Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. Artificial Intelligence **240**, 36–64 (2016)

13. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in neural information processing systems. pp. 2224–2232 (2015)
14. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1263–1272. JMLR. org (2017)
15. Grover, A., Zweig, A., Ermon, S.: Graphite: Iterative generative modeling of graphs. arXiv preprint arXiv:1803.10459 (2018)
16. Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L.: Knowledge graph embedding with iterative guidance from soft rules. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
17. Hamaguchi, T., Oiwa, H., Shimbo, M., Matsumoto, Y.: Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. arXiv preprint arXiv:1706.05674 (2017)
18. Hill, F., Cho, K., Korhonen, A.: Learning distributed representations of sentences from unlabelled data. In: Proc. NAACL-HLT. pp. 1367–1377 (2016)
19. Horrocks, I.: Ontologies and the semantic web. Commun. ACM **51**(12), 58–67 (2008). https://doi.org/10.1145/1409360.1409377, https://doi.org/10.1145/1409360.1409377
20. Jameel, S., Bouraoui, Z., Schockaert, S.: MEmbER: Max-margin based embeddings for entity retrieval. In: Proc. SIGIR. pp. 783–792 (2017)
21. Jameel, S., Schockaert, S.: Entity embeddings with conceptual subspaces as a basis for plausible reasoning. In: ECAI. pp. 1353–1361 (2016)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
23. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
24. Kok, S., Domingos, P.: Statistical predicate invention. In: Proc. ICML. pp. 433–440 (2007)
25. Kozareva, Z., Hovy, E.: A semi-supervised method to learn and construct taxonomies using the web. In: Proc. EMNLP. pp. 1110–1118 (2010)
26. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proc. EMNLP. pp. 529–539 (2011)
27. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 705–714 (2015)
28. Medina, J., Ojeda-Aciego, M., Vojtáš, P.: Similarity-based unification: a multi-adjoint approach. Fuzzy sets and systems **146**, 43–62 (2004)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems. pp. 3111–3119 (2013)
30. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proc. ACL. pp. 1003–1011 (2009)
31. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning **100**(1), 49–73 (2015)
32. Neelakantan, A., Chang, M.: Inferring missing entity type instances for knowledge base completion: New dataset and methods. In: Proc. NAACL. pp. 515–525 (2015)
33. Qian, W., Fu, C., Zhu, Y., Cai, D., He, X.: Translating embeddings for knowledge graph completion with relation attention mechanism. In: IJCAI. pp. 4286–4292 (2018)
34. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Proc. ECML/PKDD. pp. 148–163 (2010)
35. Riedel, S., Yao, L., McCallum, A., Marlin, B.M.: Relation extraction with matrix factorization and universal schemas. In: Proc. HLT-NAACL. pp. 74–84 (2013)

36. Rocktäschel, T., Riedel, S.: Learning knowledge base inference with neural theorem provers. In: Proceedings of the 5th Workshop on Automated Knowledge Base Construction. pp. 45–50 (2016)
37. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: Proc. NIPS. pp. 3791–3803 (2017)
38. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. pp. 593–607. Springer (2018)
39. Schockaert, S., Prade, H.: Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces. Artif.Intell **202**, 86–131 (2013)
40. Sourek, G., Manandhar, S., Zelezný, F., Schockaert, S., Kuzelka, O.: Learning predictive categories using lifted relational neural networks. In: Proc. ILP. pp. 108–119 (2016)
41. Speer, R., Havasi, C., Lieberman, H.: Analogyspace: reducing the dimensionality of common sense knowledge. In: Proc. AAAI. pp. 548–553 (2008)
42. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proc. of EMNLP-15. pp. 1499–1509 (2015)
43. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: Proc. ICML. pp. 2071–2080 (2016)
44. Völker, J., Niepert, M.: Statistical schema induction. In: Proc. ESWC. pp. 124–138 (2011)
45. West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., Lin, D.: Knowledge base completion via search-based question answering. In: Proc. WWW. pp. 515–526 (2014)
46. Xiao, H., Huang, M., Meng, L., Zhu, X.: Ssp: Semantic space projection for knowledge graph embedding with text descriptions. In: Proc. AAAI. vol. 17, pp. 3104–3110 (2017)
47. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Proc. of AAAI. pp. 2659–2665 (2016)
48. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Proc. of ICLR-15 (2015)
49. Zhong, H., Zhang, J., Wang, Z., Wan, H., Chen, Z.: Aligning knowledge and text embeddings by entity descriptions. In: EMNLP. pp. 267–272 (2015)