Embedding Models for Knowledge Graphs Induced by Clusters of Relations and Background Knowledge

Claudia d'Amato[1,2(✉)], Nicola Flavio Quatraro[1], and Nicola Fanizzi[1,2]

[1] Dipartimento di Informatica – Università degli Studi di Bari Aldo Moro, Bari, Italy
n.quatraro@studenti.uniba.it
[2] Centro Interdipartimentale Logica e Applicazioni – Università degli Studi di Bari Aldo Moro, Bari, Italy
{claudia.damato,nicola.fanizzi}@uniba.it

Abstract. Embedding models have been successfully exploited for predictive tasks on Knowledge Graphs (KGs). We propose TRANSROWL-HRS, which aims at making KG embeddings more semantically aware by exploiting the intended semantics in the KG. The method exploits schema axioms to encode knowledge that is observed as well as derived by reasoning. More knowledge is further exploited by relying on a successive hierarchical clustering process applied to relations, to make use of the several semantic meanings that the very same relation may have. An experimental evaluation on *link prediction* and *triple classification* tasks proves the improvement yielded by the proposed approach (coupled with different optimizers) compared to some baseline models.

1 Introduction

Knowledge Graphs (KGs) [9] are becoming increasingly important in various research and enterprise contexts. Several examples of large KGs are available, spanning from enterprise products, such as those built by Google and Amazon, to other open initiatives such as the well known DBpedia, Wikidata and YAGO. However, it is well known that KGs tend to suffer of two major problems: incompleteness and noise (e.g., see [9]). Thus, research on *knowledge graph refinement*, aiming at tackling these issues [17], has been investigated. Two tasks have gained a major attention: *Link Prediction*, focusing on predicting missing links between entities, and *Triple Classification*, that consists in assessing the correctness of a statement with respect to a KG.

Due to the need for scalable solutions, *embedding models* [3] have been widely considered, having proven their effectiveness even with very large KGs. They encode the data graph into an optimal low-dimensional space in which *graph structural information* and *graph properties* are preserved as much as possible. The various methods differ in their main building blocks [11]: the *representation space* (e.g. point-wise, complex, discrete, Gaussian, manifolds), the *encoding model* (e.g. linear, factorization, neural models) and the *scoring function* (that can be based on distance, energy, semantic matching or other criteria). The objective consists in learning embeddings such that the score of valid (positive) triples is much higher than the score of invalid triples,

N. Katzouris and A. Artikis (Eds.): ILP 2021, LNAI 13191, pp. 1–16, 2022.
https://doi.org/10.1007/978-3-030-97454-1_1

regarded as a sort of negative examples. However, KGs mostly encode positive assertions (examples) whilst negative constraints are more rarely found [1]. As positive-only learning settings may be tricky and prone to over-generalization, negative examples (invalid triples) have to be sought for, either by randomly *corrupting* true/observed triples or having them derived from additional special assumptions, such as the *local-closed world assumption*, which may conflict with the intended semantics of the data collection. In both cases, incorrect negative information may be generated and then used for training the embedding models. Hence alternative solutions are being investigated [1].

Even more so, existing learning methods often overlook the rich prior knowledge that already comes with the KGs, and is expressed through schema-level representations (ontologies) which call for *semantic embedding methods*, as argued in [4]. Some proposals for exploiting external *background knowledge* (BK) have been made, but often unnaturally resorting to external representations (e.g. Datalog clauses or fuzzy rules). Adopting a BK expressed as axioms in rich representations like RDFS and OWL new models have recently been proposed (e.g., see [15]).

In this work, we aim at extending a promising recent model, TRANSROWL [5], adopting a finer-grained treatment of the relationships in the BK. Specifically, inspired by the model dubbed HRS (*Hierarchical Relation Structure*) [21], the aim is to further empower the solution by taking into account the several sub-interpretations that each relationship may have. This has been shown to improve base models like TRANSE [2] and TRANSH [18]. We focus on the application of this idea to enhance TRANSROWL, and its base model TRANSR [12], with the final goal of setting up a general framework for KG embedding empowered by a larger usage of the available BK. The resulting variant is dubbed TRANSROWL-HRS. The proposed solution takes also advantage of an informed corruption process that leverages on reasoning and is able to limit the amount of false negatives introduced by an unconstrained random corruption process.

It is important to remark that, in principle, the proposed approach could be applied to more complex KG embedding methods. In this work we intended to show the feasibility of the approach, starting with well established models before moving on towards more sophisticated ones, which would require an additional formalization. The proposed solution is actually able to improve the effectiveness compared to the original models as proved through an experimentation on standard datasets focusing on link prediction and triple classification tasks.

The paper is organized as follows: basics on KG embedding models that are functional to our method definition are presented in Sect. 2; the formalization of our solution is illustrated in Sect. 3; the experimental evaluation is provided in Sect. 4; related work is discussed in Sect. 5; conclusions and future research directions are delineated in Sect. 6.

2 Basics on KGs and Embedding Models

In what follows, we shall assume familiarity with the standard representation and reasoning frameworks RDF, RDFS and OWL, as we will consider graphs made up of triples $\langle s, p, o \rangle$ of *RDF terms*, respectively the *subject*, the *predicate*, and the *object*,

s.t. $s \in U \cup B$ where U is a set of URIs and B is a set of blank nodes, $p \in U$ and $o \in U \cup B \cup L$ where L stands for a set of *literals*. Given an RDF graph G, we denote with $\mathcal{E}_G$ the set of all entities occurring as subjects or objects in G, and with $\mathcal{R}_G$ the set of all predicates occurring in G.

Several KG embedding models have been proposed [3], as learned *distributed representations* (or *embeddings*) for each entity and predicate in a KG, considering different representation spaces (e.g. point-wise, complex, discrete, Gaussian, manifold). We will adopt vectors of real numbers: given a KG G, each entity $x \in \mathcal{E}_G$ is represented by a continuous *embedding vector* $\mathbf{e}_x \in \mathbb{R}^k$, where $k \in \mathbb{N}$ is user-defined. Similarly, each predicate $p \in \mathcal{R}_G$ is associated to a *scoring function* $f_p : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$. For each pair of entities $s, o \in \mathcal{E}_G$, the score $f_p(\mathbf{e}_s, \mathbf{e}_o)$ measures the *confidence* that the statement $\langle s, p, o \rangle$ holds true.

In the following, we recall the basics of models that will be successively extended.

TransR: In this model each entity $x \in \mathcal{E}_G$ is represented by an embedding vector $\mathbf{e}_x \in \mathbb{R}^k$, and each predicate $p \in \mathcal{R}_G$ is represented by a *rotation operation* $\mathbf{e}_p \in \mathbb{R}^k$. The score of a triple $\langle s, p, o \rangle$ is given by the similarity (negative L_1 or L_2 distance) of the rotated subject embedding to the object embedding $\mathbf{e}_o$ preliminarily projected into the d-dimensional space of the relational embeddings via a suitable matrix $\mathbf{M} \in \mathbb{R}^{k \times d}$:

$$f'_p(\mathbf{e}_s, \mathbf{e}_o) = -\|(\mathbf{M}\mathbf{e}_s + \mathbf{e}_p) - \mathbf{M}\mathbf{e}_o\|_{\{1,2\}}. \tag{1}$$

The learning method is a *stochastic optimization process* that iteratively updates the distributed representations by increasing the score of the observed triples in G, contained in a given set Δ, while decreasing the score of unobserved triples in Δ', standing as negative examples. The latter are generated by means of a random *corruption process* which replaces either the subject or the object of observed triples with other entities in G. Formally, given $t \in \Delta$ and the set $\mathcal{C}_G(t)$ of all triples derived by corrupting t:

$$\Delta' = \bigcup_{\langle s,p,o \rangle \in \Delta} \mathcal{C}_G(\langle s, p, o \rangle) = \bigcup_{\langle s,p,o \rangle \in \Delta} \{\langle \tilde{s}, p, o \rangle \mid \tilde{s} \in \mathcal{E}_G\} \cup \{\langle s, p, \tilde{o} \rangle \mid \tilde{o} \in \mathcal{E}_G\}.$$

The embedding of all entities and predicates in G is learned by minimizing a *margin-based ranking loss*. Formally, let $\theta \in \Theta$ denote a configuration for all entity and predicate embeddings, i.e. the *model parameters* in the parameters space Θ. The optimal model parameters $\hat{\theta} \in \Theta$ are learned by solving the following constrained optimization problem with a specific loss functional:

$$\min_{\theta \in \Theta} \sum_{\substack{\langle s,p,o \rangle \in \Delta \\ \langle \tilde{s},p,\tilde{o} \rangle \in \Delta'}} \left[\gamma + f'_p(\mathbf{e}_s, \mathbf{e}_o) - f'_p(\mathbf{e}_{\tilde{s}}, \mathbf{e}_{\tilde{o}})\right]_+ \text{ subject to: } \|\mathbf{e}_x\| = 1, \forall x \in \mathcal{E}_G \tag{2}$$

where $[c]_+ = \max\{0, c\}$, and $\gamma \geq 0$ is the *margin*. It enforces higher scores for observed triples w.r.t. unobserved triples, with constraints preventing trivial solutions.

TransROWL: This model [5] extends TransR by injecting more BK in the learning process. This is obtained by introducing constraints, corresponding to BK axioms, that influence the way embedding vectors are learned. Corrupted triples, that represent negative instances, are generated by a reasoner (exploiting the axioms on domain, range, disjointWith, functionalProperty). The resulting loss function is reported below:

$$L = \sum_{\substack{\langle h,r,t\rangle \in \Delta \\ \langle h',r,t'\rangle \in \Delta'}} [\gamma + f'_r(h,t) - f'_r(h',t')]_+ + \lambda_1 \sum_{\substack{\langle t,q,h\rangle \in \Delta_{\mathsf{inverseOf}} \\ \langle t',q,h'\rangle \in \Delta'_{\mathsf{inverseOf}}}} [\gamma + f'_q(t,h) - f'_q(t',h')]_+$$
$$+ \lambda_2 \sum_{\substack{\langle h,s,t\rangle \in \Delta_{\mathsf{equivProperty}} \\ \langle h',s,t'\rangle \in \Delta'_{\mathsf{equivProperty}}}} [\gamma + f'_s(h,t) - f'_s(h',t')]_+ + \lambda_3 \sum_{\substack{\langle h,\mathsf{typeOf},l\rangle \in \Delta \cup \Delta_{\mathsf{equivClass}} \\ \langle h',\mathsf{typeOf},l'\rangle \in \Delta' \cup \Delta'_{\mathsf{equivClass}}}} [\gamma + f'_{\mathsf{typeOf}}(h,l) - f'_{\mathsf{typeOf}}(h',l')]_+$$
$$+ \lambda_4 \sum_{\substack{\langle t,\mathsf{subClassOf},p\rangle \in \Delta_{\mathsf{subClass}} \\ \langle t',\mathsf{subClassOf},p'\rangle \in \Delta'_{\mathsf{subClass}}}} [(\gamma - \beta) + f'(t,p) - f'(t',p')]_+$$

where $q \equiv r^-$, $s \equiv r$ (properties), $l \equiv t$ and $t \sqsubseteq p$ (classes) and the triple sets Δ_π, $\pi \in \{\mathsf{inverseOf}, \mathsf{equivProperty}, \mathsf{equivClass}, \mathsf{subClass}\}$, contain additional triples generated by *reasoning* on these properties and $f'(h,p) = \|\mathbf{e}_h - \mathbf{e}_p\|$ considering the embedding vectors coming from TRANSR. The different formulation for the case of subClassOf is motivated by the fact that it encodes the additional constraint $f'_{\mathsf{typeOf}}(e,p) > f'_{\mathsf{typeOf}}(e,h)$ where e is an instance, h subClassOf p and $f'_{\mathsf{typeOf}}(e,p)$ is as for the original formulation in Eq. 1. A further term, β, is required to determine the direction of the inequality to be obtained for the score values associated to subclass entities (one w.r.t. the other). The parameters $\lambda_1, \ldots, \lambda_4$ weigh the influence of each term during the learning phase.

An alternative formulation of the model, dubbed TRANSROWL^R, has been also proposed in [5], grounded on the exploitation of *axiom-based regularization*, in which the constraints that represent the related properties of the entities and relations are explicitly expressed in the loss function:

$$L = \sum_{\substack{\langle h,r,t\rangle \in \Delta \\ \langle h',r',t'\rangle \in \Delta'}} [\gamma + f'_r(h,t) - f'_r(h',t')]_+ + \lambda_1 \sum_{r \equiv q^- \in \mathcal{T}_{\mathsf{inverseOf}}} \|r + q\| + \lambda_2 \sum_{r \equiv q^- \in \mathcal{T}_{\mathsf{inverseOf}}} \|\mathbf{M}_r - \mathbf{M}_q\|$$
$$+ \lambda_3 \sum_{r \equiv p \in \mathcal{T}_{\mathsf{equivProp}}} \|r - p\| + \lambda_4 \sum_{r \equiv p \in \mathcal{T}_{\mathsf{equivProp}}} \|\mathbf{M}_r - \mathbf{M}_p\|$$
$$+ \lambda_5 \sum_{e' \equiv e'' \in \mathcal{T}_{\mathsf{equivClass}}} \|e' - e''\| + \lambda_6 \sum_{s' \subseteq s'' \in \mathcal{T}_{\mathsf{subClass}}} \|1 - \beta - (s' - s'')\|$$

where $\mathcal{T}_{\mathsf{inverseOf}} = \{r_1 \equiv q_1^-, r_2 \equiv q_2^-, ..., r_n \equiv q_n^-\}$, and $\mathcal{T}_{\mathsf{equivProp}} = \{r_1 \equiv p_1, r_2 \equiv p_2, ..., r_n \equiv p_n\}$ are, resp., the sets of inverse and equivalent properties, while $\mathcal{T}_{\mathsf{equivClass}} = \{e'_1 \equiv e''_1, e'_2 \equiv e''_2, ..., e'_n \equiv e''_n\}$ and $\mathcal{T}_{\mathsf{subClass}} = \{s'_1 \sqsubseteq s''_1, s'_2 \sqsubseteq s''_2, ..., s'_n \sqsubseteq s''_n\}$ are, resp., the sets of equivalent classes and subclasses. Parameters $\lambda_1, \ldots, \lambda_6$ determine the weights associated to each constraint. An additional term is required for inverseOf and equivProp triples to favor the equality of their projection matrices. This is for having the same scores to the triples in their respective sets, that is the score of $\langle h,r,t\rangle$ and $\langle h,p,t\rangle$ should be equal if axiom $\langle r, \mathsf{equivProp}, p\rangle$ holds.

3 TRANSROWL-HRS

The proposed model TRANSROWL-HRS aims at making KG embeddings more semantically aware. This is obtained by extending TRANSROWL so to take into account the subtleties in the meaning of each relation. We first motivate the choice for this research direction, hence we present the formalization and discuss on its training phase.

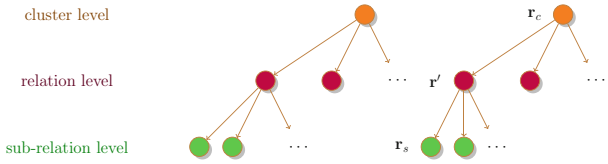

Fig. 1. Hierarchical structure for the relations in HRS [21]

3.1 TRANSROWL and Relation Hierarchies

Most translational models, including TRANSR, do not have substantial variation in the representation of the relations, whereas each relation may be associated to different meanings [19]. To this purpose, a specific *Hierarchical Relation Structure* (HRS) has been proposed [21] as a more complex model for the semantics of the relations, that was proven to improve basic models like TRANSE and TRANSH. Following this idea, we propose TRANSROWL-HRS extending TRANSROWL with a finer grained usage of the BK via a more complex treatment of the embeddings for relationships. The focus is on a three-level hierarchical structure, as also depicted in Fig. 1:

- *Relation clusters*: sets of semantically similar properties. For example the similarity of read and study may be derived from their sharing the domain and range, resp. Person and Book (or some of their super-classes). The aim is training semantically similar relations collectively, so that properties with fewer triples available would benefit from being fitted together with others with more triples in the KG;
- *Relations*: standard notion of relation as a predicate connecting subjects to objects;
- *Sub-relations*: given any relation, *sub-relations* can be defined in terms of its various interpretations. As an example, one may consider partOf: it may assume different meanings that refer, resp., to the mereological relation or to the association based on the geographic location, as in the triples $\langle \mathsf{CPU}, \mathsf{partOf}_1, \mathsf{Computer} \rangle$ and $\langle \mathsf{Vatican}, \mathsf{partOf}_2, \mathsf{Italy} \rangle$.

3.2 Model Formalization

Distinctions in the partitioning levels are reflected in the embeddings as follows: given a relation r and its embedding $\mathbf{r} \in \mathbb{R}^d$, the latter is defined by the linear combination of three further embedding vectors, namely $\mathbf{r} = \mathbf{r}_c + \mathbf{r}' + \mathbf{r}_s$ where $\mathbf{r}_c \in \mathbb{R}^d$ represents the cluster r belongs to, $\mathbf{r}' \in \mathbb{R}^d$ relation embedding for r and $\mathbf{r}_s \in \mathbb{R}^d$ representing the sub-relation related to the considered triple. Clusters of relations and sub-relations are determined through a clustering algorithm on the grounds of the metric of the embedding space for the relations, indicated with $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_{|\mathcal{R}|}$. Specifically:

- *Clusters of relations*: The set of n_c clusters of relations, indicated with $\mathcal{C} = \{\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_{n_c}\}$, is found by a suitable algorithm, such as K-MEANS, running on the relations vectors $\mathbf{r}_i$ initialized by TRANSE.

– *Sub-relations:* The sub-relations, whose instances are indicated with $\hat{\mathbf{r}}_i$, are sets representing semantic nuances that may characterize each relation r_i w.r.t. its context, i.e. the triples where it appears. Given a generic triple $\langle h, r, t\rangle$, the corresponding sub-relation results from $\hat{\mathbf{r}} = \mathbf{t} - \mathbf{h}$ (with embeddings computed by TRANSE). So, for each $\langle h_i, r, t_i\rangle$, the different sub-relations $\hat{\mathbf{r}}_i$ associated to $\mathbf{r}$ can be computed. The grouping $\mathcal{S}_1^r, \mathcal{S}_2^r, \mathcal{S}_3^r, \ldots, \mathcal{S}_{n_r}^r$ of sub-relations is organized through the clustering algorithm, where n_r is the number of sub-relations of r.

Moving from the HRS score function: $f_r(h,t) = \|\mathbf{h} + \mathbf{r}_c + \mathbf{r}' + \mathbf{r}_s - \mathbf{t}\|_n$ the TRANSROWL-HRS score function is obtained by replacing the embedding vector for the relation with the linear combinations of the terms coming from the hierarchical structure, that is:

$$f'_r(h,t) = \|\mathbf{h}_r + \mathbf{r}_c + \mathbf{r}' + \mathbf{r}_s - \mathbf{t}_r\|_n \tag{3}$$

$$f'(h,t) = \|\mathbf{h}_r - \mathbf{t}_r\|_n \tag{4}$$

where n indicates the norm (L_1 or L_2) and $f'(h,t)$ is the score function that considers the subClassOf-axioms. Similarly to TRANSR, the projections of h and t to the vector space of r are computed via the matrix $\mathbf{M}_r$: $\mathbf{h}_r = \mathbf{h}\mathbf{M}_r$ and $\mathbf{t}_r = \mathbf{t}\mathbf{M}_r$ (see Sect. 2).

3.3 Training the Model

The formalization of the TRANSROWL-HRS loss function moves from the one defined for HRS [21] whilst requiring additional formulation due to different base models adopted (TRANSROWL for TRANSROWL-HRS and TRANSE for the case of HRS).

In HRS the adopted loss function is a combination of two terms: $L_{\text{Tot}} = L_{\text{B}} + L_{\text{HRS}}$, where L_{B} is the loss of the base-model HRS is applied to (that is TRANSE), taking into account that it must also consider the clusters the relations belong to, indicated by $\mathcal{C} = \{\text{C}_1, \text{C}_2, ..., \text{C}_{n_c}\}$. The term L_{HRS} manages the influence that each embedding vector among $\mathbf{r}_c$, $\mathbf{r}'$, $\mathbf{r}_e$ has in the definition of the embedding associated to $\mathbf{r}$. This is formalized considering the linear combination of each group of embeddings in the hierarchical structure of the relations, with a different specific weight:

$$L_{\text{HRS}} = \lambda_c \sum_{\mathbf{r}_c \in \mathcal{C}} \|\mathbf{r}_c\|_2^2 + \lambda_r \sum_{\mathbf{r}' \in \mathcal{R}} \|\mathbf{r}'\|_2^2 + \lambda_s \sum_{\mathbf{r}_s \in \mathcal{S}} \|\mathbf{r}_s\|_2^2 \tag{5}$$

where $\mathcal{C} = \{\text{C}_1, \text{C}_2, \ldots, \text{C}_{n_c}\}$ is the set of clusters, $\mathcal{S} = \{\mathcal{S}_1^r, \mathcal{S}_2^r, \ldots, \mathcal{S}_{n_r}^r \mid r \in \mathcal{R}\}$ is the set of sub-relations for each r, and λ_c, λ_r and λ_s are regularization parameters.

As for the case of TRANSROWL-HRS, the L_{HRS} term remains the same as above, whilst for the L_B term, the new base-model TRANSROWL needs to be taken into account, as well as the clusters the relations belong to. Hence the formulation of L_{B} for TRANSROWL-HRS will be given by the TRANSROWL loss function modified so as to consider the clusters of relations. The formulation is the following:

$$
\begin{aligned}
L_{\mathrm{B}} = & \sum_{c=1}^{n_c} \sum_{r \in \mathsf{C}_c} \sum_{\substack{\langle h,r,t\rangle \in \Delta \\ \langle h',r',t'\rangle \in \Delta'}} [\gamma + f'_r(h,t) - f'_r(h',t')]_+ \; + \; \lambda_1 \sum_{c=1}^{n_c} \sum_{q \in \mathsf{C}_c} \sum_{\substack{\langle t,q,h\rangle \in \Delta_{\mathsf{inverseOf}} \\ \langle t',q,h'\rangle \in \Delta'_{\mathsf{inverseOf}}}} [\gamma + f'_q(t,h) - f'_q(t',h')]_+ \\
& + \lambda_2 \sum_{c=1}^{n_c} \sum_{s \in \mathsf{C}_c} \sum_{\substack{\langle h,s,t\rangle \in \Delta_{\mathsf{equivProperty}} \\ \langle h',s,t'\rangle \in \Delta'_{\mathsf{equivProperty}}}} [\gamma + f'_s(h,t) - f'_s(h',t')]_+ \\
& + \lambda_3 \sum_{c=1}^{n_c} \sum_{\mathsf{typeOf} \in \mathsf{C}_c} \sum_{\substack{\langle h,\mathsf{typeOf},l\rangle \in \Delta_{\mathsf{equivClass}} \\ \langle h',\mathsf{typeOf},l'\rangle \in \Delta'_{\mathsf{equivClass}}}} [\gamma + f'_{\mathsf{typeOf}}(h,l) - f'_{\mathsf{typeOf}}(h',l')]_+ \\
& + \lambda_4 \sum_{c=1}^{n_c} \sum_{\mathsf{typeOf} \in \mathsf{C}_c} \sum_{\substack{\langle t,\mathsf{subClassOf},p\rangle \in \Delta_{\mathsf{subClass}} \\ \langle t',\mathsf{subClassOf},p'\rangle \in \Delta'_{\mathsf{subClass}}}} [(\gamma - \beta) + f'(t,p) - f'(t',p')]_+ \qquad (6)
\end{aligned}
$$

Further variants can be applied by considering: the *top-middle* and the *middle-bottom* settings. The *top-middle* focuses exclusively on the first and second level of the hierarchical structure, that is clusters of relations and relations: $\mathbf{r} = \mathbf{r}_c + \mathbf{r}'$. This affects the formalization of the L_{HRS} term in the loss function and of the score function, which is as follows:

$$L_{\mathrm{HRS}} = \lambda_c \sum_{\mathbf{r}_c \in \mathcal{C}} \|\mathbf{r}_c\|_2^2 + \lambda_r \sum_{\mathbf{r}' \in \mathcal{R}} \|\mathbf{r}'\|_2^2 \qquad (7)$$

$$f_r(h,t) = \|\mathbf{h}_r + \mathbf{r}_c + \mathbf{r}' - \mathbf{t}_r\|_n \qquad (8)$$

The *middle-bottom* model focuses exclusively on the second and third level of the hierarchical structure $\mathbf{r} = \mathbf{r}' + \mathbf{r}_s$ (relations and sub-relations), leading to modify the formulation of the score function and L_{HRS} loss as follows:

$$L_{\mathrm{HRS}} = \lambda_r \sum_{\mathbf{r}' \in \mathcal{R}} \|\mathbf{r}'\|_2^2 + \lambda_s \sum_{\mathbf{r}_s \in \mathcal{S}} \|\mathbf{r}_s\|_2^2 \qquad (9)$$

$$f_r(h,t) = \|\mathbf{h}_r + \mathbf{r}' + \mathbf{r}_s - \mathbf{t}_r\|_n \qquad (10)$$

Following [19], TRANSROWL-HRS adopts the *top-middle* variant as it is very likely that a KG include numerous semantically similar relations, clusters of relations then, rather than a large number of sub-relations. The single variant chosen is motivated also for controlling the complexity of the model.

Algorithm 1 reports the procedure associated to the TRANSROWL-HRS model. Preliminarily, it requires the training set S, with the related set of entities $\mathcal{E}$ and relations $\mathcal{R}$, the sets of all axioms of interest A_z, A_w used, resp., to generate further triples for the training and to generate corrupted triples, the set of clusters $\mathcal{C}$ and the dimensionality hyperparameters. The embedding vectors for entities and relations are initialized by TRANSE to avoid overfitting, as suggested in [12]; moreover the vectors associated to the clusters and the projection matrices are also initialized. The main loop iterates the following steps for a fixed number of *epochs*:

- Embedding vectors are normalized to satisfy the constraints;
- A minibatch S_{bat} of size b is sampled from S, while T_{bat} is initialized with $\emptyset$;

Algorithm 1: TRANSROWL-HRS

parameters:
$S = \{\langle h, r, t\rangle\}$: training set;
$\mathcal{E}, \mathcal{R}$: entity and relation sets;
$\mathcal{A}_z$: axiom sets, with $z \in \{\mathsf{inverseOf}, \mathsf{equivProperty}, \mathsf{equivClass}, \mathsf{subClassOf}\}$;
$\mathcal{C}$: cluster set;
$\mathcal{A}_w$: axiom sets, with $w \in \{\mathsf{range}, \mathsf{domain}, \mathsf{functionalProperty}, \mathsf{disjointWith}\}$;
γ: margin;
k, m: embeddings dim. for entities and relations;
$nepoch$: number of epochs;

$\mathbf{e}, \mathbf{r}' \leftarrow$ TRANSE results $\forall r' \in \mathcal{R}, \forall e \in \mathcal{E}$; (* initialization *)
$\mathbf{r}_c \leftarrow \mathbf{0}\ \forall r_c \in \mathcal{C}$;
$\mathbf{M}_r \leftarrow \mathbf{I} \in \mathbb{R}^{k \times m}\ \forall r \in \mathcal{R}$;
while $epoch < nepoch$ **do**
  normalize $\mathbf{e}, \mathbf{r}, \mathbf{t}, \mathbf{e}\mathbf{M}_r, \mathbf{t}\mathbf{M}_r$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$;
  $S_{\text{bat}} \leftarrow$ sample(S,b);
  $T_{\text{bat}} \leftarrow \emptyset$;
  for $\langle h, r, t\rangle \in S_{\text{bat}}$ **do**
    $\langle h', r, t'\rangle \leftarrow$ corrupt($\langle h, r, t\rangle, A_w$);
    $T_{\text{bat}} \leftarrow T_{\text{bat}} \cup \{(\langle h, r, t\rangle, \langle h', r, t'\rangle)\}$;
    switch r **do**
      case $r \in \mathcal{A}_{\mathsf{inverseOf}}$ **do**
        $\langle t', q, h'\rangle \leftarrow$ corrupt($\langle t, q, h\rangle, A_w$);
        $T_{\text{bat}} \leftarrow T_{\text{bat}} \cup \{(\langle t, q, h\rangle, \langle t', q, h'\rangle)\}$;
      end
      case $r \in \mathcal{A}_{\mathsf{equivProperty}}$ **do**
        $\langle h', s, t'\rangle \leftarrow$ corrupt($\langle h, s, t\rangle, A_w$);
        $T_{\text{bat}} \leftarrow T_{\text{bat}} \cup \{(\langle h, s, t\rangle, \langle h', s, t'\rangle)\}$;
      end
      case $r = \mathsf{typeOf}$ **and** $t \in \mathcal{A}_{\mathsf{equivClass}}$ **do**
        $\langle h', r, t'\rangle \leftarrow$ corrupt($\langle h, r, t\rangle$,A_w);
        $T_{\text{bat}} \leftarrow T_{\text{bat}} \cup \{(\langle h, r, l\rangle, \langle h', r, l'\rangle)\}$;
      end
      case $r = \mathsf{typeOf}$ **and** $t \in \mathcal{A}_{\mathsf{subClassOf}}$ **do**
        $\langle t', r, p'\rangle \leftarrow$ corrupt($\langle t, \mathsf{subClassOf}, p\rangle$,$A_w$);
        $T_{\text{bat}} \leftarrow T_{\text{bat}} \cup \{(\langle t, \mathsf{subClassOf}, p\rangle, \langle t', \mathsf{subClassOf}, p'\rangle)\}$;
      end
    end
  end
  $g_t \leftarrow \sum_{((h,r,t),(h',r,t')) \in T_{\text{bat}}} \nabla L_{\text{Tot}}$; (* gradient *)
  $\Delta_t \leftarrow -\eta g_t$; (* update *)
  $epoch \leftarrow epoch + 1$;
end

- For each $\langle h, r, t\rangle \in S_{\text{bat}}$, a corrupted $\langle h', r, t'\rangle$ is produced[1] exploiting the axioms in A_w and the entities/relation in the triple; the pair of triples is added to T_{bat}.
- Analyzing r, all the applicable cases are considered (inclusively) w.r.t. the properties of the axioms in A_z, hence new pairs of positive and negative triples are generated and added to T_{bat};
- Lastly, the gradient g_t and updates are computed based on the triple pairs in T_{bat}, and the embedding parameters are updated by the optimizer of choice.

[1] A standard corruption strategy can be employed: `unif` generates negative triples by sampling a pair of entities for subject and object from $\mathcal{E}_G$, assigning uniform probabilities to the possible replacements; `bern` assigns Bernoulli distributed chances based on the type of property/mapping (1-to-1,1-to-N, N-to-N).

The algorithm terminates when the fixed number of epochs, *nepoch*, is reached.

Three variants will be considered depending on the optimizer employed.

4 Empirical Evaluation

In this section we illustrate the experimental evaluation of TransROWL-HRS compared to TransR, TransROWL, TransROWL^R as baselines. Note that TransE was only used for the embeddings initialization (see Sect. 3.3) and not in the comparison as it has been shown [5] that the considered baselines are able to outperform it.

We tested the performance of the models on the tasks of *Link Prediction*, together with *Type Prediction* (that, given typeOf-triple for a subject, verifies if the model can correctly predict a class the individual belongs to) and *Triple Classification*, i.e. the ability to classify new triples as true or false. Overall the runtimes for the various models were on average comparable, with a slight overhead required by the clustering method. Further details are publicly available in the project documentation[2].

4.1 Experiment Setup

Datasets. The models were tested on four datasets drawn from well known KGs, that have been considered for experiments in related works [6, 13].

- *DBpedia*: data extracted from Wikipedia. It includes 320 classes and 1650 properties. We considered two datasets extracted to ensure axioms to test the models, namely axioms on domain, range, disjointWith, functionalProperty, equivalentClass, equivalentProperty, inverseOf, subClassOf, in the two variants: *DBpedia100K*[3] [6], containing about 100K entities, 321 relations in 600K triples; *DBpedia15K*[4] [13], containing about 12.8K entities and 278 relations in 180K triples.
- *DBPediaYAGO*: YAGO[5] is a KG with knowledge coming from different sources e.g. *WordNet*, *GeoNames*, *Wikipedia*, including 350K+ classes, 10M entities, 120M assertions [17]. It has been used to extend *DBpedia15K*, resulting in *DBPediaYAGO* having about 290K triples, with 88K entities and 316 relations.
- *NELL*: The dataset[6] comes from a knowledge extraction system from corpora of Web pages. The resulting KG amounts to 2.810K+ assertions regarding 1.186 different relations and categories. We considered a fragment of NELL2RDF-vanilla[7] that does not contain all of the properties that can be exploited by the proposed model. The considered dataset is made up of about 150K triples, with 272 properties and 68K entities. The aim was to have a dataset with a limited set of exploitable properties, namely subClassOf, inverseOf, functionalProperty, disjointWith, range and domain. The abundance of subClassOf-triples and limited number of typeOf-triples, is meant to test the ability to compensate this partial incompleteness.

[2] https://github.com/Keehl-Mihael/TransROWL-HRS.
[3] https://github.com/iieir-km/ComplEx-NNE_AER/tree/master/datasets/DB100K.
[4] https://github.com/nle-ml/mmkb/tree/master/DB15K.
[5] https://yago-knowledge.org/.
[6] http://rtw.ml.cmu.edu/rtw/.
[7] http://nell-ld.telecom-st-etienne.fr/.

Each dataset was randomly partitioned into *training*, *validation* and *test* sets by selecting 70%, 10%, 20% of the triples.

Parameter Settings. All models were set up along the same procedure and parameter values, consistently with the experiments illustrated in [2,12,21]: learning rate: 0.001; minibatch dimension: 50; entity/relation vector dimension = 100; epochs: 1000. This choice is motivated by the fact that our first aim is to verify the possible improvements of the proposed solution over the basic models when exactly the same conditions, including the parameter values, apply. The `bern` strategy for the triple corruption phase was adopted, as this choice led to a better performance compared to the `unif` strategy in previous experimental evaluations of this class of models [12,18].

As for the hyperparameters λ_i in the loss functions, the following values have been found: as for TRANSROWL, inverseOf $\lambda_1 = 1$; equivalentProperty $\lambda_2 = 1$; equivalentClass $\lambda_3 = 0.1$; subClassOf $\lambda_4 = 0.01$; for TRANSROWL^R: $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = \lambda_6 = 0.1$; as for TRANSROWL-HRS: $\lambda_c = 0.00001$; $\lambda_r = 0.0001$.

Three TRANSROWL variants are considered depending on the optimizer employed: TRANSROWL-HRS (that uses SGD), TRANSROWL-HRS Momentum and TRANSROWL-HRS AdaGrad.

4.2 Link Prediction

Following the standard procedures we focus on predicting individuals in given incomplete triples, specifically triples $\langle h, r, t\rangle$, with $h, t \in \mathcal{E}_G$ and $r \in \mathcal{R}_G$, corresponding to the patterns $\langle ?, r, t\rangle, \langle h, r, ?\rangle$. The typical metrics considered for this task are *Mean Rank* (the lower the better) and *H@10* (the higher the better). The *Raw* and *Filtered* variants are considered, the latter filtering off the corrupted triples generated for training the model. For a more specific insight, we measured separately the performance considering all properties but typeOf, and then typeOf only, which allows to focus separately on *Type Prediction* problems with the classes in the KGs. As mentioned in Sect. 3.3, following the approach adopted by the baseline methods, the embeddings were initialized by a first run of TRANSE, and the models were trained for a fixed number of epochs.

The complete outcomes of the link prediction experiments are illustrated in Table 1. Considering preliminarily the link prediction problems (no typeOf), we found that TRANSROWL-HRS adopting the AdaGrad optimizer had the best performance on almost all of the datasets and measures with some exceptions. As regards the case of *DBpediaYAGO* dataset, the TRANSR was able to do slightly better, in terms of the H@10 metric, than the new model which is a very close runner-up. A more difficult testbed was represented by the *NELL* dataset on which TRANSR had a slightly better performance also in terms of MR. The reason for this decay was due to the limited number of properties that can be exploited by the proposed model (as discussed in Sect. 4.1) as well as a limited number of relations compared to the much larger number of entities.

As for the results on the type prediction problems (typeOf columns), TRANSROWL-HRS with AdaGrad proved as the best model on *DBpediaYAGO* and *NELL* (in terms of H@10), whereas TRANSROWL-HRS with Momentum showed a better performance on *DBpedia15K*. This proves our intuition (see Sect. 4.1) that the

Table 1. Link Prediction results (MR = Mean Rank and H@10 = Hits@10)

	DBpedia15K							
	no typeOf				typeOf			
	MR	H@10	MR	H@10	MR	H@10	MR	H@10
model	(raw)	(raw)	(flt.)	(flt.)	(raw)	(raw)	(flt.)	(flt.)
TransR	600.12	60.67	586.83	63.57	504.13	85.01	13.96	95.50
TransROWL	606.73	60.59	593.45	63.48	484.04	**85.18**	**13.53**	96.54
TransROWL^R	607.43	60.71	594.13	63.65	497.40	85.12	16.50	96.24
TransROWL-HRS	600.08	60.62	586.83	63.43	485.08	85.17	14.96	96.61
TransROWL-HRS Momentum	605.23	60.48	591.95	63.40	**472.22**	85.12	14.59	**96.85**
TransROWL-HRS AdaGrad	**579.47**	**61.18**	**566.21**	**64.00**	506.06	85.01	25.72	94.77
	DBpedia100K							
	no typeOf				typeOf			
	MR	H@10	MR	H@10	MR	H@10	MR	H@10
model	(raw)	(raw)	(flt.)	(flt.)	(raw)	(raw)	(flt.)	(flt.)
TransR	2142.10	53.17	2112.42	55.96	**1957.42**	92.04	**1480.26**	92.25
TransROWL	2147.56	53.24	2117.87	56.03	1961.75	**92.29**	1503.87	**92.43**
TransROWL^R	2121.52	53.08	2091.81	55.95	1971.98	92.24	1511.07	**92.43**
TransROWL-HRS	2127.12	52.90	2097.58	55.62	1957.96	92.09	1503.34	92.23
TransROWL-HRS Momentum	2126.35	52.88	2096.84	55.58	1970.28	92.16	1526.95	92.31
TransROWL-HRS AdaGrad	**2087.74**	**53.47**	**2058.10**	**56.25**	1960.09	92.10	1519.92	92.23
	DBpediaYAGO							
	no typeOf				typeOf			
	MR	H@10	MR	H@10	MR	H@10	MR	H@10
model	(raw)	(raw)	(flt.)	(flt.)	(raw)	(raw)	(flt.)	(flt.)
TransR	7271.50	**44.64**	7239.09	**46.07**	844.51	81.98	348.65	88.99
TransROWL	7209.02	44.45	7176.64	45.84	868.27	82.81	373.90	91.17
TransROWL^R	7226.55	44.13	7194.21	45.52	845.42	81.71	352.16	88.77
TransROWL-HRS	7189.25	44.42	7156.92	45.68	705.91	82.96	213.18	90.98
TransROWL-HRS Momentum	7144.80	44.26	7112.47	45.59	731.13	82.82	241.69	92.59
TransROWL-HRS AdaGrad	**7104.88**	44.61	**7072.72**	45.92	**642.92**	**84.85**	**176.32**	**95.20**
	NELL							
	no typeOf				typeOf			
	MR	H@10	MR	H@10	MR	H@10	MR	H@10
model	(raw)	(raw)	(flt.)	(flt.)	(raw)	(raw)	(flt.)	(flt.)
TransR	**6891.20**	**47.40**	**6681.76**	**55.93**	2315.08	79.94	2140.16	80.50
TransROWL	7136.77	46.72	6929.10	55.40	2334.50	80.00	2161.67	80.56
TransROWL^R	7339.53	46.09	7132.22	54.15	**2310.11**	79.52	**2138.99**	80.21
TransROWL-HRS	7203.32	46.83	6996.17	55.34	2397.16	79.95	2223.64	80.44
TransROWL-HRS Momentum	7144.77	46.87	6936.06	55.29	2373.70	79.29	2201.06	79.85
TransROWL-HRS AdaGrad	7153.78	46.67	6949.22	55.26	2903.37	**80.35**	2730.42	**80.80**

abundance of subClassOf-triples in *NELL*, even if with a limited number of typeOf-triples, allows the method to compensate this partial incompleteness and improve the performances whilst this does not happen for the more general link prediction problem (results analyzed above) where, similarly to TransROWL and TransROWL^R, TransROWL-HRS resulted to suffer more of the missing axioms in the KG that are

considered in the formalization of the model. In the case of *DBpedia100K*, TRANSR and TRANSROWL got slightly better scores, proving the new model not bing able to improve the baselines on type prediction problems on larger and complete datasets.

4.3 Triple Classification

Triple Classification focuses on discerning correct from incorrect triples. The evaluation measures the ability to predict whether a triple is positive or negative, i.e. it represents a true or false fact w.r.t. the KG. To make this decision, a threshold s_r is to be determined for each $r \in \mathcal{R}_G$, so to maximize the *False Positive Rate* (FPR), then test triples are deemed as positive when their score is greater than s_r, and negative otherwise [14, 18]. The value for s_r was estimated considering a random sample of r-triples selected from the training set. They represent the triples that the model has learned to deem as true; for each sampled triple the score value is computed and the threshold s_r is determined by the minimum value. The ability of the model to correctly classify triples is evaluated considering the thresholds obtained per single relation; this unavoidably increases the chance of predicting as true, triples that are actually false, thus it allows to better evaluate the model robustness on the classification of typeOf-triples.

Analogously to the previous experiments, the performance indices were determined separating the cases of typeOf-triples from those involving the other properties. This allows to better focus on the performance of the model on this relation between individuals and classes. The negative triples required for the tests, were generated by reasoning on range and domain axioms for the experiment excluding typeOf, while reasoning on disjointWith axioms were exploited to get false typeOf-triples. The experimental setting was analogous to the first part (see Sect. 4.1). Table 2 reports the complete results for each dataset in terms of *accuracy*, *precision*, *recall*, and *false positive rate*.

Focusing preliminarily on the experiments with non-typeOf relations, we observe a general similarity of the performance of the base and new models on the three *DBpedia*-based datasets with some difference on recall, which allows a little margin in favor of the best scoring model. The results observed for the experiments with *NELL* show a more contrasted outcome where a slightly higher recall yields a higher accuracy, whereas the better precision showed by TRANSROWL-HRS AdaGrad is also reflected in a lower FPR. Again this more sparse (incomplete) dataset turned out to be the most difficult testbed in the experiments, especially for methods relying on a rich BK.

Considering the experiments regarding typeOf, in the case of *DBpedia15K*, TRANSROWL-HRS Momentum was the most accurate one also because of a high precision and recall (TRANSROWL^R showed a slightly higher precision but also a much lower recall). TRANSROWL-HRS AdaGrad also had a high recall but precision dropped as testified by the high FPR. Difference in performance was even less sensible in the case of the *DBpedia100K*. The performance on *DBPediaYAGO* was sensibly better for the new models, especially in favor of TRANSROWL-HRS AdaGrad. This happened also for the experiments on *NELL* but mostly in favor TRANSROWL-HRS Momentum in this case. Again this dataset presented a particularly hard problems because of its incompleteness as testified by the low precision rates (and high FPR).

Table 2. Triple Classification results (Accuracy, Precision, Recall and FP Rate)

	DBpedia15K							
	no typeOf				typeOf			
model	Acc.	P	R	FPR	Acc.	P	R	FPR
TransR	**0.641**	**0.998**	**0.364**	**0.001**	0.972	0.966	0.946	0.378
TransROWL	0.631	0.997	0.347	0.002	0.962	**0.999**	0.882	**0.006**
TransROWL^R	0.628	**0.998**	0.342	**0.001**	0.981	0.969	0.972	0.523
TransROWL-HRS	0.634	0.995	0.353	0.002	0.985	0.994	0.961	0.135
TransROWL-HRS Momentum	0.628	0.997	0.342	**0.001**	**0.988**	0.997	0.966	0.074
TransROWL-HRS AdaGrad	0.629	**0.998**	0.343	**0.001**	0.977	0.954	**0.978**	0.682
	DBpedia100K							
	no typeOf				typeOf			
model	Acc.	P	R	FPR	Acc.	P	R	FPR
TransR	0.711	0.998	0.313	0.001	0.976	0.884	0.800	**0.344**
TransROWL	0.705	**0.998**	0.300	**0.001**	**0.987**	**0.940**	0.895	0.353
TransROWL^R	0.704	0.998	0.298	0.001	0.981	0.872	0.890	0.543
TransROWL-HRS	**0.714**	**0.998**	**0.320**	**0.001**	0.979	0.900	0.832	0.353
TransROWL-HRS Momentum	0.709	0.997	0.310	0.001	0.985	0.874	**0.945**	0.711
TransROWL-HRS AdaGrad	0.693	**0.998**	0.270	**0.001**	0.977	0.885	0.816	0.367
	DBpediaYAGO							
	no typeOf				typeOf			
model	Acc.	P	R	FPR	Acc.	P	R	FPR
TransR	0.644	0.964	0.300	0.016	0.844	0.946	0.247	0.018
TransROWL	**0.649**	0.968	**0.307**	0.014	0.905	0.973	0.547	0.032
TransROWL^R	0.636	**0.981**	0.277	**0.007**	0.854	0.953	0.299	0.020
TransROWL-HRS	0.643	0.966	0.296	0.015	0.873	0.967	0.386	0.021
TransROWL-HRS Momentum	0.645	0.974	0.299	0.011	0.873	0.981	0.381	**0.012**
TransROWL-HRS AdaGrad	0.647	0.975	0.302	0.011	**0.950**	**0.986**	**0.765**	0.043
	NELL							
	no typeOf				typeOf			
model	Acc.	P	R	FPR	Acc.	P	R	FPR
TransR	**0.758**	0.843	**0.636**	0.245	0.803	0.389	0.519	0.630
TransROWL	0.744	0.835	0.608	0.234	0.763	0.334	0.560	0.717
TransROWL^R	0.739	0.845	0.587	0.207	0.760	0.337	0.598	0.745
TransROWL-HRS	0.735	0.818	0.603	0.252	0.748	0.330	0.633	0.778
TransROWL-HRS Momentum	0.741	0.810	0.628	0.284	**0.823**	**0.421**	0.437	**0.516**
TransROWL-HRS AdaGrad	0.740	**0.847**	0.585	**0.202**	0.667	0.269	**0.692**	0.859

5 Related Work

The exploitation of hierarchies of relations in embedding methods has received increasing attention in the last few years, resulting as a promising approach particularly for link prediction tasks.

In [20], the *Hierarchy-Aware Knowledge Graph Embedding* (HAKE) model is proposed, showing the ability to learn complex semantic hierarchies. However, no use of the BK is considered. Similarly, in [10] a data-driven method for automatically discovering the distinct semantics associated with high-level relations in KGs and deriving an optimal number of sub-relations has been proposed, where vector embedding of entities and relations are preliminarily computed.

Additionally, various embedding approaches have been proposed that can leverage different forms of prior knowledge to learn better representations exploited for KG refinement tasks. Generally entities and relations are embedded into latent vectors with little exploitation of the rich information of the available relational structure. In [7] a method for jointly embedding KGs and logical rules has been proposed, where triples and rules are represented in a unified framework. Triples are represented as atomic formulae while rules are represented using t-norm fuzzy logics. A common loss over both representations is defined which is minimized to learn the embeddings. The specific forms of BK required, and the gap from the standard semantics of the KGs, constitute the main drawback. In [16] a solution based on adversarial training is proposed that exploits Datalog clauses to encode assumptions which are used to regularize neural link predictors. An inconsistency loss is derived that measures the degree of violation of such assumptions on a set of adversarial examples. A specific form of BK is required and a special assumption (*local CWA*) is to be made when reasoning with it. The availability of such clauses and the assumptions on their semantics represent the main limitations.

A common shortcoming of the related methods is that BK is often not embedded in a principled way. In [8], investigating the compatibility between ontological knowledge and different types of embeddings, they show that popular methods are not capable of modeling even very simple types of rules, hence they are not able to learn the underlying dependencies. Then a general framework is introduced in which relations are modeled as convex regions which exactly represent ontologies expressed by a specific form of rules, that preserve the semantics of the input ontology.

6 Conclusions and Future Work

An approach to learning embedding models has been proposed, that is based on exploiting the available prior knowledge (schema axioms) in both the training and the triple corruption process. A more complex model for the semantics of the relations is formalized as a three-level hierarchical structure for a fine-grained representation of their semantics. The resulting model TransROWL-HRS has been experimentally evaluated showing the improvements w.r.t. the baseline methods, particularly for link and type prediction tasks. Interestingly, the model was able to outperform the baseline models on almost all tasks, when missing axioms and limited typeOf assertions were available (the case of the *NELL* dataset, adopted for assessing the ability of the model to cope with challenging knowledge configurations, resulted hard for the baseline models), thus showing that the abundance of subClassOf-triples, even if with a limited number of typeOf-triples, allows the method to compensate this partial incompleteness and improve the performance.

Nevertheless, some shortcomings also emerged, particularly for the case of type prediction and triple classification tasks (the case involving all but typeOf relationships)

when more comprehensive datasets have been considered. This suggests that a more complex hierarchical structure mostly has a value added when limited axioms are available whilst it does not play a significant role when all axioms and a sufficient number of triples can be found, thus opening a valuable research direction to be pursued.

We are currently working on the application of the presented approach to more complex embedding models which could be suitable for our purposes. We also intend to extend our solution by exploiting further schema-axioms. Furthermore, we are planning to reuse the collected additional knowledge for building and providing explanations for the answers to queries obtained exploiting the embedding models.

References

1. Arnaout, H., Razniewski, S., Weikum, G.: Enriching knowledge bases with interesting negative statements. In: AKBC 2020 (2020). https://doi.org/10.24432/C5101K
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS 2013. Curran Assoc., Inc. (2013)
3. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(09), 1616–1637 (2018). https://doi.org/10.1109/TKDE.2018.2807452
4. d'Amato, C.: Machine learning for the semantic web: Lessons learnt and next research directions. Semant. Web **11**(1), 195–203 (2020). https://doi.org/10.3233/SW-200388
5. d'Amato, C., Quatraro, N.F., Fanizzi, N.: Injecting background knowledge into embedding models for predictive tasks on knowledge graphs. In: Verborgh, R., et al. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 441–457. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_26
6. Ding, B., Wang, Q., Wang, B., Guo, L.: Improving knowledge graph embedding using simple constraints. In: ACL 2018, vol. 1, pp. 110–121. ACL (2018). https://doi.org/10.18653/v1/P18-1011
7. Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L.: Jointly embedding knowledge graphs and logical rules. In: EMNLP 2016. ACL (2016). https://doi.org/10.18653/v1/D16-1019
8. Gutiérrez-Basulto, V., Schockaert, S.: From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules. In: Thielscher, M., et al. (eds.) KR 2018, pp. 379–388. AAAI Press (2018)
9. Hogan, A., et al.: Knowledge graphs (2020). arXiv:2003.02320
10. Jain, N., Krestel, R.: Learning fine-grained semantics for multi-relational data. In: ISWC 2020 Demos and Industry Tracks. CEUR Workshop Proceedings, vol. 2721, pp. 124–129. CEUR-WS.org (2020). http://ceur-ws.org/Vol-2721/paper529.pdf
11. Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: Representation, acquisition and applications (2020). arXiv:2002.00388
12. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI 2015, pp. 2181–2187. AAAI Press (2015)
13. Liu, Y., Li, H., Garcia-Duran, A., Niepert, M., Onoro-Rubio, D., Rosenblum, D.S.: MMKG: multi-modal knowledge graphs. In: Hitzler, P., et al. (eds.) ESWC 2019. LNCS, vol. 11503, pp. 459–474. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21348-0_30
14. Lv, X., Hou, L., Li, J., Liu, Z.: Differentiating concepts and instances for knowledge graph embedding. In: EMNLP 2018, pp. 1971–1979. ACL (2018). https://doi.org/10.18653/v1/D18-1222

15. Minervini, P., Costabello, L., Muñoz, E., Nováček, V., Vandenbussche, P.-Y.: Regularizing knowledge graph embeddings via equivalence and inversion axioms. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10534, pp. 668–683. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71249-9_40
16. Minervini, P., Demeester, T., Rocktäschel, T., Riedel, S.: Adversarial sets for regularising neural link predictors. In: UAI 2017. AUAI Press (2017)
17. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. Semant. Web **8**, 489–508 (2016). https://doi.org/10.3233/SW-160218
18. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI 2014, pp. 1112–1119. AAAI Press (2014)
19. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR 2015 (2015). arXiv:1412.6575
20. Zhang, Z., Cai, J., Zhang, Y., Wang, J.: Learning hierarchy-aware knowledge graph embeddings for link prediction. In: AAAI 2020, vol. 34, pp. 3065–3072 (2020). https://doi.org/10.1609/aaai.v34i03.5701
21. Zhang, Z., Zhuang, F., Qu, M., Lin, F., He, Q.: Knowledge graph embedding with hierarchical relation structure. In: EMNLP 2018, pp. 3198–3207. ACL (2018). https://doi.org/10.18653/v1/d18-1358