A Hybrid Self-Correcting Approach for Embedding Ontology-based Knowledge Graphs*

Abstract

One significant challenge in Knowledge Graph (KG) *Embedding* is the generation of *negative triples*. Negative triples are essential as they enable a training model to distinguish between relationships that exist within the KG and those that do not. In this paper, we propose TransHySeCo approach: a Hybrid and Self-Correcting approach for embedding knowledge graphs. TransHySeCo is based on a hybrid training using both the domain semantics provided in the ontology related to the KG and the topology underlying the graph structure. Moreover, it is self-correcting. It generates new negative triples by leveraging the embeddings from previous training iterations and the *(quasi-)true negatives* obtained with the *ontology-based negative generation* method proposed in this paper. The self-correction terminates when no new *(quasi-)true negative* triple is generated. To evaluate TransHySeCo, we conducted experiments on different benchmark datasets and assessed the embeddings' effectiveness for the link prediction task. The results show that TransHySeCo provides KG embeddings of promising quality for link prediction.

CCS Concepts

• **Computing methodologies** → **Machine learning approaches**; • **Theory of computation** → **Description logics**; **Automated reasoning**.

Keywords

Knowledge Graph, Ontology, Translational Embedding, Link Prediction

ACM Reference Format:

. 2025. A Hybrid Self-Correcting Approach for Embedding Ontology-based Knowledge Graphs. In *Proceedings of KRR@40th ACM/SIGAPP Symposium On Applied Computing (KRR).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The advent of Knowledge Graphs (KG) embeddings, transforming KG entities and relations into vector spaces, enhances KG utility for machine learning tasks [**?** **?** **?** **?** **?**], including link prediction [**?** **?** **?**] and triple classification [**?** **?**]. These embeddings aim to minimize information loss, primarily utilizing positive and negative triples for training. Positive triples, drawn directly from the KG, represent true information, while negative triples, artificially generated, signify false or non-existent KG information [**?** **?** **?** **?**]. During training, embeddings learn to differentiate factual content from falsehoods. It is therefore pivotal to ensure that negative triples are genuinely incorrect and that positive triples are truthful and that they are not the result of a noisy or incomplete KG [**?**]. A foundational translational embedding approach TransE, proposed by Bordes et al. [**?**], pairs each positive triple with a randomly generated negative during training. TransOWL, introduced by D'Amato et al. [**?**], as well as other numerous models in the literature [**?** **?** **?** **?**], leverages the background knowledge contained within ontologies to craft negative samples, aiming for a more informed and beneficial generation process that supports the learning mechanism more effectively. Other approaches harness the intrinsic structure of the KGs. Among these, SANS approach, adopted by Ahrabian et al. [**?**] creates negative triples by substituting entities within a positive triple based on its neighborhood in the KG. A different approach defined in [**?**] applies first a link prediction algorithm with embeddings trained with a known algorithm, and then identifies incorrect predictions against an ontology, and uses these as negative feedback to refine the model through iterative training. Despite community research progress these approaches exhibit notable limitations:

(i) Ontologies may not cover all entities and relationships, leading to unbalanced and unrepresentative training data.

(ii) Structure-based approaches might overlook some semantic nuances that are captured in ontologies but absent in KGs.

(iii) The presence of noise in KGs can compromise the integrity of positive triples, inadvertently training models on inaccuracies. Although the literature includes methods for fact validation and techniques for eliminating inconsistencies [**?** **?** **?**], these are rarely incorporated into the embedding processes.

(iv) The iterative method of [**?**] restricts the number of negative triples available for self-correction, as the link prediction algorithm is not designed to identify incorrect triples, resulting in a potentially low count of such examples.

In this work, we present TransHySeCo, a novel, *hybrid*, and *self-correcting* approach to embed KGs, specifically aimed at enhancing link prediction efficacy. TransHySeCo aims to mitigate all the aforementioned problems related to the state of the art algorithms. Our main contributions are:

(1) An efficient algorithm to remove inconsistencies and reduce noise within the positive triples of a KG (limitation **iii**);

(2) A method to create negative samples by blending semantics with graph topology, overcoming ontology limitations and semantic oversights (limitations **i**, **ii**);

(3) A self-correct approach that, differently from [**?**], exploits a triple classification task, applied to a large number of *(quasi-)true negative* triples, to spot misclassified true negatives, that are used as negative triples for the next self-correct training (limitation **iv**).

*Produces the permission block, and copyright information

KRR, March 31 - April 4, 2025, Sicily, Italy

ACM ISBN 978-1-4503-XXXX-X/18/06

https://doi.org/XXXXXXX.XXXXXXX

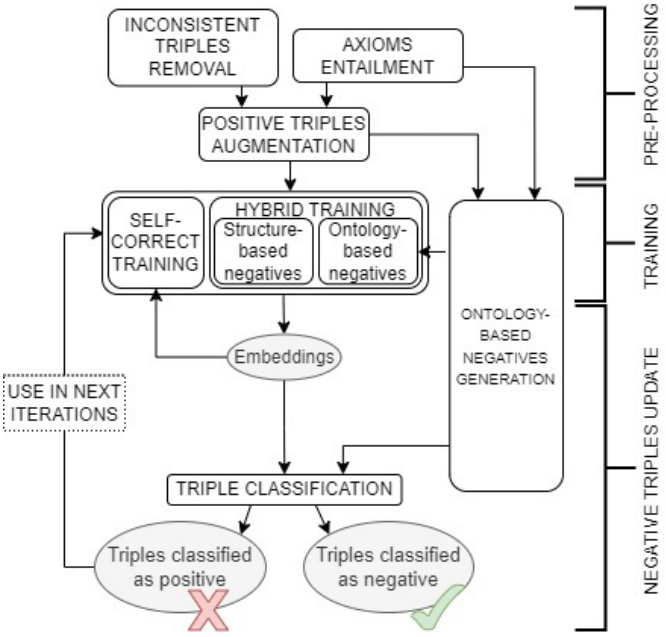

Figure 1: TransHySeCo approach Framework

The paper is organised as follows. Section 2 introduce an overview of TransHySeCo framework. Sections 3, 4, 5 and 6 detail the different phases of TransHySeCo. Section 7 presents our evaluations showing that TransHySeCo outperforms popular models like TransOWL, TransR [**?**] and TransROWL [**?**] on various benchmarks. Section 8 discusses related work on KG embeddings. Section 9 concludes the paper with the perspectives of our approach.

2 TransHySeCo Overview

Let $\mathcal{O}$ be a domain ontology characterized by a set of terminological axioms defined on a set of concepts C and a set of roles $\mathcal{R}$ in a given profile of OWL standard[1]. The set E denotes the set of entities. A knowledge graph $\mathcal{G}$ is a set of triples (h, r, t), where $h \in E$, $r \in \mathcal{R}$, $t \in E$ if $r \neq type$ otherwise $t \in C$, where $type$ denotes the instance relation as *rdf:type* in RDF standard[2].

The objective of TransHySeCo is to define a knowledge graph embedding approach that mitigates the limitations presented above and outperforms in the link prediction task. We achieve this by leveraging the deep semantic information provided by the ontology associated to the KG, which is key to the entire framework of TransHySeCo, depicted in Figure 1, and includes three components:
Pre-processing phase aims to remove inconsistent triples from the KG and to enrich it by inferring new positive triples. This step requires effective inconsistency handling and inference methods.
Training phase includes a hybrid training and a self-correct training. TransHySeCo develops a novel methods for generating negative samples based on ontology and KG. Hybrid training is performed once over both structure-based negatives and ontology-based negatives. For subsequent iterations, self-correct training is performed, where positive samples are paired with the negative samples discovered in the Negative Triples Update phase below.
Negative Triples Update phase prepares new negative triples to be used during the self-correct training by exploiting the embeddings created in the previous training and a triple classification task defined in TransHySeCo. Note that the triple classification is purely based on automated ontology reasoning without requiring additional labelled benchmark.

3 Pre-processing

The pre-processing aims to first remove the triples that make $\mathcal{O}$ and $\mathcal{G}$ inconsistent, resulting in a sub-graph $\mathcal{G}_1 \subseteq \mathcal{G}$ such that $\mathcal{G}_1$ is consistent with $\mathcal{O}$. Then the positive triple augmentation is achieved using $\mathcal{G}_1$ and $\mathcal{O}$ to generate new positive training triples.

3.1 Basic Notations

Following the OWL standard[3], the top and bottom concepts, denoted $\top$ and $\bot$ represent, *resp.*, *owl:Thing* and *owl:Nothing*. An ontology is a set of triples (x, y, z), whose forms are given below. (i) $(x, r, y), r \in \{EquivClass, DisjClass, SubClass, SubProp, DisjProp, DisjClass, Dor$ or (ii) $(x, type, y), y \in \{Symmetric, Funct, Transitive, InvFunct\}$. The semantics of an OWL ontology $\mathcal{O}$ and knowledge graph $\mathcal{G}$ defined by interpretations. An interpretation $I = (\Delta^I, \cdot^I)$, where $\Delta^I \neq \emptyset$ is the domain of the interpretation and $\cdot^I$ is a mapping that maps concepts, roles, and entities is defined as follows: $\top^I = \Delta^I, \bot^I = \emptyset, C^I \subseteq \Delta^I$ for a concept C, $r^I \subseteq \Delta^I \times \Delta^I$ for a role $r \neq type$, and $e^I \in \Delta^I$. I satisfies an ontology $\mathcal{O}$ if it satisfies each axiom $(x, y, z) \in \mathcal{O}$, denoted by $I \models (x, y, z)$. These axioms include equivalence, disjointness, subclass and subproperty relationships, domain and range specifications, inverse properties, and characteristics like symmetry, asymmetry, irreflexivity, functionality, transitivity, and inverse functionality. A knowledge graph $\mathcal{G}$ is consistent *wrt.* $\mathcal{O}$, if it exists I that satisfies both $\mathcal{G}$ and $\mathcal{O}$, denoted by $I \models \mathcal{O} \cup \mathcal{G}$. A triple (x, y, z) is entailed from a knowledge graph $\mathcal{G}$ and an ontology $\mathcal{O}$, written $\mathcal{O} \cup \mathcal{G} \models (x, y, z)$, if for each interpretation I satisfying $\mathcal{O}$ and $\mathcal{G}$, it also satisfies (x, y, z). That is, $I \models (x, y, z)$ if $I \models \mathcal{O} \cup \mathcal{G}$.

3.2 Inconsistent Triples Removal

Given an ontology $\mathcal{O}$, it is known that if $\mathcal{G}$ is inconsistent *wrt.* $\mathcal{O}$, there can be different subgraphs of $\mathcal{G}$ that are consistent with $\mathcal{O}$. It is known that the computation of all the maximal consistent subgraphs is a NP-hard problem even for tractable ontology languages, such as *OWL2EL* [**?**]. In TransHySeCo, we explore a linear algorithm to get one particular consistent subgraph from $\mathcal{G}$ wrt. $\mathcal{O}$. For each entity $e \in E$, we define the set $T_e \subseteq \mathcal{G}$ as the set of triples where e appears. We denote $\mathrm{cnt}(e, c)$ the counter of e *wrt.* a concept $c \in C \setminus \{\top\}$ as the number of triples in T_e that can infer that e is of type c *wrt.* $\mathcal{O}$: $\mathrm{cnt}(e, c) = \#\{t \in T_e \mid \mathcal{O} \cup \{t\} \models (e, type, c)\}$ Intuitively, $\mathrm{cnt}(e, c)$ is a metric to measure the times of an entity e being an instance of the class c under ontology entailment.

[1] https://www.w3.org/OWL/
[2] https://www.w3.org/TR/rdf11-concepts/
[3] https://www.w3.org/TR/owl2-primer/

Example 3.1. Consider $\mathcal{G}$ and $\mathcal{O}$ defined as:
$\mathcal{G}$ ={(*Amsterdam*, *type*, *Agent*), (*Amsterdam*, *timezone*, *EuropeanTime*),
(*BaruchSpinoza*, *birthplace*, *Amsterdam*), (*DutchRepublic*, *capital*, *Amsterdam*)},
$\mathcal{O}$ ={(*capital*, *Range*, *City*), (*City*, *subClass*, *Location*),
(*capital*, *Domain*, *Country*), (*Location*, *EquivClass*, *Place*),
(*timezone*, *Domain*, *Location*), (*birthplace*, *Range*, *Place*)}.

$\mathsf{cnt}(Amsterdam, Place) = 3$ because $(Amsterdam, type, Place)$ can be inferred from $\mathcal{O}$ with triples 2, 3, and 4 in $\mathcal{G}$. However, $\mathsf{cnt}(Amsterdam, Agent) = 1$ as $(Amsterdam, type, Agent)$ is directly from the first triple in $\mathcal{G}$.

Let $mpc(e)$ be one of the most probable concepts, the concepts with the largest counters for e: $mpc(e) \in \mathrm{argmax}_{c \in C,\ \mathsf{cnt}(e,c)>0} \mathsf{cnt}(e, c)$. Let T_{mpc} be the set of triples where each e is associated to its most probable class:
$T_{mpc} = \{(e, type, mpc(e)) \mid \mathsf{cnt}(e, c) \neq 0\}$.

Example 3.2 (Example 3.1 contd.). $mpc(Amsterdam) = Place$, and $T_{mpc} = \{(Amsterdam, type, Place), (DutchRepublic, type, Country)\}$. Note that no type information can be entailed for *BaruchSpinoza* so it is missing in T_{mpc}.

In this example, T_{mpc} is consistent with $\mathcal{O}$. Indeed, this is a general conclusion as shown by the following lemma.

Lemma 3.3. *$\mathcal{O} \cup T_{mpc}$ is consistent.*

We defined a recursive *binarySearch* procedure to remove the inconsistent triples with $\mathcal{O}$ from the original graph $\mathcal{G}$, starting with the consistent triple set $T \leftarrow T_{mpc}$ and $subSetG \leftarrow \mathcal{G}$ as the set of triples to be checked for inconsistency with both $\mathcal{O}$ and T. Each time, *binarySearch* involves checking the consistency of $subSetG$ and $\mathcal{O}$ and T. When consistent, the triples in $subSetG$ are added to T. Otherwise, $subSetG$ is divided into two halves, on each of which a recursive call of the *binarySearch* algorithm is launched. If $|subSetKG|$ reaches one triple and the triple is still inconsistent, it is not added to T. After inconsistent triples removal the output is a new graph.

Proposition 3.4. *Let $\mathcal{G}_1 = binarySearch(\mathcal{G}, \mathcal{O}, T_{mpc})$. Then $\mathcal{G}_1 \cup \mathcal{O}$ is consistent.*

In the subsequent phases of our framework, only the consistent triples $\mathcal{G}_1$ are considered. The algorithmic complexity of this procedure applied to a graph $\mathcal{G}$ with n triples is $O(nf(n))$, where $f(n)$ is the complexity of consistency checking on n triples, which is polynomial for OWL2EL, OWL2QL and OWL2RL profiles of OWL2. This is significantly more efficient than the $O(2^n)$ complexity required to verify the consistency of all possible subsets of n triples.

3.3 Positive Triples Augmentation

Several studies have demonstrated the effectiveness of enriching KGs by applying ontology-based inference [**?**][**?**][**?**]. In our approach, TransHySeCo, we exploit the inference by selecting carefully the set of inference rules to generate new triples that are added to $\mathcal{G}_1$. For instance, $(e, type, c), (c, SubClass, c') \models (e, type, c')$; and $(e_1, a, e_2), (a, EquivProp, b) \models (e_1, b, e_2)$; we thus derive from $\mathcal{G}_1$ a consistent set of positive training triples S which remains unchanged through all training phases of TransHySeCo, including the initial training and subsequent self-correction iterations. We will further discuss the hybrid method for constructing negative triples for the initial training in Section 4 and outline the strategy for updating negative triples in Section 5.

4 Hybrid Training

In TransHySeCo, each positive triple is matched with a negative one generated using a hybrid approach that combines ontology-based and structure-based methods, alongside a portion of randomly generated negatives. This allows a uniform training and improves generalization. The set of negative training triples is defined as follows:

$$S' = S'_{Ont} \cup S'_{Str} \cup S'_{Rnd}.$$

For the training phase, essentially, any embedding model using positive and negative triple set, S and S', is suitable for our needs. Here, for simplicity of presentation, we use the simple but efficient TransE [**?**] model, for which the general loss function to be optimized using stochastic gradient descent (SGD) [**?**], is defined as: $L = \sum_{(h,r,t) \in S} \sum_{(h',r,t') \in S'} \min(0, \gamma + f(e_h, e_r, e_t) - f(e_{h'}, e_r, e_{t'}))$, where e_h, e_r and e_t are the vector embeddings of h, r and t, respectively. We say the negative triple (h', r, t') as "paired" with the positive one (h, r, t). Then the loss function of TransHySeCo can be expressed using the three kind of negative triples as follows:

$$\begin{aligned} L = &\sum_{(h,r,t) \in S_{Rnd}} \sum_{(h',r,t') \in S'_{Rnd}} \max(0, \gamma + f(e_h, e_r, e_t) - f(e_{h'}, e_r, e_{t'})) \\ &+ \sum_{(h,r,t) \in S_{Ont}} \sum_{(h',r,t') \in S'_{\mathrm{Ont}}} \max(0, \gamma + f(e_h, e_r, e_t) - f(e_{h'}, e_r, e_{t'})) \\ &+ \sum_{(h,r,t) \in S_{Str}} \sum_{(h',r,t') \in S'_{Str}} \max(0, \gamma + f(e_h, e_r, e_t) - f(e_{h'}, e_r, e_{t'})) \end{aligned}$$

Additionally, during the training of TransHySeCo, we ensure that $S \cap S' = \emptyset$ and control the random component S'_{Rnd} by a hyperparameter that is optimized empirically.

In the next sections, we define how ontology-based and structure-based triples, S'_{Ont} and S'_{Str}, are generated.

4.1 Ontology-based negative triples generation

Given a positive triple $s \in S$, we create a negative-based triple $s' \in S'_{Ont}$ that is (highly possibly) inconsistent with s leveraging ontological axioms: we call such negatives *(quasi-)true negatives*.

Definition 4.1. A triple s' is called *true negative* wrt. S and $\mathcal{O}$, if there is a triple $s \in S$ such that $\{s\} \cup \mathcal{O} \models \neg s'$.

In contrast to true negatives, quasi-true negatives take the assumption that sibling classes (classes that are at the same hierarchical level but represent different, distinct entities) are mutually disjoint to create inconsistency, which is good practice, for instance for Biological and Biomedical Ontologies community [**?**].

Definition 4.2. A triple s' is called *quasi-true negative* wrt. S and $\mathcal{O}$, if there is a triple $s \in S$ such that $\{s\} \cup \textsc{SiblingDisj}(\mathcal{O}) \models \neg s'$, where $\textsc{SiblingDisj}(\mathcal{O})$ is the ontology $\mathcal{O}$ enriched by adding disjoints among all sibling concepts.

Given a positive triple $(h, r, t) \in S$, the different categories of negative triples in S'_{Ont} are generated from ontological axioms as follows:
O1) (h', r, t) if $r = type$, $(h', type, c) \in S$, $(t, DisjClass, c) \in \mathcal{O}$;

O2) (h, r, d) if $r = type$, $(t, DisjClass, d) \in O$;
O3) (h, p, t) if $r \neq type$, $(r, DisjProp, p) \in O$;
O4) (h, r, h) if $r \neq type$, $(r, type, Irreflexive) \in O$;
O5) (t, r, h) if $r \neq type$, $(r, type, Asymmetric) \in O$;
O6) (h', r, t) if $r \neq type$, $(h', type, c) \in S$ but there is no c such that $(r, Domain, c)$;
O7) (h, r, t') if $r \neq type$, $(t', type, c) \in S$ but there is no c such that $(r, Range, c)$.

Note that O and S are enriched in Section 3.3, then we have that such created negatives are true or quasi-true negatives: negative triples of type O1-O5 are true negatives, and those of type O6-O7 are quasi-true negatives.

Example 4.3. Consider the positive triple $(Germany, capital, Berlin) \in S$ and an axiom $(capital, Domain, country) \in O$.
If $(LesMiserable, type, book) \in S$, we can have an quasi-true negative triple $s' = (LesMiserable, capital, Berlin)$ by O6. s' is highly possible to be a negative triple, though it does not cause inconsistency with O.

S'_{Ont} diverges from the set of negatives generated in TransOWL [**?**] due to the utilization of a broader range of axiom types, in greater quantity, facilitated by "Axioms Entailment", as defined in Section 3.3.

For the negatives related to Domain axiom (type O6), we have an intuitive true negative (h', r, t) if $(r, Domain, c) \in O$, $(c, DisjClass, d) \in O$, and $(h', type, d) \in S$. However, this manner limits the number of generated negatives since it depends on the DisjClass axioms in O. O6 relaxes this requirement and introduces quasi-true negatives. The same reasoning is applied to the negatives of type O7. Such a large number of quasi-true negatives is essential for self-correct training in TransHySeCo. Types O1, O6, and O7 exclude entities not part of any concept, leading to potential training set imbalances. This is addressed by supplementing ontology-based triples with structure-based ones, proportionally to entities without concept instances, mitigating the issues '1' and '2' presented in section **??**: balancing the ontological information but still capturing semantic nuances within it.

4.2 Structure-based negative triples generation: S'_{Str}

In TransHySeCo, the creation of structure-based triples aligns with the principle employed in SANS[**?**]. TransHySeCo enhances SANS by ordering the entities' neighbors: the k-hop neighbors of the entities to corrupt in order to form the negative triples are arranged in ascending order according to their Clustering Coefficient (CC) [**?**]. We define $N_{cc}(x)$ the list of the k-hop neighbors of x in the increasing order based on CC. Given $s \in S$, we create $s' \in S'_{Str}$ by substituting the head h (or the tail t) of s with its k-hop neighbor with the lowest CC. This idea, inspired by Kaoudi et al. [**?**], prioritizes the use of entities in sparse regions of the graph. It's worth noting that, each neighbor related to an entity is used only once in substution of such entity to create a negative triple. Theoretically, for two triples $(h1, r1, t1)$ and $(h2, r2, t1)$, $t1$ could be replaced by the same neighbor e.g., $t*$, resulting in two distinct negative triples $(h1, r1, t*)$ and $(h2, r2, t*)$. However, our empirical observations revealed that reusing neighbors multiple times led to an excessive utilization of certain entities and the neglect of others during the learning process.

5 Negative Triples Update

We define the embedding of an entity x produced during the previous training step as e_x. This phase aims to create a set of new effective negative triples, S'_{Corr}, to be used during the following iteration of training. To create such negative triples, at first a set I of triples inconsistent with $S \cup O$ is created. The triples in I are then classified through a triple classification task and the misclassified triples constitute the set S'_{Corr}. The presence of misclassified negative triples suggests room for improvement in their corresponding embeddings: using S'_{Corr} as negative triples in a new training step we are able to "correct" them.

Ontology-based Negatives (I) Generation. Let I be a set of quasi-true negative triples created following the ontology-based negatives procedure (Section 4.1). These are the candidates for use as negative triples in the subsequent self-correction training. Note that I is different from the one used in the hybrid training phase. And a different negative triple set I should be generated anew in each self-correction training to enhance the embedding. Furthermore, $|I| = Epochs \times |S|$ (where $Epochs$ are the number of training epochs used during training) to ensure that each epoch uses a distinct negative triple for a given positive triple s.

Triple classification. The triples in I are classified as positive or negative using the generated embeddings. For each $(h, r, t) \in I$, $f(e_h, e_r, e_t) = \|e_h + e_r - e_t\|_2^2$. If $f(e_h, e_r, e_t) < \delta(r)$, then (h, r, t) is classified as positive. Otherwise, it is negative. $\delta(r)$ is defined as:$\delta(r) = \max\{f(e_h, e_r, e_t) \,|\, (h, r, t) \in S\}$. We define the set of triples for self-correction as those misclassified according to the current embedding: $S'_{Corr} = \{(h, r, t) \in I \mid f(e_h, e_r, e_t) < \delta(r)\}$ because the triples from I should all be classified as negative.

6 Self-correct Training

The primary objective of the training iterations subsequent to the initial one is self-correction, leveraging the misclassified triples generated in the preceding iteration. The Algorithm 1 outlines the functioning of self-correction in the iterations following the first. The positive triples S, the score function f and the embedding optimization via SGD remain the same with the first iteration. However, the negative triples S', to avoid overfitting, are different: $S' = S'_{Corr} \cup S'_{RndC}$.. S'_{RndC} comprises negatives generated randomly under the following two empirical constraints, whose aim is to reduce the likelihood of generating false negatives and to avoid degrading the quality of the embeddings. For a positive $s = (h, r, t) \in S$ and a randomly generated negative $(h', r', t') \in S'_{RndC}$ if and only if it satisfies:
C1 (Line 10): $f(e_{h'}, e_{r'}, e_{t'}) > f(e_h, e_r, e_t)$
C2 (Line 10,12): $(cos\_sim(e_h, e_{h'}) < 0$ and $t' = t)$ or
$(cos\_sim(e_t, e_{t'}) < 0$ and $h' = h)$
where $cos\_sim(e_x, e_y)$ is the cosine similarity function between the two embedding vectors. The first constraint is based on the definition of f, suggesting that (h', r', t') is less likely to be true than (h, r, t). The second is based on the intuition that, to create a likely negative triple, the entity to be corrupted should be replaced with an entity that is substantially different from it. Substituting it with a similar entity might result in false negatives. Hence, it

Algorithm 1 TransHySeCo self-correct training

TransHySeCo_Self_Corr(prev_embeddings,S'_{Corr},S)

1: Initialize embeddings with *prev_embeddings*
2: **for** $i = 1$ **to** *Epochs*, if $|S'_{Corr}| \neq 0$ **do**
3: **for** $j = 1$ **to** $|S|$ **do**
4: Sample positive triple $(h, r, t) \in$ S
5: **if** $\exists\ m \in S'_{Corr}$ paired to (h, r, t) **and** rand_num(0,100)> $pct\_no\_class\_entities$ **then**
6: Neg. triple $(h', r, t') \leftarrow m$
7: Remove m from S'_{Corr}
8: **else**
9: **if** rand_num(0,100)>$pct\_no\_class\_entities$ **then**
10: Neg. triple $(h', r, t') \leftarrow$ *randomNegative*(h, r, t) satisfying C1 **and** C2
11: **else**
12: Neg. triple $(h', r, t') \leftarrow$ *randomNegative*(h, r, t) satisfying C2
13: **end if**
14: **end if**
15: *updateEmbeddings*(h, r, t, h', r, t')
16: **end for**
17: **end for**

is preferred that the cosine similarity is low. The threshold for accepted similarity was determined empirically and set to 0.

Given the two types of negatives, the loss function becomes:

$$L = \sum_{(h,r,t)\in S_{\text{Corr}}} \sum_{(h',r',t')\in S'_{\text{Corr}}} \max(0, \gamma + f(e_h, e_r, e_t) - f(e_{h'}, e_{r'}, e_{t'}))$$
$$+ \sum_{(h,r,t)\in S_{RndC}} \sum_{(h',r,t')\in S'_{RndC}} \max(0, \gamma + f(e_h, e_r, e_t) - f(e_{h'}, e_r, e_{t'}))$$

It is important to note that the negatives in S'_{Corr} are created as the ontology-based negatives from their corresponding positive triples S_{Corr}. Therefore the same consideration as in the hybrid training on entities without an associated class holds: therefore, negatives in S'_{Corr} are used in a quantity inversely proportional to the percentage of entities that are not instances of any class ($pct\_no\_class\_entities$, Line 5). Excessive use of S'_{RndC} negatives can lead to overfitting, as entities with similarity < 0 to a given one ($C2$) are limited. To address this, the percentage of triples subject to $C2$ is inversely proportional to the percentage of entities that are not instances of any class (Line 12).

Continuation of the pipeline. The pipeline proceeds with an iterative succession of phases “Negative Triples Update" and “Training Self-correction". The two termination conditions for the pipeline are: $I = \emptyset$, the absence of new ontology-based negatives generatable to conduct the classification task; $S'_{Corr} = \emptyset$, the absence of misclassified triples during the triple classification.

After completing theTransHySeCo iterations, one can compare the quality of the embeddings obtained across various iterations and choose to use those of the highest quality for the preferred application context.

7 Experience and Evaluation

In this section, we present the experiments we conducted to evaluate TransHySeCo for link prediction purposes on three distinct real-world KGs: DBPEDIA15k[**?**], YAGO[**?**], and NELL[**?**]. The questions we aim to address are:

(1) To what extent is link prediction influenced by the use of TransHySeCo’s pre-processing method?

(2) How does the hybrid utilization of ontology and structural properties during training impact the performance in comparison to their separate usage?

(3) What is the effectiveness of the iterative self-correcting method?

(4) How much TransHySeCo performs compared to TransE, TransR, TransOWL and TransROWL?

Experimental settings: Experiments were conducted on a a cluster of 16 Dell R610 machines with a dual Intel Xeon processor, 8 cores each and each machine with 24 GB RAM with Ubuntu 18.04. HermiT reasoner version used is HermiT 1.3.8[**?**]. Standard parameters commonly employed in the literature [**?**] were used to enable a fair comparison of the approaches under identical conditions. The chosen parameters include a learning rate of 0.001, 1000 epochs, an embedding dimension of 100, and a margin $\gamma = 1$. Two extra parameters, associated with the training employed by TransHySeCo, pertain to the process of identifying entity neighbors. A choice of $k = 3$ hops was made, and varying numbers of random walks were used for the three KGs: DBPEDIA15k with 10,000 random walks, YAGO with 3,000 random walks, and NELL with 5,000 random walks. These choices were informed by the quantity of neighbors required for training and will be discussed in Section 7.2. The code source and the datasets are available here.

The evaluation of the predicted links relies on two standard metrics, MR and $H@10$ [**?**]. Having two evaluation metrics can complicate the comparison between two sets of embeddings, for example when one set of embeddings has a lower MR while another exhibits a higher $H@10$, because these two metrics assess different aspects of the model’s performance. To address this issue, a single evaluation criterion, the ratio $R_{10} = \frac{H@10_{norm}}{MR_{norm}}$ with range [0,∞], where $H@10_{norm}$ and MR_{norm} are the normalised versions of the two metrics, was adopted: it increases when $H@10$ improves and/or MR decreases, and decreasing when $H@10$ declines and/or MR increases. The normalization of metrics is a necessary step to assign equal weight to the two metrics, despite their differing dimensionalities.

Evaluation datasets include DBPEDIA15k, YAGO, and NELL, ranging from structured Wikipedia data to complex KGs combining sources like DBPedia, Wordnet, and Geonames. Maintaining comparability with TransOWL, we adopted their data split for consistency. Key metrics for these KGs (DBPEDIA15k, YAGO, NELL) are: size (183218; 265285; 148437), entities (12862; 87795; 68620), relations (276; 316; 272), training data (70%, 80%, 80%), testing data (20%, 10%, 10%), validation data (10%, 10%, 10%), and classless entities (29%, 86%, 59%). Additionally, the number of O axioms stands at (26699; 1122106; 296580).

The three ontologies exhibit distinct axiom distributions: DBpedia15k is characterized by a moderate diversity of axioms, including 277 disjoint classes; YAGO instead presents no disjoint classes axioms, but 78720 domain and 33834 range axioms; NELL has a vast number of disjoint classes axioms(18679), alongside specialized axioms such as 239 asymmetric and 268 irreflexive properties, absent in the others. This variance influences their utility in generating new knowledge and the effectiveness of data enrichment and reasoning processes.

7.1 Pre-processing Phase Results

After the “Inconsistent triples Removal" step, the DBPEDIA15K KG has preserved 83% of its triples, the YAGO KG the 100% and for nell NELL only its 63%. In the case of NELL, a larger number of triples

are eliminated compared to DBPEDIA15k, despite the latter having a larger initial KG size. This can be justified by examining the size of their respective ontologies: NELL ontology is considerably richer. In contrast, YAGO, despite having a large ontology, remains consistent. This is explained by the absence of disjoint class axioms in the YAGO ontology, which often contribute to inconsistencies. During the step of "Positive Triples Augmentation", DBPEDIA15K increased its positive triples by 16%, YAGO by 13% and NELL by 106%.

7.2 Training Results

Once the pre-processing phase is completed, training sets are used to produce embeddings and then evaluated on test sets for link prediction purposes. To provide a comprehensive overview of the method's performance, the training results obtained with the link prediction task are presented in table 1. In such table, the best results obtained in a specific iteration of TransHySeCo are compared with: TransE, TransR, TransOWL and TransROWL applied to both the original KGs (referred to as "OG") and the pre-processed KGs (referred to as "PP"); "$Str.only(1it)$" and "$Ont.only(1it)$" the algorithms obtained applying the first iteration of TransHySeCo using respectively only the structure-based and only the ontology-based negative triples; "$Str.w/oCC(1it)$", obtained from applying a version of TransHySeCo where the clustering coefficient (CC) ordering for structure-based negative triples generation is not performed for one iteration (the neighbors are chosen randomly, as in SANS[**?**]);"$TransHySeCo(1it)$", where only the first iteration of TransHySeCo is performed; "$TransRHySeCo$", the algorithm obtained by applying TransR but initializing the embeddings with TransHySeCo, as opposed to the conventional initialization method that uses TransE. The table 1 shows the results obtained with the best percentage of random triples for TransHySeCo, found empirically: 30% for DBPEDIA15k, 60% for YAGO, and 50% for NELL. The results of the trials that led to these choices are detailed in Table 2, where percentages were attempted at intervals of 10 until the optimal option was identified.

The choice of neighbors with a hop number of $k = 3$ for TransHySeCo is motivated by the SANS authors' recommendation, which suggests that the semantic meaningfulness of negatives declines as the neighborhood size increases [**?**]. Empirically, $k = 2$ did not provide enough neighbors for sustained training diversity, as each entity's related neighbors is used only once (as explained in Section 4). It is also counterproductive to TransHySeCo's fundamental aim of compensating for ontology's limitations by generating meaningful negatives using KG structure. So we select $k = 3$ to strike a balance, offering an adequate number of neighbors throughout training without compromising the quality of negative triples. The number of random walks for obtaining the neighbours is influenced by time constraints, as larger KGs require more time to calculate k-hop neighbours, and by the fact that we need have sufficient negative triples for all training epochs.

Discussion - Iteration 1 results: The pre-processing phase demonstrated effectiveness, both in KGs with a multitude of inconsistencies and not: TransE, TransOWL, TransR and TransROWL results improved significantly when applied to the pre-processed KG (PP) with respect to the original one(OG). "$Str.w/oCC(1it)$" results show that the ordering of the neighbors is effective for these dataset: R_{10} is higher with respect to "$TransHySeCo(1it)$" In DBPEDIA15K and YAGO, "$TransHySeCo(1it)$" is better than using only the structural or ontological information. In YAGO, with respect to DBPEDIA, the improvement noticed between using only ontology-based negatives and combining them with structure-based ones is significant. This is likely due to the fact that in YAGO, entities without a class make up 86% of the total entities (see Table **??**). This means that ontology-based triples are created using only 14% of the entities, which does not allow for a representative training set. YAGO highlights the significance of employing an alternative method for generating negative triples alongside ontology-based ones. This additional approach intervenes when ontology-based negatives cannot be created, ensuring the creation of effective and meaningful triples that contribute to learning, as exemplified by the structure-based negatives used in TransHySeCo. In NELL, unlike the other two KGs, the combination of ontology-based triples with structure-based ones in this case is not the best option. The reason lies in the pre-processing phase; in NELL, the training dataset had more than doubled. It is expanded by 106%. All these new triples were generated using ontological knowledge. Consequently, the positive training set for NELL is already saturated with ontological information, which can potentially lead to overfitting when such type of knowledge is used also in the generation of negative triples. The case of NELL therefore highlighted a potential shortcoming of the TransHySeCo approach: when the training set is extensively augmented with ontological knowledge, the utilization of ontology-driven negative triples can inadvertently instigate overfitting. In such scenarios, relying exclusively on structure-based triples proved more effective for its inaugural training iteration.

Discussion - Self-correct training results: In all three KGs, various misclassified triples were found due to the imperfections in embeddings, which fail to encapsulate all semantic nuances of the entities. These misclassified triples were then used as negative examples in training for subsequent iterations. The initial 10 iterations on each KG are summarized in table **??**. It was feasible to conduct more than 10 iterations for each KG: 14 for DBPEDIA15k and over 50 for both YAGO and NELL, though only the first 10 iterations are discussed. These iterations revealed that the highest quality embeddings were achieved within the first 10 iterations for all three KGs: the fourth iteration for DBPEDIA15k, the second for YAGO, and the sixth for NELL. Post these optimal iterations, performance began to decline due to the characteristics of the negative triples used in self-correcting trainings. Specifically, in NELL, while initially preferring structure-based negative triples was advantageous, later iterations using ontological information (which was not redundant with the positive training triples) led to better outcomes. This improvement underscores the utility of using misclassified triples, focusing training on aspects not captured in initial iterations. The TransHySeCo self-correction approach demonstrated effectiveness across all three examined KGs by focusing on the enhancement of inadequately learned embeddings and imposing constraints. These constraints, introduced in the initial iterations, prevent any deterioration of other embeddings. However, the introduction of such constraints eventually results in the recurrent utilization of the training triples (because entities with a cosine similarity < 0 with respect to a

Table 1: TransHySeCo-TransRHySeCo link prediction results.

	DBPEDIA15k			YAGO			NELL		
	MR	**H@10**	R_{10}	**MR**	**H@10**	R_{10}	**MR**	**H@10**	R_{10}
Trans-E (OG)	1099.10	45.53	5.33	8004.35	26.86	2.95	10034.53	25.56	1.75
Trans-OWL (OG)	1104.30	45.55	5.31	8031.60	27.93	3.05	11808.47	23.01	1.34
Trans-R (OG)	1133.47	45.11	5.12	8086.53	26.19	2.84	9220.99	27.07	2.01
Trans-ROWL (OG)	1122.51	45.32	5.19	8057.14	26.95	2.94	9457.35	27.07	1.96
Trans-E (PP)	906.76	46.07	6.53	7903.84	28.81	3.20	9639.58	25.91	1.84
Trans-OWL (PP)	929.65	46.43	6.42	7766.15	28.88	3.26	9779.21	23.78	1.67
Trans-R (PP)	918.10	46.11	6.46	7805.21	29.45	3.31	8964.36	26.87	2.06
Trans-ROWL (PP)	913.12	45.87	6.46	7755.98	29.48	3.34	8793.43	27.08	2.11
Str. only(1 it)	866.20	46.20	6.86	7487.72	29.40	3.45	7993.48	28.29	2.43
Ont. only(1 it)	883.10	46.03	6.70	7813.46	28.87	3.24	9552.01	26.10	1.87
Str. w/o CC(1 it)	877.14	45.96	6.74	7182.62	27.94	3.34	8063.32	28.00	2.11
TransHySeCo (1 it)	861.80	46.04	6.87	7493.36	29.46	3.45	7937.91	27.85	2.41
TransHySeCo	845.91	45.92	**6.98**	7512.84	29.56	3.45	7646.79	28.90	**2.59**
TransRHySeCo	863.30	46.18	6.88	7374.39	29.25	**3.48**	7739.69	27.91	2.47

Table 2: TransHySeCo (1 it) with different percentages of random negatives.

	DBPEDIA15k			YAGO			NELL		
%rd. neg.	**MR**	**H@10**	R_{10}	**MR**	**H@10**	R_{10}	**MR**	**H@10**	R_{10}
10	888.54	46.61	6.75	7809.94	28.99	3.26	8401.65	27.45	2.24
20	875.17	46.33	6.81	7719.40	29.27	3.33	8380.67	27.56	2.25
30	861.80	46.04	**6.87**	7628.87	29.55	3.40	8359.68	27.68	2.27
40	886.03	45.60	6.62	7787.20	28.79	3.25	8313.84	27.51	2.27
50	880.58	45.70	6.68	7657.63	29.39	3.37	7937.91	27.85	**2.41**
60	865.53	45.36	6.74	7493.36	29.46	**3.45**	8342.34	27.38	2.25
70	888.75	45.35	6.56	7847.64	26.57	2.97	8487.74	27.45	2.22
80	888.41	44.87	6.50	7754.23	29.23	3.31	8596.27	26.82	2.14
90	899.32	44.12	6.31	7564.56	29.48	3.42	8714.95	27.17	2.14
100	1042.54	39.56	4.88	7996.59	29.16	3.20	9426.68	26.46	1.93

given entity are limited), leading to a subsequent decline in embedding quality after reaching its best performance. This progressive degradation, attributed to overfitting in iterations subsequent to the optimal one, suggests that the pipeline might justifiably be terminated a few iterations after achieving optimal results. Lastly, as far as TransRHySeCo is concerned, its results are very close to those of TransHySeCo. This demonstrates that the hybrid approach and self-correction mechanism succeed in ensuring a robust modeling of complex relations, capturing their nuances by considering both their semantics and dynamics.

Discussion - Scalability and Practical Limits: TransHySeCo's adaptability to KGs is not hindered by size, provided an associated ontology exists. Training is swift, completing within minutes even for large KGs like YAGO and NELL, indicating size isn't a major barrier. The primary challenge lies in the step "Inconsistent triples removal", because reliant on Hermit reasoner power. While DBPEDIA15K's inconsistency removal took about 4 hours, it was instant for YAGO and extended to 5 days for NELL. Despite this variability, we believe, in the light of the results presented, that this one-time step is crucial for achieving high-quality embeddings.

8 Related Work

Negative samples generation is a challenging issue for KG embedding. Corrupting randomly the head or tail of each triple [**?** **?**] could be counterproductive due to a large amount of missing information in KG. To navigate this pitfall, TransOWL [**?**] harnesses ontological axioms to generate negatives, and SANS [**?**] derives negatives by substituting either head or tail in a positive triple with one of its neighbors. This paper aims to explore a hybrid way for negative sampling. Our experiments demonstrate that different negative sampling methods are indeed complementary. For example, we find that random sampling is helpful for a better generalization, making it indispensable for a knowledge graph embedding model. When combined with the structure-based and the ontology-based negative samplings, we observe a noticeable improvement in performance for our experiments. Ontology-based negative sampling is one way to incorporate ontological knowledge into knowledge graph (KG) embeddings. Compared to classical way of integrating ontological knowledge via the modification of loss functions [**?** **?**], ontology-based negative sampling has the benefit of being implementable on any embedding model, making the two approaches inherently supplementary. The work [**?**] and our approach TransHySeCo follow this direction. TransHySeCo and [**?**] also share the capacity of iterative embeddings training. However, after each iteration of [**?**], inconsistent triples serving as negative samples in the next iteration come from a link prediction dataset. In contrast, TransHySeCo generates a substantial number of negative triples, used in different iterations, purely from the input ontology and KG. Besides discovering inconsistent triples, ontology can also help to enrich positive triples [**?**]. To avoid the high complexity of checking all possible entitlements, often some ad-hoc strategies are applied, such as exploring classical hierarchy for implicit class instances [**?**]. TransHySeCo designs a set of generative rules for generating implicit deducible axioms and triples. Note that such augmentation is only performed in TransHySeCo after removing inconsistent triples from the original knowledge graph, preventing the propagation of noisy data during the training.

9 Conclusion

TransHySeCo is a novel, hybrid and self-correcting approach to embedding KGs, specifically designed to enhance link prediction accuracy. It shows that embeddings training benefits from a combination of ontology-based and structure-based negatives. The self-correction mechanism leverages misclassified triples as negative examples in subsequent iterations, enhancing embedding quality without adversely affecting unaffected embeddings. TransHySeCo's is designed to be orthogonal to the embedding model, providing a cost-effective way to improve the model without altering the trainable parameters. Future work involves developing methods to automatically balance randomly created negative triples and

Table 3: TransHySeCo 10 iterations results.

	DBPEDIA15k			YAGO			NELL		
Iteration number	**MR**	**H@10**	R_{10}	**MR**	**H@10**	R_{10}	**MR**	**H@10**	R_{10}
1	861,8	46,04	6,871	7493,36	29,46	3,451	7937,91	27,85	2,407
2	844,4	45,75	6,968	7512,84	29,56	**3,454**	7793,45	27,28	2,401
3	847,2	45,84	6,959	7558,1	29,62	3,440	7720,2	28,58	2,540
4	845,91	45,92	**6,982**	7570,55	29,55	3,426	7633,41	28,36	2,549
5	846,71	45,83	6,961	7649,01	29,3	3,363	7590,27	28,47	2,573
6	845,87	45,54	6,924	7646,772	29,73	3,413	7646,79	28,9	**2,593**
7	847,41	45,56	6,915	7679,93	29,34	3,354	7761,07	28,02	2,477
8	845,72	45,55	6,927	7703,84	29,3	3,339	7680,71	28,38	2,535
9	846,95	45,64	6,931	7690,78	29,35	3,350	7657,58	28,85	2,585
10	846,88	45,61	6,927	7664,24	29,54	3,383	7728,08	29,02	2,576

studying criteria to assess ontology-based data augmentation, thus preventing overfitting and making TransHySeCo more optimal.
Supplemental Material Statement: The source code of the whole TransHySeCo framework, the benchmark datasets used for the experiments are available at [4].

References

[4] TransHySeCo repository: https://anonymous.4open.science/r/TransHySeCo-8CE0/