Improving Transitive Embeddings in Neural Reasoning Tasks via Knowledge-Based Policy Networks

Shervin Mehryar[1] and Remzi Celebi[2]

[1] Department of Electrical & Computer Engineering, University of Toronto, Canada
shervin.mehryar@utoronto.ca
[2] Institute of Data Science, Maastricht University, Maastricht, The Netherlands
remzi.celebi@maastrichtuniversity.nl

Abstract. This paper proposes an approach to embed ontologies in order to deal with reasoning based on transitive relations, using the datasets provided for the SemRec Challenge at ISWC 2022. Knowledge Graph Embedding (KGE) methods provide a low-dimensional representation of the entities and relationships extracted from the knowledge graph and have been successfully used for a variety of applications such as question answering, reasoning, inference, and link prediction. However, most KGE methods cannot handle the underlying constraints and characteristics of ontologies, preventing them from performing important reasoning tasks such as subsumption and instance checking. We propose to extend translation-based embedding methods to include subsumption and instance checking reasoning by leveraging transitive relations. Experimental results show that our approach can achieve Hits@10 as high as %73 using samples generated by a policy network.

Keywords: ontology embedding · knowledge graph embedding · reasoning · policy network · neural reasoning

1 Introduction

Neural-symbolic computing, an approach to combine symbolic approaches with neural models, can provide reasoning and explainability capabilities while being highly scalable in analyzing and inferring new data from a large volume of data. Deep learning has achieved significant success in many areas in recent years, but the success of deep learning models is limited by the availability of sufficient quality labeled data for the task at hand. Deep Learning models input-output relationships by generating neural representation of data, but these neural representations often fail to consider common sense or acquired domain knowledge. A promising recent approach to neural-symbolic integration is Knowledge Graph Embeddings (KGE), which map the symbolic representation of a domain data into vector spaces. KGE can perform approximate reasoning that takes place in vector space to reduce the complexity of reasoning time.

There are many existing KGE methods to try to preserve certain properties of KG [1,13,11,2]. TransE, being one of the simple and successful knowledge graph embedding methods, proposes a scoring function to measure plausibility of a fact [1]. The TransE scoring function tries to minimize the distance between the translated head vector embedding $(h + r)$ and the tail vector embedding (t) where the facts are represented as a triplet $< h, r, t >$ (head, relation, tail). The authors later proposed rTransE, an extension of TransE, which learns rules that compose relationships as a sequence of translations in the embedding space. Particularly, rTransE embeds the composition of two relationships (l_1, l_2) as the sum of their embeddings and calculates the distance between the translated head embedding $(h + l_1 + l_2)$ and the tail embedding (t) as a scoring function.

KGs can take the form of Ontologies, which are in turn meant to represent more complex relationships (i.e. negation, conjunction, disjunction and quantifiers). However, most KGE methods cannot handle the underlying constraints and characteristics of the ontologies. As a result, researchers have recently tried to address this issue by incorporating the geometric structure of the EL description logic into the embedding space (EmEl) [6] or modifying translation-based embeddings that can handle complex many-many roles in the EL embeddings (EmElvar)[9].

In this work, we extend TransE and rTransE to make subsumption and instance checking reasoning possible whereby reasoning can take place in the vector space by leveraging transitive relations. We further improve the quality of embeddings using multi-hop samples generated by an agent's policy network. The agent is a neural network that takes as input an entity vector embedding (i.e. state) and outputs a relationship (i.e. action). Through careful design of a reward function, the agent learns a good policy to choose actions that lead to more meaningful and longer sequence of translations. We investigate qualitatively and quantitatively the behaviour and performance of the proposed policy-based embedding method, which we call aTransE.

2 Related Work

In [7], Liu et al. proposed a neural-symbolic reasoning approach PoLo that is based on reinforcement learning and logical rules. Their method extends MINERVA framework [3], which trains a reinforcement learning agent to perform a random walk with the goal of reaching a target entity. They addressed the issue of noisy reward signals by incorporating logical rules (used as meta-paths) in their reward function for agent to perform policy-guided random walk on a background knowledge graph. They evaluated their method on a link prediction task where hold-out drug-disease links were predicted in the drug repurposing setting. They reported that their approach outperformed the the state-of-the-art methods on the Hetionet biomedical dataset.

In [8], the authors combined embedding and rule-based approaches using ensemble learning for knowledge graph completion tasks. They claim that these two methods complement each other to improve the performance of knowledge

graph completion methods. These methods are combined with linear blending function at the relation level. Also the paper introduced RuleN, a rule learning method, which supports Type P_n (n-length path rules) and Type C rules (rules with constants in their heads). To show advantages of the combined approaches, a sort of experiment is conducted. Their ensemble method outperformed embedding methods, including HolE, RESCAL, and TransE, and rule-based learning methods, AMIE and RuleN, in the evaluation. Their methodology differs from ours in that it combines two classes of methods with linear blending function and a classifier is trained with an equal number of positive triplets and generated negative triplets. Whereas we combine rules and embeddings into labeling functions and ensemble of output labels from each labeling function. While we used labeling functions to generate labels for a triplet from each rule-guided walk and integrated labels with a probabilistic model, in [8] logistic regression was used to train on the input data of as normalized scores from each individual model (either rule and embedding based model).

Zang et al. [14] proposed a novel framework IterE which aims to predict new triplets by iteratively combining embedding and rule learning. To address the quality issue in the embeddings trained on sparse entities, IterE uses ontological axioms to enrich the knowledge graph. The framework takes a knowledge graph and relation embedding R as input and produces a set of axioms and confidence scores. After new axioms (triplets) with high scores are added to the knowledge graph, an embedding learning step is initiated with the updated KG. This process continues with k iterations to improve the embeddings.

3 Method

We propose an algorithm that given a set of triplets describing facts from a (partial) KG, learns embeddings for the entities and relations. The embeddings are subsequently used to perform reasoning tasks, such as link prediction. In order to account for the case where multiple relation types are present in the data (see Figure 1 left), we train a policy agent that generates heterogenous paths across the KGE. The one-to-one relations are embedded into a vector space directly using the subject, relation, and objects relations in the training data. In order to further capture and embed properties that increase the expressivity of the model, we incorporate two additional sampling techniques. We generate quadruples using the transitive property in which triplets (1 hop) in the training data are extended to quadruples (2 hops), by following a second predicate sharing a common subject (i.e. 2-hop reasoning). We believe that this is a critical consideration to allow the model to infer subsumption and instance assertions. The details are given in the following.

More formally, we formulate the problem as an optimization problem for which optimal vector embeddings are learned for entities and relations so that given a knowledge base graph $\mathcal{G}$ represented with a relation r between entities e_1 and e_2, the following relation also holds in the corresponding vector space: $\vec{e_2} = \vec{e_1} + \vec{r}$. This is a representation of single triplets in the ontology in the

form $(e_1, r, e_2) \in \mathcal{G}$. We further extend this notation to include K-hop reasoning, i.e. $(e_1, \{r_k\}_{k=1}^K, e_K) \in \mathcal{G}$. Adopting the notation in [4], the energy for this path is given by $d(e_1, \{r_k\}_{k=1}^K, e_K) = ||\vec{\mathbf{s}}_K(e_1, \{r_k\}_{k=1}^K, e_K) - \vec{e}_K||_2$. Here, $\vec{\mathbf{s}}_k$ is the vector entity starting from entity e_0 and following relations $r_1, \cdots, r_k$, i.e. $\vec{\mathbf{s}}_K(e_1, \{r_k\}_{k=1}^K, e_K) = \vec{e}_1 + \vec{\mathbf{r}}_1 \cdots + \vec{\mathbf{r}}_K$. In this view, we proceed to minimize the following loss functions in order to learn good entity and relation embeddings:

Triple Loss: given a set $\mathcal{S}$ of triplets representing facts in the KG, the Triple loss refers to the following criterion:

$$L_{tri} = \sum_{(s,r,o)\in\mathcal{S}} ||\vec{s} + \vec{r} - \vec{o}||_2, \tag{1}$$

where as before, $\vec{s}$, $\vec{r}$, and $\vec{o}$ are vector representations in $\mathbb{R}^d$ corresponding to source entity, relation, and object entity in the ontology. These representations are learned using the triplets in the data and embeded as a d-dimensional vector similar to the process in TransE.

Quadruple Loss: to improve the quality of representations, we consider a secondary loss corresponding to paths with length $K = 2$. Given triplets (s, r_1, o) and (o, r_2, t), the 2-hop path loss referes to the following crietrion:

$$L_{quad} = \sum_{(s,r_1,r_2,t)\in\mathcal{S}} ||\vec{s} + \vec{r_1} + \vec{r_2} - \vec{t}||_2, \tag{2}$$

where as before, $\vec{s}$, $\vec{r_1}$, $\vec{r_2}$, $\vec{t}$ are vector representations in $\mathbb{R}^d$ corresponding to first source entity, first relation, second relation, and second tail entity. This structure ensure that learned vector representations are near each other for entities at two links away following the composition rule [4]. Note that with this setting, diversity in type of relations to the second order degree is achievable. However, this introduces a squared polynomial level of complexity in the number of relations. In other words, given the edge diversity size (i.e. number of edge types) R and maximum path length support K, the time and space requirements grow in the order of $\mathcal{O}(R^K)$ and $\mathcal{O}(dR^K)$, respectively. We next propose a policy agent in the following in order to circumvent this issue and provide expressivity in large KG graphs. For KG facts generation in order to compute this loss, we form quadruples that share a common middle node, i.e. (s, r_1, o) and (o, r_2, t) from the the triplet (s, l, t), which we generate from the all the present triplets and store as quintuple samples in the form (s, r_1, o, r_2, t).

Agent Loss: to accommodate path lengths $K > 2$, we propose a Policy Agent similar to [12] that is parameterized by a neural network to provide action a_k at time step $k > 2$ corresponding to one of the relations. The agent learns a distribution over all possible actions based on the current entity relation and the path taken up to then. First, we train the agent on the K=1 and K=2 paths in a supervised fashion using the representations learned above. We then retrain

the agent using memory replay and the following reward function:

$$R(k) = \begin{cases} 0.0 & s = t \ for \ (s, a_k, t) \\ +0.1 & (s, a_k, t) \notin \mathcal{S}, \\ +1.0 & k > 2 \ and \ (s, \{a_1 \cdots a_k\}, t) \in \mathcal{S} \\ -0.1 & otherwise \end{cases} \tag{3}$$

in which the first condition allows self-loop (but doesn't encourage or prohibit it), the second condition encourages unseen but potential new paths, and the third condition rewards paths of length $k > 2$ for which a corresponding triplet exists in the KG (i.e. new paths). In section 5 we provide quantitative examples of these cases. In general, the agent is a multi-layer neural network with ReLu activity functions, and a soft-max output, parameterized by θ. It outputs probabilities according to distribution $\pi(a|s)$ for each action given the current state. The state corresponds to the current entity node along the path, which in our case is represented by an embedding vector of dimension d from TransE. The action corresponds to the relation type conditioned on the state at path length k. The given reward function aims to maximize the following expected objective:

$$L_{agent}(\theta) = \sum_k \sum_a \pi(a_k|s_k;\theta) R(k). \tag{4}$$

In order to learn the parameters of the agent's policy consistent with the underlying structure of the KGE, at first the agent is trained in a supervised manner given available triplets. In this fashion, the agent learns to make myopic decisions or 1-hop reasoning by design (i.e. $k < 2$). For the case of $k > 2$, at each level the agent makes a choice among R actions according to a soft-maxed distribution conditioned on the current input state. In contrast to the worst case scenario where the agent would have been making a choice among R^K branches, the agent instead makes local decisions here and receives a reward at the end of each episode. We define episodes by the agent taking actions up to a pre-specified maximum length K or when another node is reached as a valid target following actions a_1 to a_k. At that point, the agent receives a maximum reward of 1 in our framework if successful. Since in fact at step k the choice will depend on the total number of sub-branches at step $k + 1$ and this is computationally intractable, we train in parallel a value function that estimates the value of next state.

4 Results

We use three datasets, namely OWL2Bench, ORE, and CaLiGraph [5], in which training, validation, and test data are given in triplet format as part of the SemREC Challenge at ISWC 2022. The OWL2Bench is further provided in two separate sub-datasets, namely OWL2Bench1 containing 7989 class assertions in training data and 157 in test data as well as 105 subclass relations in training

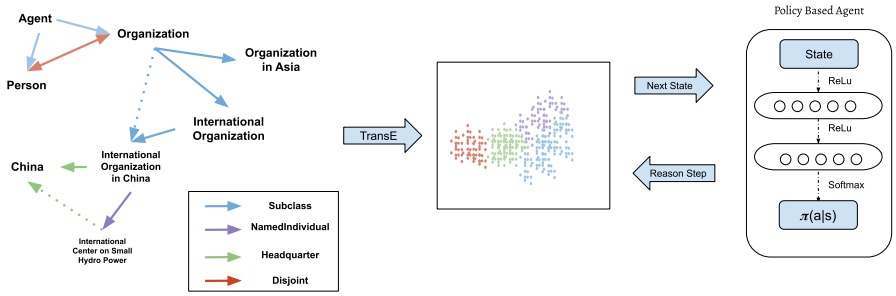


Fig. 1: Embedding generation for entities and relations in a given Knowledge Graph using Policy Network that generates paths (hops) of length upto K.

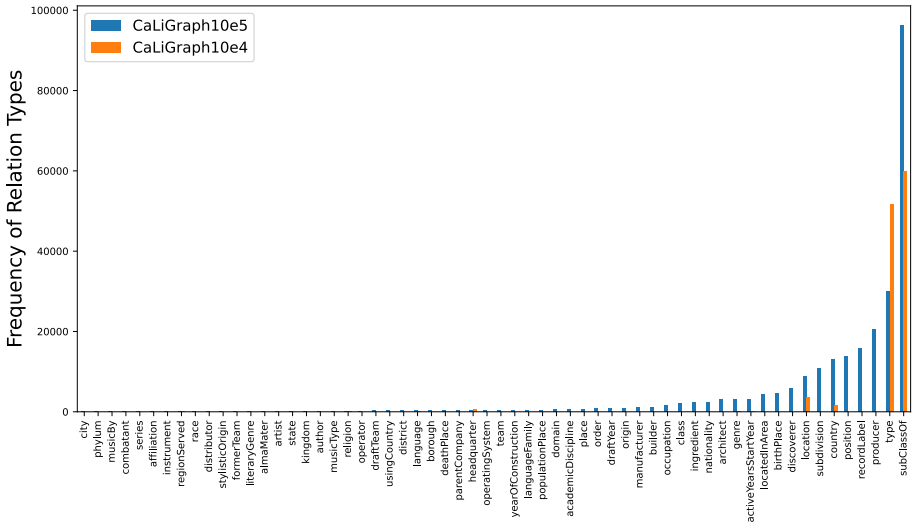


Fig. 2: Distribution of relation types in CaLiGraph10e4 and CaLiGraph10e5.

data and 64 in test data; and OwL2Bench2 containing 15526 assertions in training data and 146 in test data as well as 105 class relations in training data and 53 in test data. ORE is further provided in three subsets, namely ORE1 with 8194 training and 9073 test, ORE2 with 8204 training and 9369 test, and ORE3 with 8187 training and 9122 test data points of subclass relations, as well as, 53048 training and 42851 test assertions for ORE1, 53081 training and 42432 test assertions for ORE2, and 53014 training and 43181 test assertions for ORE3. The last dataset is CaLiGraph which contains more ontology types, again with subclass and type forming the majority of data, provided in different subsets of size 10^4 and 10^5 (as well as the full ontology which we do not consider due to

Table 1: Evaluation results on the final test data sets using TransE, rTransE (recurrent TransE), and aTransE (our method). Reported Hits@N separately for SubClass relation, Assertation type, and all relations combined.

		SubClass Relation		Assertion Relation		Combined Relations	
		Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10
OWL2Bench1	TransE	0.033	0.233	0.004	0.115	0.006	0.088
	rTransE	0.03	0.3	0.363	0.466	0.033	0.348
	aTransE	-	-	-	-	0.053	0.583
OWL2Bench2	TransE	0	0.3	0.001	0.025	0.001	0.014
	rTransE	0.033	0.2	0.225	0.510	0.007	0.115
	aTransE	-	-	-	-	0.027	0.397
ORE1	TransE	0.019	0.130	0.004	0.090	0.031	0.273
	rTransE	0.012	0.111	0.094	0.253	0.030	0.171
	aTransE	-	-	-	-	0.007	0.115
ORE2	TransE	0.004	0.107	0.012	0.113	0.009	0.162
	rTransE	0.107	0.115	0.055	0.233	0.020	0.145
	aTransE	-	-	-	-	0.012	0.124
ORE3	TransE	0.010	0.087	0.011	0.090	0.009	0.079
	rTransE	0.011	0.119	0.094	0.281	0.046	0.288
	aTransE	-	-	-	-	0.011	0.138
CaLiGraphi10e4	TransE	0.021	0.121	0.008	0.091	0.007	0.099
	rTransE	0.2901	0.408	0.424	0.604	0.460	0.679
	aTransE	-	-	-	-	0.556	**0.730**
CaLiGraphi10e5	TransE	0.002	0.036	0.008	0.091	0.005	0.043
	rTransE	**0.678**	**0.713**	0.017	0.095	0.671	0.687
	aTransE	-	-	-	-	**0.688**	0.703

memory requirements). The distribution of all predicates for both is shown in Figure 2.

During the training phase, each unique entity is represented by a vector embedding of dimension size $D = 30$, which we found to be the most effective across all datasets from the validation data. Initially, vector representations are generated randomly. During training we further normalize the magnitude of each vector representation to one [4]. Similarly, each relation type is represented with a vector of the same size, i.e. $D = 30$, in order to simplify vector operations as described in Section 3. Overall, for ORE there are about 6650-6673 entity vectors and 2 relation vectors. For OwL2Bench1 there are about 113-115 unique entity vectors and again 2 relation vectors that we consider. For CaLiGraph10e4 and CaLiGraph10e5 there are 10311 and 75195 with multiple relation types (see Figure 2). Batch sizes of 25 from triplets and quadruple are selected in order to minimize the the loss functions in relations 1 and 2, respectively. An stochastic gradient optimizer such as ADAM with a learning rate of 10^{-5} is used to update the vector representations as model parameters.

One of the important contributions from this work is additional sample generation for the training process in order to improve the quality of transitive

Table 2: Unique Samples Generated for Embedding Learning. There are three types of samples used during training: 1) triplets, extracted directly from the training data; 2) quadruples, generated by combining triplets sharing a common node; 3) multi-hop samples of length greater than 3 generated using the agent's policy network.

Relation Generation	Samples
Triplet (1-hop)	FineArts → NonScience, Elective Course → Thing, Civil Engineering → Engineering
Quadruple (2-hop)	ModernArts → FineArts → NonScience, Phylosophy → Hummanities → NonScience, CivilEngineering → Engineering → NonScience
Agent (multi-hop)	Environmental Organization → Energy Organization → Non Renewable Resource Company → Coal Company → Non-renewable Resource Company Disestablished in 1911, Organization → Educational Organization → Philosophy Organization → Human Rights Organization based in China

embeddings. There are three types of samples generated in order to learn good representations for the structure of the underlying knowledge graph data. The first set of samples is taken by converting the training data directly into a subject, predicate, and object triplet in a vectorized form. These samples intuitively provide local information as pertaining to a single hop reasoner. In order to boost the reasoning power of the model, a second set of samples is generated by considering extended link prediction and extracting subject-object relations from the training data at a 2 hops distance. This second type of sample generation is utilized to minimize the loss in relation 2. In order to still improve the predictive power of the model by taking into account longer paths in a third set, we train a policy network agent with two hidden layers of size 60 and 64, followed by a ReLU activation function following each layer. The final layer is a fully connected one with output size corresponding to the number of predicate/relations for each task. We first train the agent in a supervised manner on the triple and quadruple samples. Afterward, the agent is trained according to the reward function described in Section 3 and is capable of generating relation and entity candidates at a length greater than 2 hops away. Specifically, we limit this length to a maximum of the path length 5 in each episode.

During the test phase, the learned embeddings are used to perform the task of link prediction on the test triplets provided in each dataset as follows. For each test triplet, the tail embedding is obtained by adding the corresponding embedding vectors of given head and relation which can be retrieved directly by

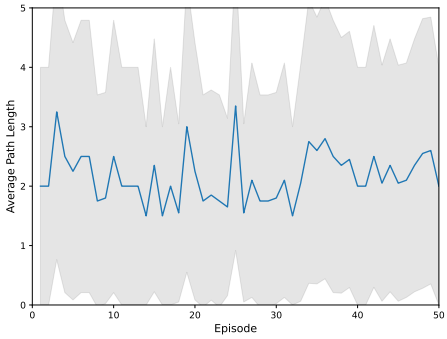
the model. The resulting embedding is then compared and ranked using cosine similarly against all entities present in the dataset. We report success per case if the true tail entity is in the top K predictions, i.e. Hits@K. In particular, we report Hits@1 and Hits@10. For each dataset, we further consider subclass, class assertion, and all relationships together for better analysis of the learned models. The training and validation code[3] are made available publicly.

Table 1 shows the results reported for each dataset for different embedding learning methods and relationship type. In particular, we report the TransE performance metrics using triplets as learned from the loss optimization in relation 1. The embeddings learned through optimization of the loss relation in equation 2 using quadruple sampling in addition, corresponding to the recurrent TransE, or rTransE, are also evaluated and reported. We consider embeddings learned for class and assertion types separately first (middle columns) as well as the combination of all relation types (last two columns). In this latter case, the three loss functions are minimized in combination with each other. The performance of model in terms of the learned embeddings using agent's generated samples in addition to triplets and quadruples, named aTransE, are included for this last case since improvements otherwise we found to be negligible. It can be observed that in each case, by including quadruple and agent's samples, the overall embedding quality is improved with highest performance achieved at %68.8 Hits@1 and %73 Hits@10 over all. It is concluded then that by including more relation combinations and simultaneously minimizing the combined losses including the agent's generated samples the performance is consistently improved.

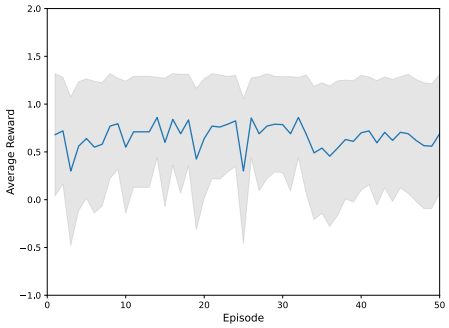
5 Discussion

One of the key contributions of this work in order to improve the quality of entity and relation embeddings learned by the model is attributed to the quality of sample generation, which is deemed important for many down-stream applications such as question answering [10]. While in the previous section we discussed quantitative results obtained across different tasks, in Table 2 we also provide qualitative examples of such samples from the CaLiGraph dataset. The triplets are unique samples directly taken from the training data. The quadruple relations are generated by joining different triplets sharing a common node. Samples of length 2 or more generated through the agent's policy network are shown in the last row. We observe quantitatively that these samples are meaningful which justifies the improvements. In one case, the agent follows subclass relationships from an entity 'Environmental Organization' down to an instance 'Non-renewable Resource Company Disetablished in 1911' at length 5 away, attesting to the quality of policy learned based the subsumption rule. In another case, the agent's policy is able to realize heterogeneity in relationships for example by following subclass relations from 'organization' down to 'philosophy organization', and location/country to instance 'Human Rights Organization based in China'.

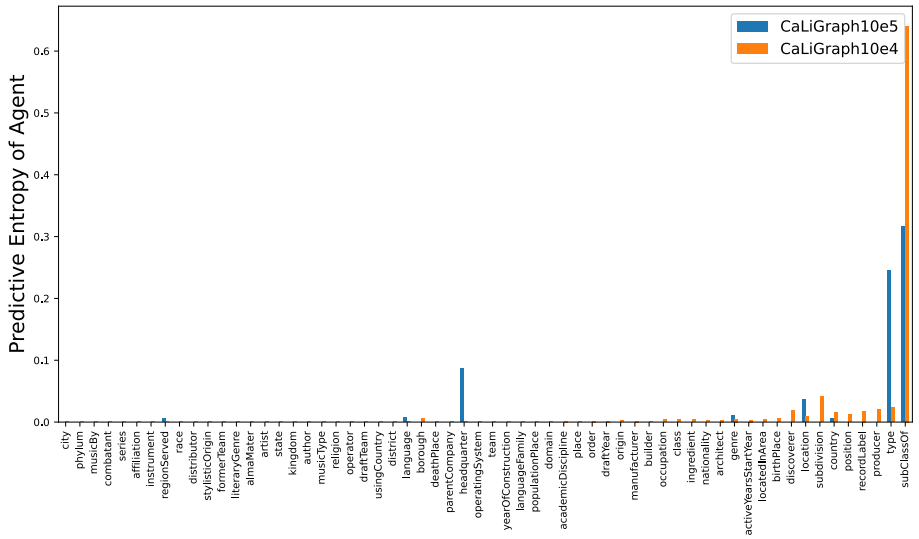
[3] `https://github.com/MaastrichtU-IDS/SemREC`



(a) Average, maximum, and minimum path lengths



(b) Average, maximum, and minimum accumulated rewards



(c) Agent's predictive entropy conditioned on the agent's state, according to which the agent is more likely to choose a relation with a higher value

Fig. 3: Path length, reward, and predictive behaviour of the the trained agent on CaLiGraph datasets. The agent is trained for 50 episodes and results are averaged over 100 runs, including maximum and minimum values.

Moreover, the quality of the agent's generated samples can be assessed based on the length of paths generated corresponding to the number of reasoning hops as well as the reward achieved during each run. In Figure 3a, the minimum, average, and maximum path lengths are shown for the agent during training up to 50 episodes, over 100 different runs. It can be seen while on average the agent tends to produce paths of length two (corresponding to quadruple samples), often it learns to explore new paths of higher lengths. In Figure 3b, the average reward

over one hundred runs of the training algorithm upto 50 episodes is depicted. Consistent with the previous figure and the reward function defined in Section 3, it can be observed that the agent is able to reach average reward of 1 when exploring path lengths greater than 2.

Lastly, in order to better understand the quality of the learned policy by the agent, we evaluate the predictive entropy over the domain of relations across the CaLiGraph datasets which contain many more possibilities with respect to the choice of the relationship types. This quantity can be interpreted as the likelihood that the agent will generate a sample by choosing any relation type given a specific state embedding as input. In other words, the agent essentially learns a distribution over the action space conditioned on the state its in, which in our development thus far corresponds to entity and relation pairs. As shown in Figure 3c, given any state then the agent has learned a meaningful distribution over all relations which consequently allows for diverse sample generation. Interestingly as compared to Figure 2, the agent's behaviour is reasonably well matched to the distribution seen in the original datasets. It's believed that with a better policy network (e.g. more layers and reward function) as well as more episode training this performance can significantly improve.

6 Conclusion

Learning good embeddings in ontological data is challenging due to underlying structure, incompleteness, and heterogeneity. While many attempts have been made to address these issues, data specific development of good models using neural reasoners remains an open problem. In this work, we propose a learning algorithm for three datasets, namely ORE, OWL2Bench, and CaLiGraph, which focuses on the transitive properties in the underlying knowledge base and improves the quality of learned embeddings through multi-hop sampling and combined loss minimization. Through experiments, we show that our approach can achieve Hits@10 metric as high as %73 using samples generated by a policy network.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 26. Curran Associates, Inc. (2013), https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf
2. Celebi, R., Yasar, E., Uyar, H., Gumus, O., Dikenelli, O., Dumontier, M.: Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction using linked open data (2018)
3. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., McCallum, A.: Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. arXiv preprint arXiv:1711.05851 (2017)

4. García-Durán, A., Bordes, A., Usunier, N.: Composing relationships with translations. Ph.D. thesis, CNRS, Heudiasyc (2015)
5. Heist, N., Paulheim, H.: The caligraph ontology as a challenge for owl reasoners (2021). https://doi.org/10.48550/ARXIV.2110.05028, https://arxiv.org/abs/2110.05028
6. Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R.: El embeddings: Geometric construction of models for the description logic el++. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 6103–6109. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/845, https://doi.org/10.24963/ijcai.2019/845
7. Liu, Y., Hildebrandt, M., Joblin, M., Ringsquandl, M., Raissouni, R., Tresp, V.: Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In: Verborgh, R., Hose, K., Paulheim, H., Champin, P.A., Maleshkova, M., Corcho, O., Ristoski, P., Alam, M. (eds.) The Semantic Web. pp. 375–391. Springer International Publishing, Cham (2021)
8. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In: International semantic web conference. pp. 3–20. Springer (2018)
9. Mohapatra, B., Bhatia, S., Mutharaju, R., Srinivasaraghavan, G.: Emelvar: A neurosymbolic reasoner for the el++ description logic. In: SemREC@ ISWC. pp. 44–51 (2021)
10. Ning, X., Ammar, A., Yilmaz, A., Mehryar, S., Celebi, R.: Semantic answer type prediction by using bert classifier and rule-based ranking strategies. In: Proceedings of the SeMantic Answer Type and Relation Prediction Task at ISWC 2021 Semantic Web Challenge (SMART2021): co-located with the 20th International Semantic Web Conference (ISWC 2021). p. 66. CEUR Workshop Proceedings (2022)
11. Sun, Z., Deng, Z., Nie, J., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=HkgEQnRqYQ
12. Wang, S., Mao, W.: Modeling inter-claim interactions for verifying multiple claims. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 3503–3507 (2021)
13. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI conference on artificial intelligence. vol. 28 (2014)
14. Zhang, W., Paudel, B., Wang, L., Chen, J., Zhu, H., Zhang, W., Bernstein, A., Chen, H.: Iteratively learning embeddings and rules for knowledge graph reasoning. In: The World Wide Web Conference. pp. 2366–2377 (2019)