ReInform: Selecting paths with reinforcement learning for contextualized link prediction

Marina Speranskaya
Center for Information and
Language Processing, LMU
Munich, Germany
speranskaya@cis.lmu.de

Sameh Methias
Technical University of Munich
Munich, Germany
samehmetias@gmail.com

Benjamin Roth
Research Group Data Mining and Machine Learning
University of Vienna
Vienna, Austria
beroth@univie.ac.at

Abstract

We propose to use reinforcement learning to *inform* transformer-based contextualized link prediction models by providing paths that are most useful for predicting the correct answer. This is in contrast to previous approaches, that either used reinforcement learning (RL) to directly search for the answer, or based their prediction on limited or randomly selected context. Our experiments on WN18RR and FB15k-237 show that contextualized link prediction models consistently outperform RL-based answer search, and that additional improvements (of up to 13.5% MRR) can be gained by combining RL with a link prediction model. The PyTorch implementation of the RL agent is available at https://github.com/marina-sp/reinform.

1 Introduction

In link prediction, also known as knowledge graph (KG) completion, the task is to find missing entries in a KG, based on other information already contained in the graph. A common formulation of this problem is to present incomplete tuples of the form $(e_x, r_q, ?)$ for which a model is expected to find the missing entity e_y that stands in relation r_q to the entity e_x. In this paper we tackle the task of *contextualized* link prediction for KGs, where additional information about e_x from its graph neighborhood, later referred to as *context*, is engaged in the prediction process.

In contextualized link prediction, two strategies have been proposed: (1) *Search-based*, were the answer entity is expected to be contained in an existing path, and relevant paths are searched for, based on the query tuple (Lao et al., 2011; Das et al., 2018); (2) *Prediction-based*, where the missing entity is predicted (out of all known entities) from a (contextualized) representation of the query tuple (Bordes et al., 2013; Wang et al., 2019).

Minerva (Das et al., 2018) is a prominent neural *search-based* approach for KG completion. In Minerva, a path search is performed using neural reinforcement learning (RL), and the entity at the endpoint of the returned path is taken as the answer to the query tuple. CoKE (Wang et al., 2019) is an highly effective neural *prediction* architecture for contextualized link prediction in knowledge graphs. CoKE takes chains of knowledge graph tuples and predicts missing entities using the Transformer architecture (Vaswani et al., 2017).

In this work, we explore how to combine the advantages of both worlds, leveraging search to provide the most useful information to a prediction model, which then has the freedom to predict any missing entity (even if not on a path returned by the search). This is achieved through the interplay of two neural networks: a prediction network, which bases its prediction on a path returned by the path search, and a path search that provides paths to the prediction model. We compare two transformer architectures for the prediction model: *Transform-CoKE*, that uses chains of relations only, as in CoKE, and *Transform-InterEnt*, an extension of CoKE that includes intermediate entities in the paths. In addition, we integrate a RL architecture based on Minerva as a path search model. However, in order to overcome the answer accessibility limitation of Minerva and tailor the search to best benefit the prediction model, the search model is trained with a modified loss function that takes the prediction model into account.

Experiments on FB15k237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2017) show that RL-based path selection trained in combination with *Transform-Coke* consistently yields better results than performing search only (Minerva) or than providing randomly sampled paths as context. The *Transform-InterEnt* extension to *Transform-CoKE* performs better for FB15k237 but worse for WN18RR than *Transform-CoKE*.

2 Related Work

Early approaches to representation-based link prediction were based on *knowledge graph embeddings* (Bordes et al., 2013). Predictions for e_y are done solely based on learned vector representations of r_q and e_x, which need to encode all relevant information. (See the survey by Rossi et al. (2021) summarizing such context-free embedding approaches for link prediction.) A more versatile approach is to employ neural models for *contextualized* link prediction, which allows for utilizing and combining the information of a wider context around the query entity e_x.

DeepWalk (Perozzi et al., 2014) applied a language-modeling approach to paths in a graph, obtaining static node embeddings. Then, RNN-based models were used to incorporate context of entities from KGs and obtain self-sufficient deep contextualized embeddings (Das et al., 2017; Guo et al., 2019), as well as an additional component to context-free embeddings (Wang et al., 2018).

Ever since the introduction of the Transformer (Vaswani et al., 2017) and specifically BERT architecture (Devlin et al., 2019), a multitude of NLP tasks and other fields has been benefiting from these approaches. The power of contextualizing embeddings with the attention mechanism for KG was shown by (Wang et al., 2019), where the authors introduced the CoKE model. To leverage non-linear context, graph convolution networks have also been applied to graph neighborhood of e_x (Shang et al., 2019; Vashishth et al., 2019; Bhowmik and de Melo, 2020).

Reinforcement learning has been exploited for the link prediction task, specifically for finding a path connecting a query entity with an answer (Xiong et al., 2017; Das et al., 2018; Godin et al., 2019). To the best of our knowledge, RL strategies have not yet been used to benefit a contextualized predictor.

3 Overview

Minerva (Das et al., 2018) is a RL-based approach to link prediction. It searches for paths in KG from a source entity to find and answer entity for a given incomplete query tuple. The last entity of the most probable path is considered the prediction. For ranking-based evaluation, the top-N most probable paths according to the RL agent are generated for evaluation. Using this ranking, evaluation metrics such as Hits@k and MRR can be calculated in a usual fashion. An LSTM-encoded traversal history and node embeddings are used to learn a policy with a policy gradient method REINFORCE (Williams, 1992).

CoKE (Wang et al., 2019) is a Transformer-based model for either embedding-based link prediction (trained on pure triples, not paths) or path query answering (PQA, considering longer paths). In PQA, the setting is that the model needs to recover the e4 from a path e_1,r_1,r_2,r_3,e_4, where e_4 is unacessible during prediction. PQA is essentially a multi-hop reasoning task, since the prediction is always made for the *last element of the path* (an entity). The problem specification in PQA relies on fixed prediction paths that cannot be changed. In contrast, our setting uses paths to enhance link prediction, i.e. fill in the *missing position of the triple*, where the model has the freedom to find additional paths to support its decision (either through sampling on the fly, or through RL).

We use two datasets for our experiments: **FB15k-237** and **WN18RR**. The former is a subset of Freebase (Bollacker et al., 2008), a collection of facts about real-world entities (e.g. celebrities, locations, events), whereas the latter stems from WordNet (Miller, 1992) and contains semantic relations between lexical units of the English language.

Dataset	#entities	#relations
FB15k-237	14,541	237
WN18RR	40,943	11

4 Experiments

In our experiments, we vary how context paths are retrieved from the graph. These paths are then given to one of pretrained Transformer models for entity prediction: *Transform-CoKE* with middle entities omitted and *Transform-InterEnt*, that uses the full path, including the middle entities. For a triple (e_x, r, e_y), a context path with N relational steps r_{xi},e_{xi} starting from e_y is produced according to the selected retrieval strategy. E.g., a unmasked input sequence for *Transform-InterEnt* with context length $N = 2$ then has the form e_x,r_q,e_y,r_{x1},e_{x1},r_{x2},e_{x2}, with 2 steps r_{xi},r_{xi} taken from e_y.

We compare three context path generation strategies: **sampling** generates a sampled context path (not informed by the query). **Minerva** takes the most probable path that Minerva has taken for the query tuple. **RL** obtains context returned by the RL agent trained in conjunction with the prediction model. The following chapter formalizes our approach to

Model Name	N	FB15k-237 H@1	 MRR	WN18RR H@1	 MRR
Minerva	3	0.1056	0.1516	0.3618	0.3942
Transform-CoKE + sampling	2	0.1781	0.2427	0.2208	0.2960
Transform-CoKE + Minerva	3	0.1758	0.2344	**0.3816**	0.4209
Transform-CoKE + RL	2	*0.2191*	*0.2910*	0.3674	***0.4312***
Transform-InterEnt + sampling	3	0.2238	0.3040	0.2454	0.3065
Transform-InterEnt + Minerva	3	**0.2242**	**0.3041**	0.2498	0.3095
Transform-InterEnt + RL	3	0.2241	0.3040	*0.3036*	*0.3552*

Table 1: Link prediction Hits@1 and MRR on **test** set for FB15k-237 and WN18RR. Bold denotes the best metric for a data set across all models, italic marks where our RL model yields best performance across different context generation strategies within a specific model variant for one data set.

RL-based context generation.

4.1 Pretraining paths

Let D be a set of original triples of form (e_x, r_q, e_y) where $e_x, e_y \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ is a relation. Similarly to Minerva, we introduce an inversed relation $r^{-1} \in \mathcal{R}^{-1}$ for every relation $r \in \mathcal{R}$ in order to provide the search models with access to all nodes during graph traversal. A reversed triple is added for every triple during both training and evaluation, resulting in an extended set $D' = D \cup \{(e_y, r_q^{-1}, e_x) | (e_x, r_q, e_y) \in D\}$. This way, the first position is always the masked one (for head or tail prediction), while context generation starts from the last position for any triple from D'. Pretraining paths are sampled randomly from a graph constructed from the train triples. For a detailed description of sampling process see Appendix A.

As a result, a set of yet unmasked chains is obtained, with the following format: $c = (e_x, r_q, e_y, r_{x1}, e_{x1}, \ldots, r_{xN}, e_{xN})$ that are used directly as input for *Transform-InterEnt*. For *Transform-CoKE*, middle entities are omitted to fit the expected input structure of a CoKE model $c = (e_x, r_q, r_{x1}, \ldots, r_{xN}, e_{xN})$.

4.2 Pretraining of contextualized predictors

In its essence, the task of link prediction is equivalent to that of masked language modeling: the model learns to recover a masked element in a sequence of items stemming from a limited vocabulary, specifically the first entity is masked to then be predicted in a sampled chain c from KG. Despite that only entities appear in the masked positions and should be predicted, the predictor's vocabulary comprises $V = \{\mathcal{E} \cup \mathcal{R} \cup \mathcal{R}^{-1}\}$ both entities and relations[1] as they are treated as equal elements of a sequence (same as nouns and verbs are not separated in BERT). Same vocabulary is used in the RL-search component to allow for a direct use of its output paths as input to the predictor.[2]

In the pretraining phase, the scorer is optimized to correctly recover a masked entity for a sampled path. Let $\mathbf{h}_k^j(c) \in \mathbb{R}^d$ be the hidden Transformer representation of the *k-th* position in the *j-th* layer obtained for an input chain c. Then, the pretraining objective can be written as

$$L(c) = CrossEntropy(PredHead(\mathbf{h}_0^L(c)), c_0),$$

where L is the last encoding layer, c is an unmasked input path (chain) with the expected entity c_0 in the first position, and *PredHead* is a one of the predictor-specific final decision layers *PredHead*: $\mathbb{R}^d \mapsto \mathbb{R}^{|V|}$, following the source implementations.[3] Hyper-parameter choices, such as the number of layers and the number of epochs, are modeled after the PQA setup in the CoKE paper.[4]

4.3 Training of the RL context selection

On a high level, an RL agent selects the most probable *action* according to a learned *policy*. A positive feedback (a *reward*) reinforces a beneficiary choice of the agent by making all actions that led to a rewarded state more probable in the future. In our case, the RL agent chooses which step in a KG to take next based on the current position, previous steps

[1] The vocabulary further includes BERT-specific tokens (MASK, CLS, UNK and SEP) that are omitted here.

[2] To account for compatibility of Minerva-generated paths with the transformer predictor, we use the UNK token as a NO_OP (no operation, stay in the current graph node) token.

[3] For *Transform-InterEnt*, the HuggingFace (Wolf et al., 2020) implementation for BERT is used, *PredHead* corresponds to an `BertOnlyMLMHead`; for *Transform-CoKE*, it is the last *FF* layer in the original terminology.

[4] Adjusting the hyperparameters to follow the setup of link prediction model, i.e. increasing the number of Tranformer layers and epochs, did not yield better performance.

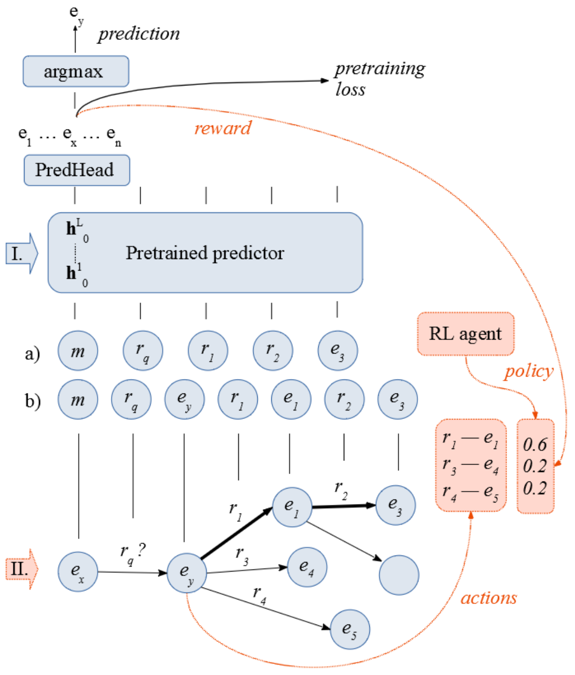

Figure 1: Workflow of the contextualized scorer component (I.) and the RL context-search (II.). Bold arrows represent the edges selected by the RL agent. r_q marks the query triple, m stands for the masked token. a) and b) stand for two path representations used by *Transform-CoKE* and *Transform-InterEnt* respectively.

taken and the query. It has the same structure and hyper-parameters as the one in Minerva (based on our reimplementation)[5], except for the reward function that now aims for a good contextualized prediction, rather than finding the correct answer.

The RL agent is trained separately, and uses the already pretrained transformer during training.[6] The modified reward of the RL agent can be characterized using following terminology: $S_t = (e_t, r_q, e_y)$ is the current state of the agent, where e_t is the entity at time step t, r_q is the relation and e_y is the tail entity of the query; c_t is the chain of KG-steps traversed by the agent. The reward at final time step N then equals

$$R(S_N) = softmax(PredHead(\mathbf{h}_0^L(c_N)))_i,$$

where *PredHead* returns the logits over all possible entities of a deep scorer and i is the vocabulary index of the correct answer entity e_x. The summary of the joint architecture is illustrated in Figure 1. Our PyTorch implementation of the RL agent is available at `https://github.com/marina-sp/reinform`.

4.4 Results and Analysis

We evaluate the models with mean reciprocal rank (MRR) and Hits@1. Table 1 shows that the prediction-based approach generally outperforms the search-based approach of Minerva for FB15k-237. Which path generation mechanism is used makes as marked difference for *Transform-CoKE*, and training the RL model specifically for this setting performs best by a large margin. Including intermediate entities in the path processed by the transformer (*Transform-InterEnt*) again increases the performance for all path search strategies (but the differences between them disappear).

Using learned context paths over random sampling is generally beneficial in case of WN18RR. With a path for a test query `(tog VB1, derivationally related form`$^{-1}$`, dresser NN3)` extracted by the RL agent `(dresser NN3, derivationally related form`$^{-1}$`, get dressed VB1, derivationally related form, dresser NN3)` the correct prediction was scored the highest by the *Transform-CoKE*, whereas with a randomly sampled one `(dresser NN3, hypernym, supporter NN3, hypernym`$^{-1}$`, hatchet man NN2)`, the expected entity `tog VB1` was ranked 2267. The RL-path also exemplifies the ability of the model to generalize beyond the entities contained in a path, which Minerva can not per definition.

5 Conclusion

We have shown how to combine path selection by reinforcement learning with transformer-based prediction models for contextualized link prediction in knowledge graphs. This approach achieves strong perfomance gains over a recent previous RL model that directly searches for an answer in the graph. Analysis indicates that this performance gain is presumably due to the fact that answer entities often do not lie on paths found by RL, and need instead be predicted from the entire pool of entities (not constrained to entities reached on a path). Our method also shows gains over using the transformer-based prediction models on paths randomly selected from around the query entity. This shows the potential of reinforcement learning to benefit prediction models that rely on path selection.

[5] See Section 2 of (Das et al., 2018) for a formal description of the agent. We were able to reproduce the results reported in (Das et al., 2018) with our reimplementation. In the original Minerva paper, however, only $(e_x, r_q, ?)$ queries are used for evaluation, i.e., it is only attempted to predict the object, see Section 3.1 "KB query answering" of (Das et al., 2018). In our experiments we evaluate on subject and object positions, and Minerva results drop significantly in this more challenging setting.

[6] We also experimented with jointly optimizing the prediction and ML module, but ran into stability issues, yielding worse results.

References

Rajarshi Bhowmik and Gerard de Melo. 2020. A joint framework for inductive representation learning and explainable reasoning in knowledge graphs. *CoRR*, abs/2005.00637.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141, Valencia, Spain. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fréderic Godin, Anjishnu Kumar, and Arpit Mittal. 2019. Learning when not to answer: a ternary reward structure for reinforcement learning based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 122–129, Minneapolis, Minnesota. Association for Computational Linguistics.

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. *CoRR*, abs/1905.04914.

Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652.

Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge graph embedding for link prediction. *ACM Transactions on Knowledge Discovery from Data*, 15(2):1–49.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3060–3067.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *CoRR*, abs/1911.03082.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Haoyu Wang, Vivek Kulkarni, and William Yang Wang. 2018. DOLORES: deep contextualized knowledge graph embeddings. *CoRR*, abs/1811.00147.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019. Coke: Contextualized knowledge graph embedding. *CoRR*, abs/1911.02168.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Appendix

A. Pretraining path sampling

For *Transform-InterEnt*, a pretraining path, or *chain* c, with a fixed amount of steps K is sampled by randomly travesing the graph starting from e_y. A step from e_y is a single outgoing edge described by the labels of the respective edge and target node (r_{xi}, e_{xi}). The same $N = K$ is used when retrieving the context with an RL agent. The underlying graph consists of triples from the training set alone. The query triple (e_x, r, e_y) itself as well as the backward connection (e_y, r^{-1}, e_x) are excluded from the sampling process to resemble the evaluation process, where the query triple is not available during graph traversal. The described process is equvalent to the *sampling* strategy.

For *Transform-CoKE*, the set of pretraining paths has mixed lengths $1 <= K <= 5$ following the original implementation. The length of context N during evaluation, i.e. the number of steps taken by the RL agent, is however constant. For both *Transform-InterEnt* and *Transform-CoKE*, N is treated as a hyper-parameter that is determined based on the development data.