Relational learning on knowledge graphs

Marek Toma, Vojtěch Kalivoda

1 Introduction

Knowledge graphs (KGs) are knowledge bases with a graph structure, where the entities are represented as nodes and relations between them with edges; typically KGs consist of different node and edge types. KGs have been widely used in biomedical domain for knowledge representation and discovery.

In this project we aim to design a model for relational learning on biomedical KGs with goal of drug re-purposing and side effect prediction. We propose a self-supervised pre-training of the model on task of graph reconstruction. Which showed a slight improvement in performance and stability of the model in downstream task of link prediction.

2 Related work

Zitnik et al.[1] designed a GCN-based model for relational learning on knowledge graphs, with the goal of predicting polypharmacy side effects. In their work, Huang et al.[2] used self-supervised training to pre-train a language model for common sense knowledge graph embedding, where their treated subgraphs as sentences and used the model to predict original sentences from corrupted ones. Liu et al.[3] used contrastive learning to train a model for concept linking across knowledge graphs. In their survey paper, Liu et al.[4], described various techniques for self-supervised learning on graphs.

3 Dataset

The dataset used in our project is the Hetionet[1] biomedical knowledge graph. It consists of 11 node types (e.g. Gene, Disease, Biological process, Compound) and 24 relationship types (e.g. Compound-binds-Gene, Disease-resembles-Disease, Gene-participates-Biological process); where the total number of nodes is 47 031 with 2 250 197 edges.

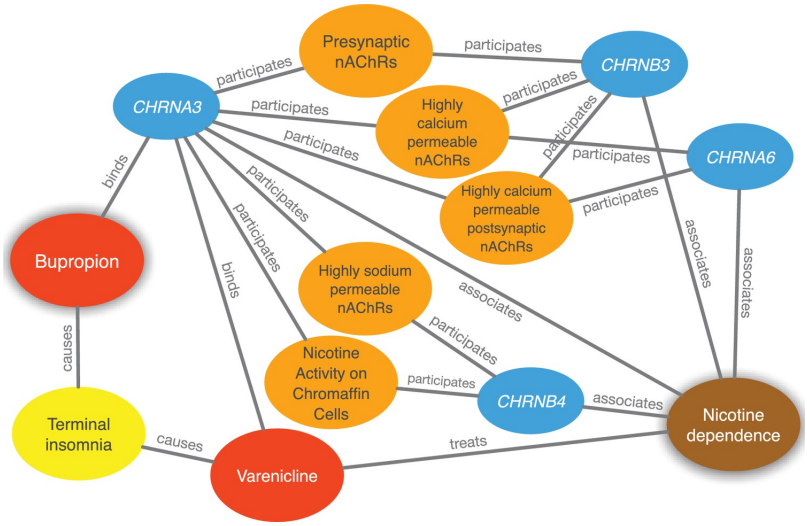

Figure 1: A snippet of the Hetionet knowledge graph. The image source [5].

[1] Available at https://het.io .

4 Model

Our model consists of an encoder, a pretext decoder used in pre-training, and a downstream decoder used for relation prediction. The encoder is given a knowledge graph with some initial node embeddings and transforms them into new node embeddings. The pretext decoder is given a perturbed graph and node embeddings and it tries to reconstruct its connectivity tensor. The downstream decoder is given a pair of node embeddings and predicts the existence of task-specific-relationship between them.

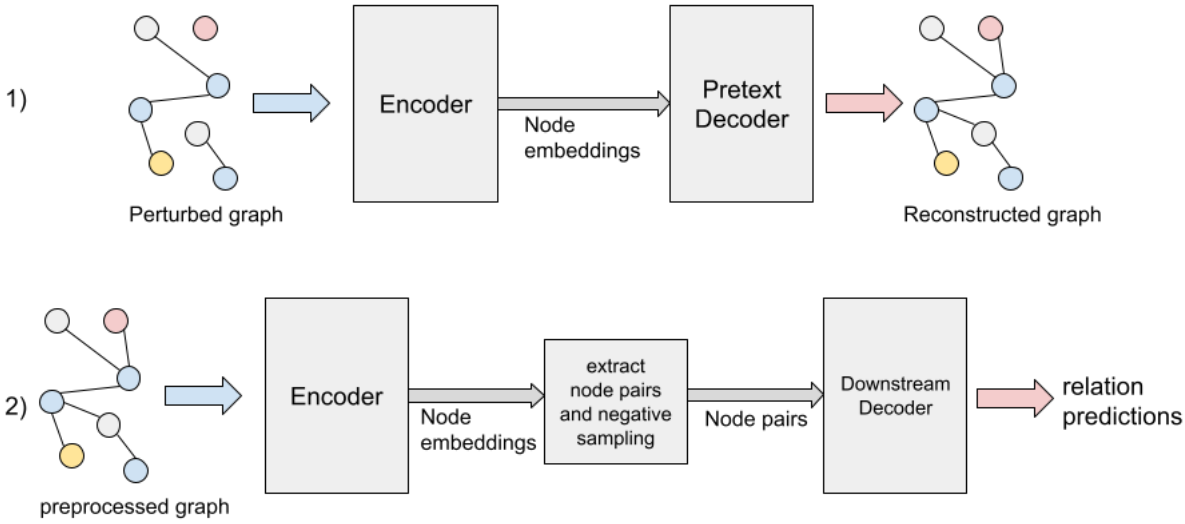

Figure 2: 1) Our pre-training pipeline where the encoder generates node embeddings from randomly perturbed graph and the decoder tries to reconstruct the original graph. 2) The downstream pipeline where the encoder generates node embeddings, from which we create node task-dependant node pairs (either (compound, disease) or (compound, side effect) pair). We also generate negative samples via a corruption of positive samples.

4.1 Encoder

The encoder is a graph convolutional network comprised of SAGEConv[6] layers. The general framework of the forward pass of these layers:

1. For each node i we aggregate node representation from its neighborhood (either initial embeddings or embeddings from the previous convolutional layer), this is done for each relation type r separately.
   $h_{\mathcal{N}(i,r)}^{(l+1)} = aggregate(h_j^l, \forall j \in \mathcal{N}(i,r))$
2. Next we concatenate the aggregated information with the embedding of node i and multiply it with the parameter matrix for relation r.
   $h_i^{(l+1)} = \sigma(W_r * concat(h_i^l, h_{\mathcal{N}(i,r)}^{(l+1)}))$
3. Afterwards a normalization might be applied.
   $h_i^{(l+1)} = norm(h_i^{(l+1)})$
4. Lastly we aggregate the outputs for all relation-specific neighborhoods and obtain new embedding for node i.

4.2 Decoders

The pre-text decoder consists of one feed-forward network for each relation type. Each of these networks predicts the probability of a relationship (of a given type) between two given nodes.

The downstream decoder consists of a single feed-forward network for the final prediction task; either predicting treats/palliates/none between disease and compound pairs or the existence of side effects between disease and side effect pairs.

5 Methodology

Our approach consists of two training parts: a pretext task aimed at training the encoder and pretext decoder, and a downstream task focused on training the encoder and downstream decoder. In the preprocessing stage, we removed a portion of relations that will be used for training and testing of the downstream decoder.

The pretext task is a reconstruction of a perturbed graph, which is obtained by dropping half of the edges that are passed to the encoder that generates node embeddings. These embeddings are passed to the pretext decoder that predicts the existence of the edges and tries to reconstruct the original graph.

The Adam optimizer with binary cross entropy loss function was used and the model was trained for 150 epochs.

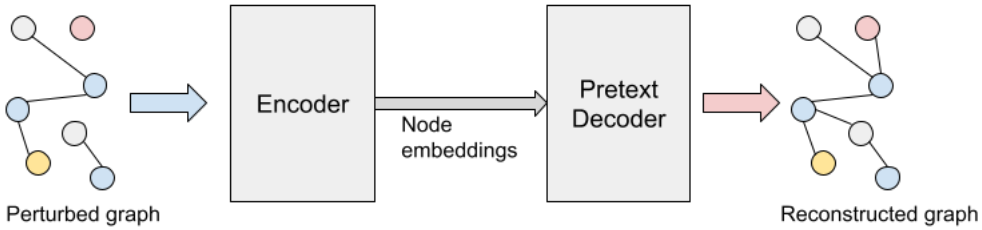

Figure 3: Reconstruction of perturbed graph

The downstream task consists of two link prediction tasks. During training, the compound-disease or compound-side effect pairs are generated; the positive pairs are generated from the portion of known relations that have been removed from the graph in preprocessing. The negative pairs are generated by random corruption of the positive pairs (such that the corrupted pair is not present in the original graph).

Pairs are then passed to the encoder that generates node embeddings, which are further fed to the downstream decoder that predicts the relations of the pairs. For compound-disease possible predictions are treats, palliates, or none, for compound-side effect pairs the prediction are causes or none.

During training, the Adam optimizer with cross-entropy loss function was used and the model was trained until validation f1 score stopped increasing.

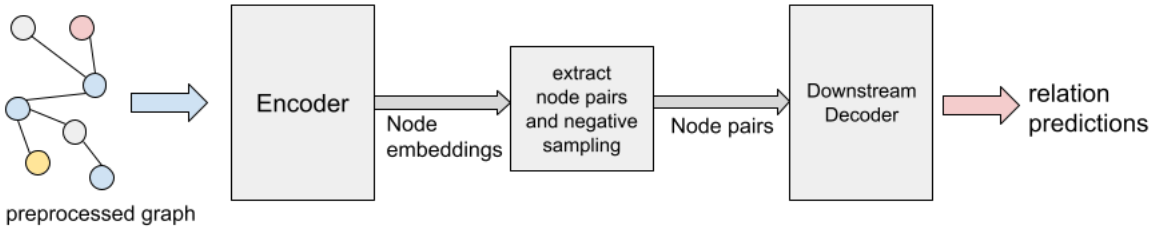

Figure 4: Prediction of relations between compound-disease pairs

Model	Accuracy	Precision	F1
TransE	0.64	0.66	0.63
GCN	0.81	0.84	0.80
pre-trained GCN	0.82	0.86	0.82

Table 1: Summarization of results of prediction task for compound-disease pairs.

6 Evaluation and results

We trained and evaluated our model both, with and without pre-training and compared it with TransE[2][7].

For compound-disease prediction task our pre-trained model slightly outperformed the non-pre-trained variant, and both of them outperformed the baseline, the results are summarized in Table 1. Most of the miss-classification our model made were that it predicted that a compound treats/palliates a disease for unknown pairs (as seen in confusion matrix), which is the desirable outcome for the purpose of drug re-purposing.

For compound-side effect prediction we were not able train the model to give any meaningful predictions, the model kept predicting none for all the samples.

7 Conclusion

Our pre-training approach slightly improved the performance in prediction task on (compound, disease) pairs, while also made the model more stable; the non-pre-trained model tended to have higher difference in f1 score between validation and testing dataset. We were not able to train a meaningful model on (compound, side effect) pairs, this might be because the data do not contain sufficient amount of relevant information for the model to make such predictions.

References

[1] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, pp. i457–i466, jun 2018.

[2] J. Huang, Y. Du, S. Tao, K. Xu, and P. Xie, "Structured self-supervised pretraining for commonsense knowledge graph completion," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1268–1284, 2021.

[3] X. Liu, L. Mian, Y. Dong, F. Zhang, J. Zhang, J. Tang, P. Zhang, J. Gong, and K. Wang, "Oag_{know} know : Self-supervised learning for linking knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1895–1908, 2023.

[4] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5879–5900, 2023.

[5] D. S. Himmelstein, A. Lizee, C. Hessler, L. Brueggeman, S. L. Chen, D. Hadley, A. Green, P. Khankhanian, and S. E. Baranzini, "Systematic integration of biomedical knowledge prioritizes drugs for repurposing," *eLife*, vol. 6, p. e26726, sep 2017.

[6] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[2]We used implementation available in Deep Graph Library: https://docs.dgl.ai/index.html .

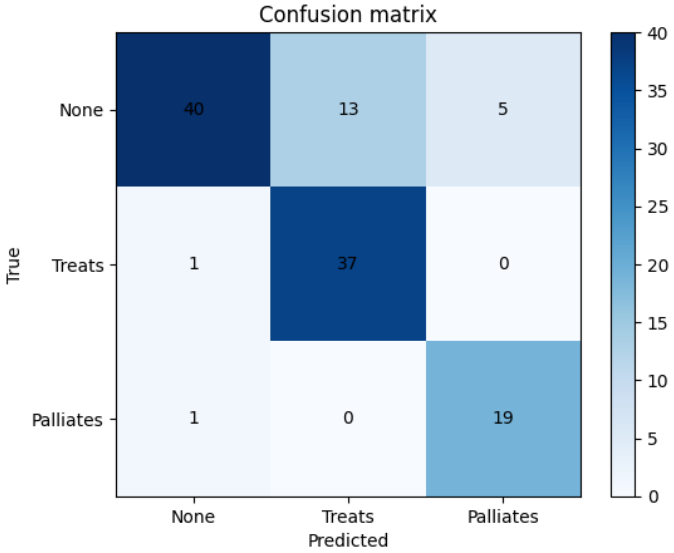

Figure 5: Confusion matrix of our pre-trained model for compound-disease pairs.

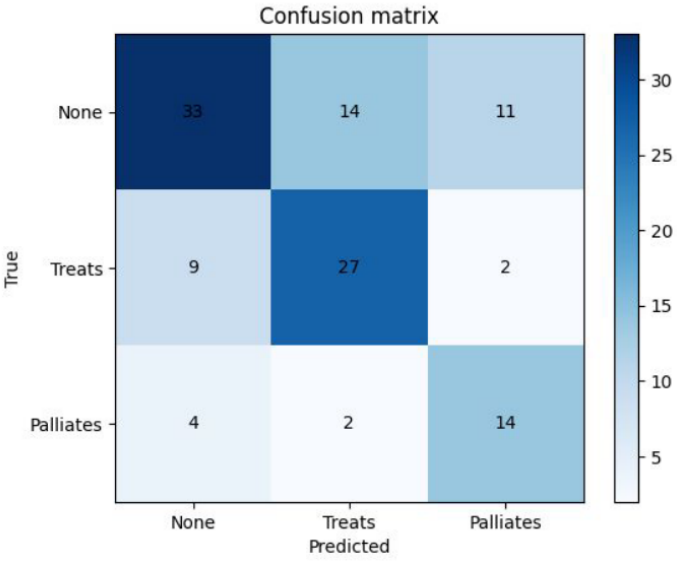

Figure 6: Confusion matrix of TransE model for compound-disease pairs.

A Instalation and startup instructions

The project contains two jupyter notebooks, disease_compound.ipynb and side_effect.ipynb, that contain hyper-parameters and full pipeline for our models (from loading of the data to evaluation), for (compound, disease) link prediction task and (compound, side effect) link prediction task respectively. The definitions of data loading, training, and other helper functions are contained in RelationalLearning/Dataloader.py and RelationalLearning/utils.py files. The non-preprocessed dataset is contained in data/.

To run the solution python 3.6+ is required along with these packages: dgl, numpy, torch, sklearn, and matplotlib.