Better Together: Enhancing Generative Knowledge Graph Completion with Language Models and Neighborhood Information

Alla Chepurova[1] **Aydar Bulatov**[1] **Yuri Kuratov**[1,2] **Mikhail Burtsev**[3]

[1]Neural Networks and Deep Learning Lab, MIPT, Dolgoprudny, Russia
[2]AIRI, Moscow, Russia
[3]London Institute for Mathematical Sciences, London, UK
chepurova@deeppavlov.ai, {bulatov.as,yurii.kuratov}@phystech.edu, mb@lims.ac.uk

Abstract

Real-world Knowledge Graphs (KGs) often suffer from incompleteness, which limits their potential performance. Knowledge Graph Completion (KGC) techniques aim to address this issue. However, traditional KGC methods are computationally intensive and impractical for large-scale KGs, necessitating the learning of dense node embeddings and computing pairwise distances. Generative transformer-based language models (e.g., T5 and recent KGT5) offer a promising solution as they can predict the tail nodes directly. In this study, we propose to include node neighborhoods as additional information to improve KGC methods based on language models. We examine the effects of this imputation and show that, on both inductive and transductive Wikidata subsets, our method outperforms KGT5 and conventional KGC approaches. We also provide an extensive analysis of the impact of neighborhood on model prediction and show its importance. Furthermore, we point the way to significantly improve KGC through more effective neighborhood selection.

1 Introduction

Knowledge graphs (KG) represent structured information in the form of a directed multi-relational graph with entities as the graph's nodes and relations between entities as the graph's edges. Entities in such graphs could be both real-world objects or abstract concepts. KGs are represented as a set of triplets with representation entities and a relation between them, i.e. (head entity, relation, tail entity) or (h, r, t) for short. Due to KGs' recent rapid development, they are now widely used in a variety of applications, including data mining, information retrieval, question answering, and recommendation systems. However, most real-world knowledge graphs suffer from incompleteness. Knowledge graph completion (KGC) algorithms attempt to fill in the gaps in the KG in order to solve this problem.

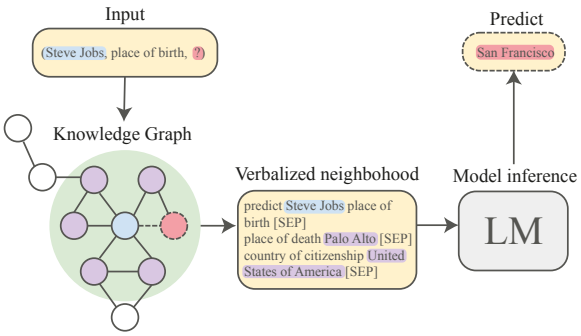

Figure 1: **Graph completion pipeline.** We extract and verbalize neighborhood of a triplet head node from the Knowledge Graph (KG). Language Model uses verbalized input triplet and neighborhood to make predictions. Predictions can be used for question answering or KG completion.

KG link prediction commonly involves learning knowledge graph embeddings (KGE), which capture structural and semantic information about KG nodes, relations, and their neighborhoods in a way that a simple distance-based function would be able to distinguish real triplets from false ones. However, building a unique embedding for each node and relation leads to a linear increase in model size and inference time as the KG size grows. Another drawback is that KG updating requires full re-training of KGEs, rather than simply learning new embeddings for updated entities or relations.

Application of language models (LMs) to KGC tasks is a promising direction (Pan et al., 2023). State-of-the-art performance in most natural language processing (NLP) tasks has recently been demonstrated by pre-trained language models like BERT (Devlin et al., 2019) and more recent large or foundational language models (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023). Language models-based approaches, such as KG-BERT (Yao et al., 2019), BERTRL (Zha et al., 2022), KGT5 (Saxena et al., 2022) and GenKGC (Xie et al., 2022) consider a triplet in the KG as a text sequence. While distance-based

Findings of the Association for Computational Linguistics: EMNLP 2023, pages 5306–5316
December 6-10, 2023

methods incorporate more structural information in dense embeddings, language models tend to capture more semantic information contained in both relations and entities of KG, enabling such models to embed previously unseen entities and operate on new facts based on semantic information contained in triplets textual representation. Thus, KGT5 positions KG link prediction as sequence-to-sequence tasks. KGT5 model trained to generate target node using textually verbalized triplets achieves competitive performance and offers a scalable and simple framework for KGC and QA tasks.

Current methods based on language models (KG-BERT, BERTRL, KGT5, and GenKGC) rely only on information that can be extracted from LMs parameters. However, since KG itself is available, we build a method that relies on both internal language model knowledge and external information from KG. For the link prediction task (h, r, ?), we propose to extract nodes and relations adjacent to h from the KG, i.e. the neighborhood. Node h with its neighborhood *better together* represents h itself: neighborhood may contain clues such as Palo Alto and United States of America for the triplet (Steve Jobs, place of birth, ?) on Figure 1.

Though some efforts have been made to incorporate neighborhood into the KGC algorithms (Chen et al., 2020), our proposed pipeline inherits the benefits of generative LLMs compared to KGE approaches: (1) scalability and size of the model, (2) applicability to both transductive and inductive KGs due to an ability to generalize unseen entities, (3) no need to rank all possible candidate triplets due to direct generation of tail entity. Also, a concurrent work by the KGT5 authors, KGT5-context (Kochsiek et al., 2023), proposed a similar idea of integrating node neighborhood in the context of the generative LM model, supporting the main concerns and results of our study.

Our contributions are as follows: (1) we propose a generative graph completion method which effectively integrates language models with graph neighborhood information. (2) We show that incorporating graph information into multiple generative language models achieve strong performance improvement of link prediction in both transductive and inductive settings. Moreover, our approach achieves state-of-the-art results on the inductive ILPC dataset. (3) We analyze and evaluate the impact of adding neighborhood information on the performance of the models.

2 Methods

One can represent KG more formally as $\mathcal{G} = (\mathcal{T}, \mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations and $\mathcal{T}$ is a set of triplets $\mathcal{T} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Therefore, the formulation of the KGC task is the following: for each triplet $\mathcal{T}_i = (h, r, t)$, where $h \in \mathcal{E}, t \in \mathcal{E}, r \in \mathcal{R}$, answer both $(h, r, ?)$ and $(?, r, t)$ queries on the graph $\mathcal{G}/\mathcal{T}_i$. The KGC tasks can have different variations depending on the KG type: inductive and transductive. In a transductive setting the KGC task is formulated as follows: given a train KG $\mathcal{G}_{train} = (\mathcal{T}_{train}, \mathcal{E}, \mathcal{R})$ train the model to predict on the inference KG $\mathcal{G}_{inference} = (\mathcal{T}_{inference}, \mathcal{E}, \mathcal{R})$, therefore in the train and inference KG there is a same set of entities and relations. In inductive KGC the model is trained on the KG $\mathcal{G}_{train} = (\mathcal{T}_{train}, \mathcal{R}, \mathcal{E}_{train})$ and predicts the inference KG $\mathcal{G}_{inference} = (\mathcal{T}_{inference}, \mathcal{R}, \mathcal{E}_{inference})$, so the trained model observed only on the entities from train subgraph of KG.

We followed the KGT5 (Saxena et al., 2022) approach to the KGC task using an encoder-decoder Transformer based on T5 (Raffel et al., 2020). The link prediction task was posed as follows: for each triplet $(entity_1, relation, entity_2)$ from KG we had to predict the tail in the queries $(entity_1, relation, ?)$ and $(entity_2, relation^{-1}, ?)$. To generate input for the model, we converted such queries into a text form. Apart from the query verbalization only as in KGT5, we formed a neighborhood for the node and relations for each triplet in KG. Thus, verbalized queries and their neighborhoods in the form of text were fed to the input of the model for training (see Figure 1) and later to predict incomplete triplets. We form a neighborhood of nodes that have an edge with the triplet head node (1-hop neighbors). The number of adjacent nodes may be too large to fit into the context size of selected language models. Therefore, we sort them based on relation semantic similarity. We take only those verbalized nodes and relations that fit into the max sequence length. We provide more details on neighborhood formation in Appendix B and verbalization in Appendix C.

To evaluate our approach to link prediction tasks we used publically available transductive and inductive KGs. Specifically, we used the Wikidata5M (Wang et al., 2021) dataset, which is one of the largest publicly available transductive KG up-to-date and large and small versions of ILPC

dataset (Galkin et al., 2022), the largest fully inductive benchmark. ILPC inference graph size is challenging for modern GNN methods with inductive capabilities (Galkin et al., 2022). Apart from being challenging to solve by common approaches, the ILPC KGs are much more sparse than the transductive Wikidata5M. The graph size and sparseness of the data make the task hard for existing methods and result in modest performance metrics. Still, there were no existing solutions that beat the baseline provided in the ILPC paper. Both used datasets are the subsets of the real-world KG – Wikidata. Datasets' statistics are provided in the Table 4).

We compared models with and without neighborhoods to evaluate our hypothesis that neighborhoods help to *better together* represent graph node. In addition, unlike the KGT5, we experimented with different model configurations, both fine-tuning an existing pre-trained model and training from scratch as common knowledge obtained during language models pre-training might be useful on real-world knowledge graphs like Wikidata.

We used the T5-small with 60M parameters as our base model with both pretrained and randomly initialized weights to evaluate the effect of pretraining of the language model on the KG task. We added additional "[SEP]" tokens to the tokenizer with 30K tokens vocabulary to separate query and neighborhood. The model was trained using AdamW optimizer (Loshchilov and Hutter, 2019), 1e-5 learning rate, 10% dropout, and batch size equals 320. The number of training steps was equal to 5M for the Wikidata5M dataset and 2M for all others. For multi-gpu training, we used the Horovod parallelization framework (Sergeev and Balso, 2018). In cases where a batch size equal to 320 did not fit GPU, we used gradient accumulation steps. We also benchmarked OpenAI `gpt-3.5-turbo`[1] zero-shot link prediction and evaluate the benefits of a neighborhood on LLM (see Appendix G). The code for all the described experiments is available at the link[2].

As KGT5, our model produces a probability distribution over all possible tokens in the vocabulary at each step of decoding. Such distribution is penalized for not matching the actual distribution, i.e., for the difference between the real target token at position i and the generated token at step i using CE loss. Such an approach is principally different from distance-based methods as it does not require generating negative samples to make them further from the target prediction for each training query. In this way, training iteration becomes independent of the KG size.

To form predictions, we first verbalize the query $(head, relation, ?)$ and pass it as an input to the generative model to produce an output sequence, which will be a verbalization of the target entity id. For inference, we employed two strategies. The first one was sampling k most probable sequences from the decoder. We follow the KGT5 (Saxena et al., 2022) setting and use a sample size of 50. We apply this approach to Wikidata5M datasets as it is transductive. The second option for inference was to search for the closest entities in KG to the generated one. We apply the second approach to ILPC inductive datasets as language models have to generate previously unseen entities. More details on inference procedure are in Appendix D.

3 Results

The hypothesis behind our approach was that neighborhood information incorporated in a context fed to the language model should enhance link prediction of generative LM. Tables 1 and 2 demonstrate the performance of our technique on Wikidata5M and ILPC datasets under various conditions, including training a T5 model with and without neighbors, using a pre-trained model or training a model from scratch, and zero-shot with `gpt-3.5-turbo`.

These results support the hypothesis that the quality of link prediction increases with the use of neighbors in the context of the generative model in both transductive and inductive settings. Moreover, our approach demonstrated SOTA performance on the ILPC dataset, while on Wikidata we achieved the best result among generative link prediction models, which is still comparable to scores of SOTA graph and LM methods with many more parameters. Relatively small absolute values of scores on the ILPC dataset can be explained by its fully inductive setup and larger sparseness compared to Wikidata5M. Still, there are no publicly available solutions that exceed the baseline score provided by ILPC authors. Our solution overperforms the existing baseline, is elegantly simple, and more interpretable than GNN-based solutions. Experiments on pre-trained models revealed that pre-training speeds up convergence but degrades

[1] `gpt-3.5-turbo-0613`: https://platform.openai.com/docs/models/gpt-3-5

[2] https://github.com/screemix/kgc-t5-with-neighbors

Table 1: Adding a neighborhood persistently improves results on the link prediction task. We report Hits@k metrics on Wikidata5M, ILPC-small, and ILPC-large test sets. Arrows ↑↓ show the effect of the neighborhood on language models. Neighbors allow to achieve the best performance considering the number of model parameters.

Model	Wikidata5M			ILPC-small			ILPC-large			
	H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10	#params
KGT5 (Saxena et al., 2022)	0.267	0.318	0.365	-	-	-	-	-	-	60M
TransE (Bordes et al., 2013)	0.170	0.311	0.392	-	-	-	-	-	-	2,400M
ComplEx (Trouillon et al., 2016)	0.255	-	0.398	-	-	-	-	-	-	614M
KEPLER (Wang et al., 2021)	0.173	0.224	0.277	-	-	-	-	-	-	125M
SimKGC (Wang et al., 2022)	0.313	0.376	0.441	-	-	-	-	-	-	220M
IndNodePiece (Galkin et al., 2022)	-	-	-	0.007	0.022	0.092	0.037	0.054	0.125	15.5K
IndNodePieceGNN (Galkin et al., 2022)	-	-	-	0.076	0.140	0.251	0.032	0.073	0.146	24K
T5-small not pre-trained	0.240	0.286	0.323	-	-	-	-	-	-	60M
T5-small not pre-trained + neighbors	0.327↑	0.363↑	0.386↑	-	-	-	-	-	-	60M
T5-small pre-trained	0.205	0.248	0.282	0.073	0.117	0.177	0.112	0.137	0.194	60M
T5-small pre-trained + neighbors	0.306↑	0.342↑	0.361↑	0.076↑	0.137↑	0.209↑	0.114↑	0.139↑	0.200↑	60M
T5-base pre-trained	-	-	-	0.073	0.130	0.199	0.110	0.136	0.198	220M
T5-base pre-trained + neighbors	-	-	-	0.082↑	0.142↑	0.216↑	0.116↑	0.141↑	0.215↑	220M

Table 2: Neighbor information improves zero-shot link prediction with OpenAI `gpt-3.5-turbo-0613`. *ILPC-Large results were obtained on 5000 random samples from the test set.

Dataset	GPT		GPT+NEIGH	
	H@1	H@3	H@1	H@3
Wikidata5M	0.098	0.131	0.113↑	0.157↑
ILPC-small	0.079	0.107	0.084↑	0.118↑
ILPC-large*	0.103	0.134	0.114↑	0.150↑

the quality of the model by a few points compared to the one trained from scratch.

Also, we demonstrated that neighbors' information enhances performance across different generative models. This includes open-source models such as T5, as well as proprietary models like OpenAI GPTs that are distributed as a model-as-a-service. Although both types of models have their own advantages and limitations, it should be noted that OpenAI GPTs are black box models that cannot be modified or interpreted further. Usage of OpenAI GPTs is restricted in the case of more specialized or proprietary KGs that can store secure or non-public information. In the context of our study, OpenAI's `gpt-3.5-turbo` has a much larger size and cost, compared to smaller and transparent T5, still our approach improves the performance of both models.

To get an understanding of how proposed models learn to use neighboring triplets we compared predictions of models trained with and without graph information in the input. One possible case when neighbors help in link prediction is when a target entity is somehow mentioned in the model input. Because of neighborhood filtering, the target triplet never appears in the model input, however in some cases target entity can occur with different relations or as a substring of another entity in a neighborhood or even in the task itself (Table 3). Despite the tail appearing in input only in a small portion of the test set, the neighborhood model learns to make use of such hints, outperforming the baseline T5 in all selected cases. The high solution rate of more than 4 out of 5 samples with hinted targets supports the importance of properly selected neighborhoods.

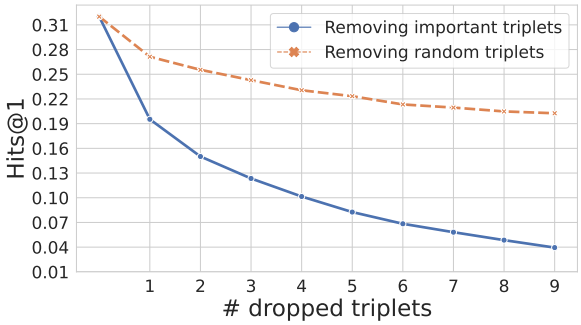

Figure 2: **KG neighborhood is critical for the model performance.** The plot shows how the quality of link prediction depends on the number of neighborhood triplets removed from the input context Wikidata5M T5-small not pre-trained model. The blue solid line corresponds to removing the most relevant triplets and the dashed red line represents random selection.

To evaluate the impact of neighborhoods, we analyzed triplets' importance by sorting them based on how they affect the probability of the target node prediction. As Figure 2 shows, a performance of the model quickly deteriorates with the removal of important triplets from the context. This implies potential avenues for improvement by prioritizing significant triplets selection prior to model training. Moreover, we analyzed attention maps of the model train on Wikidata5M and found that it indeed pays attention to relevant neighbors, e.g. Appendix F.

The analysis shows that adding relevant graph

Table 3: Adding relevant neighbors improves link prediction for T5-small on the Wikidata test set. The table shows the exact match depending on a hint of the target entity's appearance as a part of the model input.

Target position	T5	T5 + Neighbors	frequency
In task	0.93	0.94 ↑	5 %
In neighborhood	0.34	0.82 ↑	8 %
Not in input	0.18	0.24 ↑	87 %
Total	0.23	0.32 ↑	100 %

context information greatly enhances link prediction. Hence one way of improving performance is increasing neighborhood size by adding k-hop neighbors. Due to the quadratic complexity of attention and the limited input size of pretrained transformers, processing exponentially growing neighborhoods would be computationally expensive. To deal with large input sizes one can use efficient T5 architectures with input sizes up to 32k (Guo et al., 2022) and 64k (Ainslie et al., 2023) tokens or recurrent approaches (Bulatov et al., 2022; Hutchins et al., 2022) that can sequentially process effectively unlimited neighborhood sizes. Another potential way of subgraph selection is to add relevant but more distant knowledge graph information, such as random walks or more sophisticated methods (Das et al., 2018).

4 Conclusions and Future work

Our study found that incorporating KG neighborhood information into a generative model's context can enhance the quality of the link prediction task. We trained the T5-small model on transductive Wikidata5M and inductive ILPC datasets with various configurations and showed that coupling the node and its neighborhood information together better represents the node. Our approach significantly outperformed baselines on the largest real-world KGCs, with a comparable or fewer number of parameters. We also demonstrated that OpenAI's `gpt-3.5-turbo` performance on the KGC task also benefits from neighborhood information without prior fine-tuning. In addition to the quality improvement, we analyzed the impact of neighborhood imputation, highlighting the significance of selecting relevant triplets or extending the model's context in future works. Furthermore, future studies could explore the impact of model size, the effect of neighborhood imputation on the datasets from other domains, and neighborhood selection strategies.

Limitations

Limitations of this study include the training of only relatively small models, with only T5 being employed. Further study of the benefits of larger models applied to KGC tasks should be performed. Additionally, each knowledge graph in this study was limited to only one model, which may not accurately represent more complex graphs that involve multiple datasets. Another limitation is that language models are better suited for real-world textual data and may not work well for KGC representing more specialized concepts, such as biology or medicine. Also, although neighborhood selection was performed, we cannot guarantee that the selected neighborhoods are optimal. Further works should include the way of extending the ability of models to capture larger parts of a graph or selecting the most relevant neighbors. Moreover, constructing prompts for testing GPT performance may not be optimal, and the model's maximal performance cannot be guaranteed without knowledge of its training data.

Ethics Statement

As a field of study in the intersection of natural language processing and knowledge graphs, this work inherits the main ethical considerations of both of these domains. Automated graph completion tasks may produce various types of mistakes, including but not limited to factual and grammatical errors. Consequently, any application based on this work should employ robust error detection mechanisms prior to deployment. Additionally, given that our system employs a generative language model, there exists a risk of generating false or misleading content such as hallucinations and biases existing in pre-training and knowledge graph data. Users of the proposed method or any application based on it should be warned about the potential risks of using generative language models. Potential misuses may include purposeful disinformation and misleading language or factual information. Taking into account the aforementioned considerations, we believe that under ethically responsible usage this work will have a positive impact on artificial intelligence research and applications.

Acknowledgements

We are grateful to Mikhail Galkin for valuable advice on knowledge graphs and introduction to the

ILPC dataset. We are thankful to SberDevices for granting us access to additional computational resources. A.C., A.B., and Y.K.'s work was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Moscow Institute of Physics and Technology dated November 1, 2021 No. 70-2021-00138.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. 2023. Colt5: Faster long-range transformers with conditional computation.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pages 11079–11091. Curran Associates, Inc.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2020. Hitter: Hierarchical transformers for knowledge graph embeddings. *arXiv preprint arXiv:2008.12813*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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 *International Conference on Learning Representations*.

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.

Mikhail Galkin, Max Berrendorf, and Charles Tapley Hoyt. 2022. An Open Challenge for Inductive Link Prediction on Knowledge Graphs. *arXiv preprint arXiv:2203.01520*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. In *Advances in Neural Information Processing Systems*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Adrian Kochsiek, Apoorv Saxena, Inderjeet Nair, and Rainer Gemulla. 2023. Friendly neighbors: Contextualized sequence-to-sequence link prediction. *arXiv preprint arXiv:2305.13059*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, Dublin, Ireland. Association for Computational Linguistics.

Alexander Sergeev and Mike Del Balso. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. *arXiv preprint arXiv:2203.02167*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. 2022. From discrimination to generation: Knowledge graph completion with generative transformer. In *Companion Proceedings of the Web Conference 2022*, pages 162–165.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Hanwen Zha, Zhiyu Chen, and Xifeng Yan. 2022. Inductive relation prediction by bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5923–5931.

A Input data pipeline

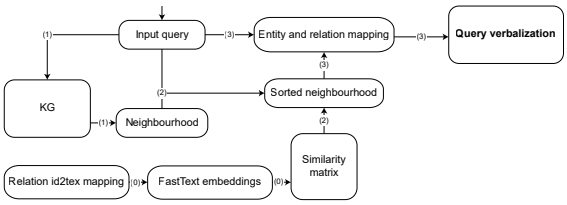

Figure 3: Visualization of the complete pipeline for the preparation of the input data of the language model: 0) pre-calculate FastText embeddings for each relation in KG and form a static similarity matrix out of them; 1) form a neighborhood for the input triplet; 2) sort the neighborhood by relation similarity to the relation in the input query; 3) verbalize and concatenate query triplet and its neighborhood.

B Neighborhood formation

To augment the semantic information contained in the entity and relationship names with the structural one, we used triplet neighborhoods. A triplet neighborhood is formed as follows: for a query ($head$, $relation$, ?) search for 1-hop neighbors associated with $head$ entity, while excluding from the formed neighborhood target $tail$ entity, which is associated with $head$ by $relation$; $head$ can be both head or tail entity in triplets that form this 1-hop neighborhood. In this manner, we obtain the context as a neighborhood of the graph, avoiding the leakage of the target entity. The process of neighborhood formation is illustrated on Figure 4.

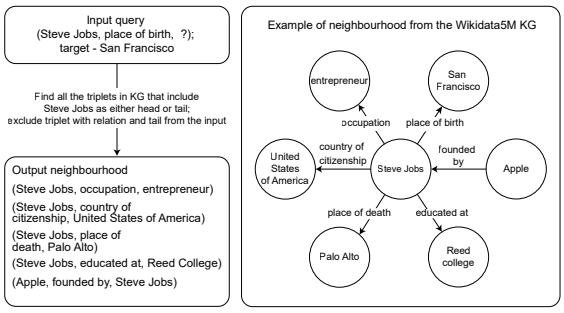

Figure 4: **Neighborhood formation** - taking the input triplet, search for the triplets in which the head from the input triplet is either head or tail, then exclude the triplet with relation and tail from the input. As a result, there is a list of triplets that form the neighborhood for a given triplet. The represented neighborhood from the Wikidata5m KG is incomplete and textually verbalized for ease of understanding.

However, some of the triplets from KGs result in extremely large neighborhoods, e.g. hub nodes such as "human". Therefore, we needed to tackle the problem of large neighborhoods to fit it into the model's context, which is limited to 512 tokens for most language models including T5. We propose to do it as follows: 1) sort the neighborhood by semantic similarity of relations from its triplets to the relation from the input triplet in order to prioritize more important information in the context; 2) limit the sorted neighborhood to 512 triplets, since this will always be at least as big as the size of the allowed context, and, after verbalization, specify the maximum length of 512 for the model tokenizer to fit the resulting verbalized neighborhood representation into the language model context. Neighborhood sorting by semantic proximity was performed using a pre-calculated matrix of cosine similarity of relations in KG, for similarity calculation the relations were embedded by the fasttext model (Bojanowski et al., 2017). Figure 5 displays a pipeline for neighborhood sorting.

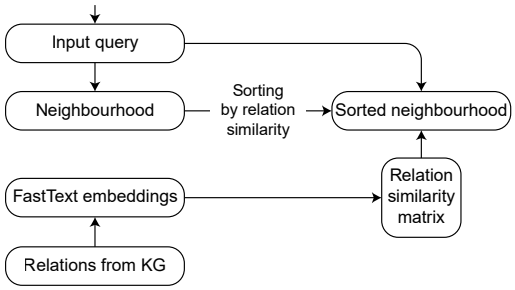

Figure 5: **Neighborhood sorting** - taking the input triplet, its neighborhood, and pre-calculated similarity matrix for verbalized relation embedded by the FastText model, sort the neighborhood by the similarity of relations in its triplets to the relation in the input query.

Neighborhood composition was a relatively straightforward task for a smaller graph like ILPC, but for Wikidata KG with a large number of triplets in the graph, neighborhood formation took a significant amount of time and RAM. To solve this problem, we used the MongoDB database[3], which stored knowledge graph triplets in its collection in JSON format and contained indices for 'head' and 'tail' fields to speed up retrieving. In order not to store verbalized data in memory either during the verbalization process or during model training, verbalized representations were outputted to the MongoDB collection too, and retrieved by batches from the database during training.

[3]https://www.mongodb.com

C Triplets verbalization

To pass the resulting representations of queries to the input of the language model, we had to perform a transformation of these queries and their neighborhoods into textual form. In order to do this, we first needed to generate a mapping of entity and relationship Wikidata IDs into corresponding texts. For all the datasets we performed our experiments on there is such textual mapping. However, often there is no unambiguous encoding in such mappings because many entities have the same name. The problem of non-unique textual representations concerns Wikidata KG entities, not relations, e.g. apple as a fruit of the apple tree and Apple as an American multinational technology company. To tackle this issue we used the same method KGT5 authors proposed: 1) concatenate non-unique entity names with corresponding descriptions if ones exist; 2) in case there is no description or after description, concatenation text representations cannot be disambiguated, add unique numeric ids. For all other smaller datasets we only added a unique numeric identifier at the end of the name of non-unique entities. The full disambiguation pipeline is displayed in Figure 6. No such transformations were required for KG relations, as they were fully distinguishable by their text representation.

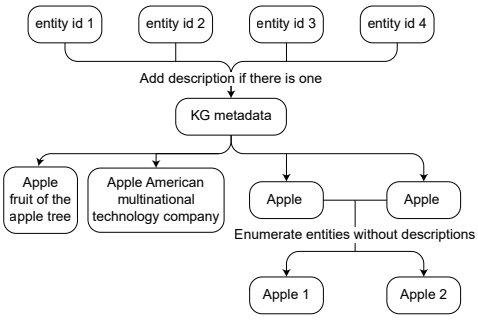

Figure 6: Disambiguation schema for wikidata5m entities: 1) concatenate entities' textual representation with corresponding description if there is one in the KG metadata; 2) if there is no description for some entities - concatenate their names with unique numerical id.

The next step after obtaining 1to1 mapping of entities and relation IDs to their text representations

was verbalization of the resulting queries and corresponding neighborhoods in the KG. Textual representations of the query $(head_q, relation_q, ?)$ from KG were formed as follows:

1. first part of query verbalization was formed via concatenation of $head_q$'s and $relation_q$'s text mappings with "predict" string, e.g. "predict Steve Jobs place of birth";

2. for each i-th triplet $(head_q, relation_i, tail_i)$ or $(head_i, relation_i, head_q)$ from the sorted query's neighborhood we formed a string consisting of $relation_i$ and entity not belonging to the input query, i.e. for query $(Steve\ \ Jobs, place\ \ of\ \ birth, ?)$ the verbalized triplet $(Steve\ \ Jobs, occupation, entrepreneur)$ from its neighborhood will be "occupation entrepreneur"; in case of so-called inverse triplet, e.g. in triplet $(head_i, relation_i, head_q)$, where $head_q$ is a tail of the neighborhood triplet, we add "inverse of " to the textual mapping of $relation_i$, i.e. for $(Apple, founded\ \ by, Steve\ \ Jobs)$ verbalization will be "inverse of founded by Apple";

3. at the end we concatenate the verbalization obtained in stage (1) and all the verbalizations from stage (2) using the" [SEP] " string inserted between all the above-mentioned strings.

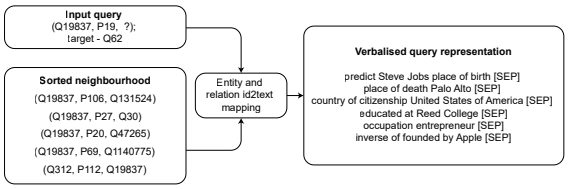

Figure 7: **Query verbalization** - map each entity and relation id from query triplet and its neighborhood triplets with corresponding textual representations; concatenate "predict" string, obtained query verbalization excluding target and the sorted triplets' verbalization without head entity from the query, separating each verbalized triplet by "[SEP]" token.

Example of query and its neighborhood verbalization is represented on a Figure 7.

D Link prediction inference

As in KGT5 (Saxena et al., 2022), to evaluate top-k possible answers, we sample k most probable sequences from the decoder and assign them their corresponding probabilities, which is a sum of log probabilities of each token in the sequence. Then, we assign $-\infty$ score for all the nodes that were not generated by the sampling procedure. In this way, we are producing results comparable with traditional KG models as we obtain plausibility scores for all possible triplets in KG. By such procedure, we can obtain whether a real score of the model (if the target sequence persists in k generated sequences) or the score that will be worse than a real (if the target sequence was not generated as one in the top-k probable sequences), thus not overestimating the resulting metric even though we do not score all the possible triplets. Additionally, to exclude both entities that do not belong to KG and known links from the train and valid graph in a transductive setting, we performed a filtering procedure specific to KGC evaluation protocol as described in the previous studies (Bordes et al., 2013).

As a main metric for choosing the best model during evaluation, we used Exact Match (EM) or in other words Hits@1, which returns 1 in case the generated and target sequences are exactly the same or 0 otherwise. However, an exact match between the target and the generated sequence can be a rather strict metric for long target entity names, and for previously unseen entities. Thus, we came up with a new approach for scoring entities produced by generative models such as T5 for inductive KGC. In this approach, we first embedded verbalized entities from the inference part of KG by Sbert model (Reimers and Gurevych, 2019) and then indexed the resulted vectors using FAISS index (Johnson et al., 2019). In such a way, we used a simple flat FAISS index to perform a lookup for the k most similar entities from inference KG to the generated one. Using the index for the inductive ILPC dataset, we retrieve entities from the KG that are semantically closest to the generated sequence even if the generated string is imperfect, but contains the right tokens referring to the target entity. In this manner, we could calculate Hits@k retrieving top-k closest to the generated entity names in a similar way as distance-based methods. Overall, in the case of inductive setup, we used a combination of exact match (EM) and FAISS-based met-

rics Hits@k calculated by the following formula: $1000EM + Hits@1$. In this way, we give more weight to the exact match metric, but in the case of equal EM in different checkpoints, the model with the best Hits@1 will be chosen.

E Datasets statistics

Table 4: Statistics of used KGs.

Dataset	# Entities	# Relations	# Triplets
Wikidata5M	4.8M	828	21M
ILPC-small	10,230	96	78,616
ILPC-large	46,626	130	202,446

F Attention map interpretation

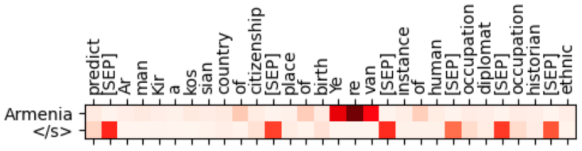

Figure 8: **Example of cross-encoder attention map** highlighting the importance of neighborhood triplets from the input

Figure 8 demonstrates a cross-encoder attention map highlighting the input neighborhood's significance for the target prediction. The query *(Arman Kirakossian, country of citizenship, ?)* identifies Yerevan, Armenia's capital, as the birthplace of the target entity, thereby amplifying its role in predicting the right country of citizenship.

G Zero-shot link prediction with `gpt-3.5-turbo`

Since OpenAI API only returns generated text, we cannot get its probabilities. Therefore, we use a slightly different evaluation pipeline from other models. We generate up to three predictions with `gpt-3.5-turbo-0613`. If any of the randomly selected k predictions match the target entity after cleaning and normalization, we set that model prediction as correct (Hits@k in Table 2).

We use the following prompts to control the `gpt-3.5-turbo-0613`.

G.1 Zero-shot link prediction without neighborhood

System prompt: *You will be provided with a incomplete triplet from the Wikidata knowledge graph. Your task is to complete the triplet with a tail (subject) based on the given triplet head (object) and relation. Your answer must only include the tail of the triplet with prefix 'Tail:'. Do not include relation into the answer.*

User prompt: *Triplet to complete: <verbalized triplet>.*

G.2 Zero-shot link prediction with neighborhood

System prompt: *You will be provided with a incomplete triplet from the Wikidata knowledge graph. Your task is to complete the triplet with a tail (subject) based on the given triplet head (object) and relation. Triplet to complete IS NOT in the list of related nodes, but it may contain helpful clues. Your answer must only include the tail of the triplet with prefix 'Tail:'. Do not include relation into the answer.*

User prompt: *Adjacent relations: <verbalized neighborhood>\nTriplet to complete: <verbalized triplet>.*

H Overview of concurrent work

While our work was under review, the concurrent work KGT5-context (Kochsiek et al., 2023) was published. Following the same idea, KGT5-context uses verbalized node neighborhood as additional context for KGC with the KGT5 model. However, KGT5-context explores the applicability of neighborhood information for KGC only in transductive setup. Also, neighborhood sampling in KGT5-context was performed randomly instead of prioritizing the neighbors, which makes the results less reproducible and does not consider the importance of neighbors in the context of a query. Furthermore, we show that this approach generalizes well to other pre-trained language models like `gpt-3.5-turbo` in a zero-shot setting without pre-training on KGC datasets.

We have made the effort and tried our best to reproduce the results of the KGT5 model (Saxena et al., 2022) on Wikidata5M. We report the results of *T5-small not pre-trained* (Table 1), which is our version of the KGT5 model. Therefore, all our other models were trained and evaluated in

the same setting and with the same pipeline as for *T5-small not pre-trained* model.