Knowledge Graph as Tokens: Knowledge Base Construction Using Language Model with Graph Neural Network and Soft Prompting

Anonymous ACL submission

Abstract

A knowledge graph represents real-world concepts as interconnected nodes, with widely recognized examples like WikiData, DBPedia, and YAGO. However, these graphs remain incomplete, and knowledge evolves over time. Constructing knowledge graphs involves extracting information from various sources, including text, images, and videos. Language models store knowledge in their parameters, and the ISWC has introduced a competition, LM-KBC, to extract this knowledge for enhancing knowledge graphs. Previous research has focused on hard prompting and few-shot methods, leaving an unexplored opportunity for soft prompts. This study proposes Knowledge Graph as Tokens (KGAT), inspired by Frozen and Seq2Path, using a graph neural network (GNN) to incorporate graph context as a soft prompt in language models. Evaluations on ISWC datasets (2022–2024) with Llama 3.1 8B show that KGAT outperforms the baseline, albeit with a minor improvement.

1 Introduction

A knowledge graph is a representation of knowledge that uses a graph data structure to store information about the real world. In a knowledge graph, the graph consists of a collection of entities represented by nodes, which are interconnected through specific relationships (Hogan et al., 2022). Although knowledge graphs have been developed for a long time, current knowledge graphs are still considered to lack complete information (Demir et al., 2023; Singhania et al., 2022; Kalo et al., 2023; Ré et al., 2014). To construct and enhance existing knowledge graphs, a variety of information and data is required, which can be obtained from text documents, images, audio, video, and other diverse sources of information (Zhong et al., 2024). In the field of natural language processing, it has been discovered that language models can store knowledge within their parameters, acquired through training. This indicates that language models could serve as a potential new source of data for constructing knowledge bases (AlKhamissi et al., 2022; Petroni et al., 2019; Roberts et al., 2020). This motivated the International Semantic Web Conference (ISWC) to organize the Knowledge Base Construction from Pre-trained Language Models (LM-KBC) competition (Singhania et al., 2022; Kalo et al., 2023).

In the LM-KBC task, the language model receives input in the form of a subject entity *s* and a relation (or predicate) *r*. The model is then expected to output a set of relevant object entities *[o1, o2, ..., ok]*. There are three possible outcomes: no matching object entities, exactly one matching object entity, or multiple matching object entities. The LM-KBC competition features two tracks: the small-model track and the open track. The small-model track limits participants to using language models with a maximum of 1 billion parameters (including the BERT track in 2022), while the open track allows contestants to use any type of language model and incorporate additional context to achieve the best results. In this paper, we focus on the open track.

In previous research, most approaches focused on variations of prompting using hard prompts, particularly few-shot prompting (Alivanistos et al., 2022; Biester et al., 2023; Li et al., 2023; Nayak and Timmapathini, 2023; Zhang et al., 2023). In addition to using few-shot prompting, there are also approaches that utilize zero-shot prompting to test the zero-shot capabilities of language models in the context of LM-KBC (Ghosh, 2023). In applying few-shot prompting, the "shots" used are generally derived from the provided training and validation

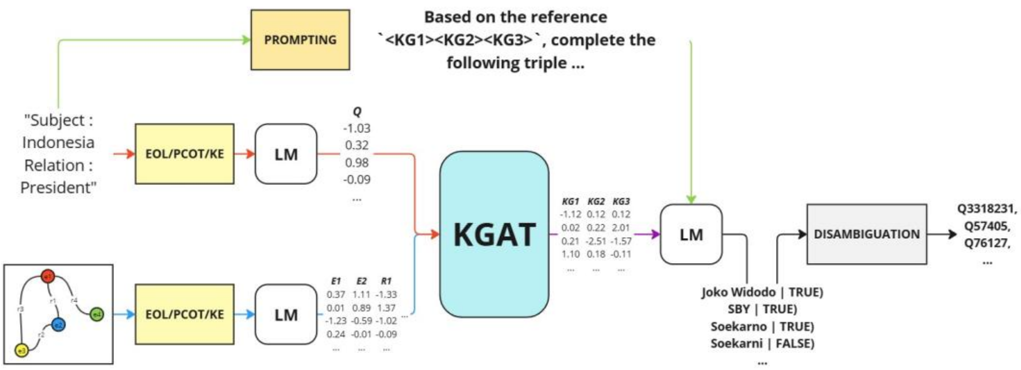

Figure 1: Overview of the KGAT process. The method takes two inputs: a prompt containing a subject and relation, and a knowledge graph. Both inputs are encoded using EOL/PCOT/KE and a language model to obtain vector representations. The KGAT module then generates knowledge graph (KG) tokens, with the number determined by the user, representing relevant parts of the knowledge graph. These KG tokens are prepended to the prompt as a form of soft prompting and fed back into the language model. Finally, the model generates object entity candidates via beam search, followed by post-processing to retrieve the corresponding entity IDs.

data. Given that the training and validation data consist of triples, this means that the provided data essentially represents the knowledge graph itself. Unfortunately, previous research has treated these data as text formatted for few-shot prompting, thereby neglecting the inherent graph structure of the training and validation data. To incorporate the graph data characteristics present in the training data, we propose a new approach that processes reference triples using a graph neural network and utilizes them as soft prompts for the language model. This approach, called Knowledge Graph as Tokens (KGAT), involves representing the context of the knowledge graph as a virtual token (soft prompt) that can be processed by the language model.

2 Methodology

We are inspired by *Frozen* (Tsimpoukelli et al., 2021), which transforms image representations into virtual tokens. In contrast, our approach involves converting the knowledge graph into virtual tokens. There are several issues that we believe need to be analyzed, particularly concerning irrelevant context (or shots) and the output length from the language model. Some approaches use rules to select the shots to be embedded in the context. However, the presence of shots does not always positively impact the language model's output. This is because shot selection is often performed automatically and can be stochastic (random), leading to potentially irrelevant shots that degrade the model's performance (Cattan et al., 2024).

Our proposed solution processes the knowledge graph using a GNN encoder. This encoder employs TransformerConv (or UniMP) (Shi et al., 2021) to handle the knowledge graph, utilizing its' attention mechanism. We designed the GNN encoder block based on the architecture of the encoder block in the vanilla Transformer (Vaswani et al., 2017). This approach assumes that if the design of the encoder block mimics that of the Transformer, the output from the GNN encoder block will be able to preserve the semantic information related to the knowledge graph.

Since GNNs treat nodes and edges as vectors, a technique is needed to convert entity names and relations into a single vector. It's important to note that the tokenization mechanism in transformers allows a single word to be split into multiple tokens. To convert entity names and relations into a single vector (or token), techniques such as EOL, PCOT, and KE (Zhang et al., 2024) are used to transform the sequence of tokens into a single vector. The resulting vectors then will be processed by KGAT module that outputs virtual tokens. We are also inspired by the *Visual Prefix* in *Frozen* for creating virtual tokens, which we name *Graph Prefix*.

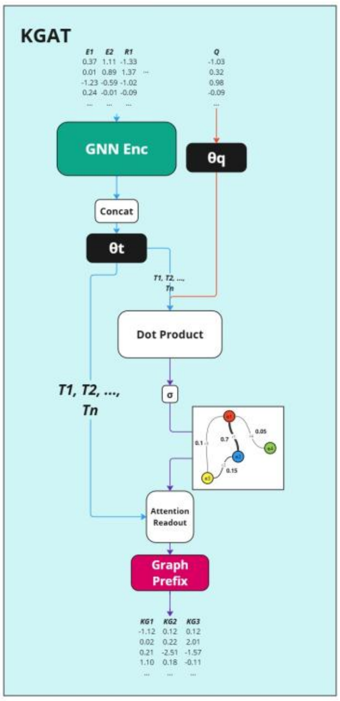

Figure 2: The KGAT flow begins by encoding inputs with EOL/PCOT/KE, followed by processing through a GNN encoder to extract node features. Transformed subject, object, and relation vectors are concatenated and passed through a feedforward network θ_t. Retrieval is performed via the dot product between triple vectors and a query vector, which is obtained by encoding the input sequence through another feedforward network θ_q. The resulting relevance scores determine which triples are selected, and the readout module aggregates the retrieved triple vectors.

3 Experimental Setup

3.1 Training and Inference

In our proposed approach, there are two phases of training: one for the retrieval task and one for the LM-KBC task. The retrieval training phase involves modules such as the GNN Encoder and several feedforward networks (θ_t and θ_q) within the KGAT model. The retrieval problem is formulated as a task where the model is given a query and several triples (knowledge graph). The model must score the relevance of triples that are related to the query. To ensure that triples provide information related to the answer to the given query and not the query itself, an answer vector is provided during retrieval training. Feedforward network θq will map the query to closely match the semantics of the answer vector, so the retrieval results involve the triple vectors, and the query vector will indirectly represent their proximity to the answer vector for the given query. For the first phase of training, the objective function will follow the criteria outlined below:

$$L = BCE\left(\sigma\left(\theta_t(T_n)\cdot\theta_q(Q)\right)\right) + BCE(\sigma(\theta_t(T_n)\cdot\bar{V})) + BCE\left(\sigma\left(\theta_q(Q)\cdot\bar{V}\right)\right) \quad (1)$$

$$\text{with, } \bar{V} = \frac{1}{n}\sum_i^n V_i$$

In the LM-KBC training phase, we address challenges related to output length, entity ordering, and hallucination issues, particularly in autoregressive. Since these models generate outputs sequentially, longer outputs affect context length and may disrupt performance. To mitigate this, we reformulate LM-KBC as a single-tuple generation task, inspired by Seq2Path (Mao et al., 2022) from aspect-based sentiment analysis. The tuple consists of the subject entity, relation, object entity, and a discriminative token. During inference, the model receives a prompt with the subject entity and relation but must predict the object entity and discriminative token ("true" or "false") using beam search, outputting "NONE" if no relevant entity exists. Our training strategy follows Seq2Path's approach, incorporating augmentation techniques, loss masking, loss computation, and pruning to enhance model accuracy and robustness.

3.2 Dataset

In the retrieval training phase, we train the model using the GraphExtQA (Shen et al., 2023) dataset, while the LM-KBC training phase utilizes data provided by ISWC, specifically LM-KBC 2022, 2023, and 2024.

3.3 Metrics

We use the same metrics defined by ISWC for the LM-KBC competition. The three main metrics are precision, recall, and F1-score. Each metric is defined as follows:

$$Precision = \frac{P \cap GT}{|P|} \qquad Recall = \frac{P \cap GT}{|GT|} \qquad f1 = \frac{2 \times precision \times recall}{precision + recall} \quad (2)$$

4 Results and Analysis

4.1 Hyperparameter Tuning

We performed hyperparameter tuning to obtain the most optimal solution candidates. We acquired the value 2 for the number of GNN Encoder blocks and 8 for the number of attention heads. For the dimensions of the feed-forward network in the GNN Encoder, we obtain D/2, where D represents the embedding size of the language model. The batch size was 8, with each batch containing 50 reference triples. The choice of 50 was due to computational constraints. For the learning rate, we get 1e-5 for the retrieval training process and 1e-6 for the LM-KBC training, using the Adam optimizer.

4.2 Evaluation Result

We conducted several evaluation scenarios, including cross-evaluation where the training and testing data come from different datasets (e.g., training data from LM-KBC 2022 and testing data from LM-KBC 2023).

Train Data	Test Data								
	LM-KBC 2022			LM-KBC 2023			LM-KBC 2024		
	P	R	F1	P	R	F1	P	R	F1
LM-KBC 2022	0.55	0.70	**0.52**	0.32	0.52	0.31	0.27	0.60	0.24
LM-KBC 2023	0.49	**0.71**	0.47	0.41	**0.63**	**0.43**	0.26	**0.66**	0.27
LM-KBC 2024	**0.60**	0.55	0.44	**0.51**	0.42	0.32	0.35	0.56	**0.28**
ALL	0.48	0.68	0.46	0.21	0.29	0.16	**0.57**	0.40	0.26

Table 1: Evaluation result.

To measure the success of our proposed approach, we compared it with a baseline method. The baseline method employs few-shot prompting with 5 shots. The results indicate that, overall, KGAT performs better than the baseline method.

Model	Test Data								
	LM-KBC 2022			LM-KBC 2023			LM-KBC 2024		
	P	R	F1	P	R	F1	P	R	F1
Baseline	0.60	0.60	0.47	0.51	0.46	0.38	0.50	0.50	0.33
KGAT	0.60	**0.71**	0.52	0.51	0.63	0.43	**0.57**	**0.66**	0.28

Table 2: Comparison with baseline.

4.3 Error Analysis

To ensure an objective and fair comparison, we conducted statistical testing between the results of KGAT and the baseline method. We utilized a one-tailed paired t-test for this purpose. The results indicate that, overall, KGAT performs better than the baseline method. However, using t-test with a confidence level of 5%, KGAT has not yet demonstrated a statistically significant advantage over the baseline method. Several potential reasons for this result include:

- **Beam Value** The beam value forces the language model to predict at least as many objects as the beam size. This can be problematic because the number of objects for each subject-entity pair and relation varies (ranging from zero to infinity). This implies that for subject-entity and relation pairs with fewer objects than the beam size, the model may predict incorrect objects (false positives), leading to a lower precision score. The beam size proposed by Seq2Path, set at 6, proves to be ineffective in the context of the data used in this study. An investigation into the average number of objects in the training and validation data reveals that the average number of objects per subject-entity and relation pair falls within the range of 2-4 objects.

- **Empty Object Case** In cases where no objects are expected, the model is anticipated to have a high precision by avoiding incorrect predictions (false positives). However, due to the use of beam search, the model often attempts to provide predictions even when they are incorrect. This issue arises partly because of the relatively low ratio of empty cases, with the proportion of such cases being less than 25%.

- **Effect of Data Augmentation** The data augmentation process introduced an additional problem by reducing the ratio of empty cases. As a result, the model became more inclined to predict object entities and less likely to output "NONE" as the prediction in empty cases.

5 Conclusion

Based on the results obtained, it was found that KGAT is better than the baseline method. However, the improvement is not considered as a significant improvement. This is attributed to the effects of beam search, data augmentation, and imperfect retrieval mechanism. Although KGAT has not yet demonstrated significantly superior performance, there are areas for improvement. These include refining the beam search mechanism, enhancing data augmentation, and replacing the objective function in subgraph generation training to achieve a better retrieval mechanism.

Limitations

This study is limited to using a moderate-sized LLM, LLaMA 3.1 8B. To ensure fair evaluation, we use the same base model for the baseline method as well.

Acknowledgments

We would like to express our sincere gratitude to the AI Center at the Bandung Institute of Technology (ITB) for providing us with the computational resources necessary for this research.

Ethics Statement

This research utilizes publicly available datasets and complies with their respective licenses. As our work focuses on language modeling and knowledge graph construction, we acknowledge that pretrained language models may inherit biases from their training data; however, addressing such biases falls under the responsibility of dataset providers. Our study does not introduce or amplify these biases beyond what is inherent in the models and datasets used.

References

Dimitros Alivanistos, Selene Baez Santamaria, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as Probing: Using Language Models for Knowledge Base Construction. In *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models 2022*, pages 11–34, Hangzhou.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A Review on Language Models as Knowledge Bases.

Fabian Biester, Daniel Del Gaudio, and Mohamed Abdelaal. 2023. Enhancing Knowledge Base Construction from Pre-trained Language Models using Prompt Ensembles. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*, Athens.

Arie Cattan, Alon Jacovi, Alex Fabrikant, Jonathan Herzig, Roee Aharoni, Hannah Rashkin, Dror Marcus, Avinatan Hassidim, Yossi Matias, Idan Szpektor, and Avi Caciularu. 2024. Can Few-shot Work in Long-Context? Recycling the Context to Generate Demonstrations.

Caglar Demir, Michel Wiebesiek, Renzhong Lu, Axel-Cyrille Ngonga Ngomo, and Stefan Heindorf. 2023. LitCQD: Multi-hop Reasoning in Incomplete Knowledge Graphs with Numeric Literals. In pages 617–633.

Shrestha Ghosh. 2023. Limits of Zero-shot Probing on Object Prediction. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*, Athens.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2022. Knowledge Graphs. *ACM Computing Surveys*, 54(4):1–37.

Jan-Christoph Kalo, Sneha Singhania, Simon Razniewski, and Jeff Z. Pan. 2023. LM-KBC 2023: 2nd Challenge on Knowledge Base Construction from Pre-trained Language Models. *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*, 3577.

Xue Li, Anthony Hughes, Majlinda Llugiqi, Fina Polat, Paul Groth, and Fajar J. Ekaputra. 2023. Knowledge-centric Prompt Composition for Knowledge Base Construction from Pre-trained Language Models. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*, Athens.

Yue Mao, Yi Shen, Jingchao Yang, Xiaoying Zhu, and Longjun Cai. 2022. Seq2Path: Generating Sentiment Tuples as Paths of a Tree. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2215–2225, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anmol Nayak and Hari Prasad Timmapathini. 2023. LLM2KB: Constructing Knowledge Bases using instruction tuned context aware Large Language Models. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? 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 2463–2473, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher Ré, Amir Abbas Sadeghian, Zifei Shan, Jaeho Shin, Feiran Wang, Sen Wu, and Ce Zhang. 2014. Feature Engineering for Knowledge Base Construction. *IEEE Data Eng. Bull.*, 36(3):26–40.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuanchun Shen, Ruotong Liao, Zhen Han, Yunpu Ma, and Volker Tresp. 2023. GraphextQA: A Benchmark for Evaluating Graph-Enhanced Large Language Models.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554, California. International Joint Conferences on Artificial Intelligence Organization.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge Base Construction from Pre-trained Language Models. *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models 2022*, 3274:1–10.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal Few-Shot Learning with Frozen Language Models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Bohui Zhang, Ioannis Reklos, Nitisha Jain, Albert Meroño Peñuela, and Elena Simperl. 2023. Using Large Language Models for Knowledge Engineering (LLMKE): A Case Study on Wikidata. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*, Athens.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple Techniques for Enhancing Sentence Embeddings in Generative Language Models.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2024. A Comprehensive Survey on Automatic Knowledge Graph Construction. *ACM Computing Surveys*, 56(4):1–62.