Graph Augmentation with LLMs for Knowledge-Aware Recommender Systems

Giuseppe Spillo[1,*,†], Cataldo Musto[1,†], Matteo Mannavola[1], Marco de Gemmis[1], Pasquale Lops[1] and Giovanni Semeraro[1]

[1]*University of Bari Aldo Moro, Dept. of Computer Science, Via Edoardo Orabona, 4, 70125, Bari, Italy*

Abstract

In this paper, we propose a recommendation model exploiting a graph augmentation technique based on Large Language Models (LLMs) to enrich information in its underlying Knowledge Graph (KG). We assume KG triples can be noisy or incomplete, leading to sub-optimal modeling of item characteristics and user preferences. Graph augmentation can thus improve data quality and provide high-quality recommendations.

Accordingly, we propose our framework, that starts with a KG and designs *prompts* for querying an LLM to augment the graph by incorporating: *(a)* further item features; *(b)* further nodes describing user preferences, obtained by reasoning over liked items. The augmented KG is then passed through a Knowledge Graph Encoder, which learns user and item embeddings. These embeddings are used to train a recommendation model, providing personalized suggestions. Experiments show LLM-based graph augmentation significantly improves our recommendation model's predictive accuracy, confirming its effectiveness and the validity of our intuitions.

Keywords

Recommender Systems, Large Language Models, Knowledge Graph Augmentation

1. Introduction

Nowadays, Recommender Systems (RSs) effectively handle *information overload* and support user decision-making [15]. Knowledge-Aware RSs (KARSs) exploit side information, often Knowledge Graphs (KGs) [6], to learn effective item representations and provide precise recommendations [14, 13, 11].

Despite their effectiveness [32, 1, 22, 24, 21, 25], KGs have flaws: *first,* they may overlook descriptive item features [5]. *Second,* KGs in KARS typically overlook user preference into item groups (e.g., fantasy movies) or specific characteristics (e.g., a director).

This paper[1] [18] proposes a methodology using Large Language Models [10] (LLMs) to augment the original KG. Our graph augmentation strategy aims to incorporate: *(a)* missing item features; *(b)* user preferences into item features. Our contributions are:

1. We design a novel framework for LLM-based graph augmentation in KARS.
2. We design prompts to extract new item features and user preference KG triples from LLMs.
3. We conduct extensive experiments, including ablation studies, and release the source code.

2. Related Works

Lately, KGs have enhanced KARS performance. **CKE** [32] and **CFKG** [1] enriched Collaborative Filtering (CF) data with item features learned from KGs using TransE [3]. Later methods leveraged graph neural networks. **KGCN** [22] used Graph Convolutional Networks (GCNs) [34] to aggregate KG

IIR2025: 15th Italian Information Retrieval Workshop, 3th - 5th September 2025, Cagliari, Italy

[*]Corresponding author.

[†]These authors contributed equally.

giuseppe.spillo@uniba.it (G. Spillo); cataldo.musto@uniba.it (C. Musto); m.mannavola15@alumni.uniba.it (M. Mannavola); marco.degemmis@uniba.it (M. d. Gemmis); pasquale.lops@uniba.it (P. Lops); giovanni.semeraro@uniba.it (G. Semeraro)

[1]Full version of this paper has been accepted at ACM UMAP 25 with the title *GAL-KARS: Exploiting LLMs for Graph Augmentation in Knowledge-Aware Recommender Systems* [17]

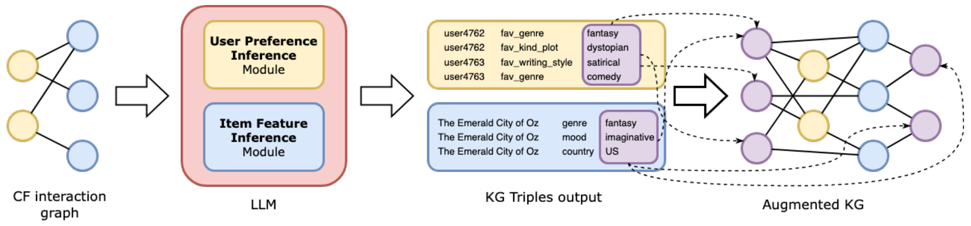
CF interaction graph
User Preference Inference Module
Item Feature Inference Module
LLM
user4762 fav_genre fantasy
user4762 fav_kind_plot dystopian
user4763 fav_writing_style satirical
user4763 fav_genre comedy
The Emerald City of Oz genre fantasy
The Emerald City of Oz mood imaginative
The Emerald City of Oz country US
KG Triples output
Augmented KG

Figure 1: Starting from the CF graph, we infer new triples describing users and items, and we incorporate them to obtain our augmented KG.

information, capturing higher-order relationships. **KGAT** [24] employed Graph Attention Networks to model high-order connections. None of them focus on encoding more detailed user preferences inferred from liked items. **KTUP** [4] used TransH [26] for KG completion, transferring embeddings to user modeling. **KGNNLS** [21] learned user-specific item embeddings by transforming the KG into a user-specific graph.

Most graph augmentation for recommendation uses contrastive learning by perturbing the user-item graph [9, 31, 29, 33]. In contrast, our method leverages pre-trained LLMs for KG augmentation. While LLMs have been explored for graphs [30, 35], their application in recommendation remains limited. **KAR** [28] and **LLMRec** [27] are the only related works. KAR infers textual knowledge for sequential recommendation, later encoded via BERT [7], making direct comparison difficult. LLMRec uses LLM-based feature augmentation via OpenAI APIs. Unlike these, we use an open-weight LLM (LLama) to generate new triples, guided by tailored prompts that enrich the KG with unseen relations (e.g., a movie's *mood*, a book's *style*), enhancing recommendation quality.

3. Problem Formulation

We define the data used and formalize the recommendation task.

CF Data. Given users $\mathcal{U}$ and items $\mathcal{I}$, the user-item interactions are represented by a binary matrix $\mathcal{Y} \in \mathbb{R}^{n \times m}$, where $y_{u,i} = 1$ if user u liked item i, and 0 otherwise. This can be modeled as an interaction graph $\mathcal{G}$ with nodes for users/items and edges labeled *like*/*dislike*.

Knowledge Graph. Each item $i \in \mathcal{I}$ is described by triples from a knowledge graph $\mathcal{KG}$, such as *(item, relation, entity)* (e.g., *(Tender is the Night, author, Fitzgerald)*).

Graph Augmentation. We use LLMs to generate additional triples describing item features and user preferences. For each item i, a prompt p_i is used to produce triples $\mathcal{T}_i = LLM(p_i)$ (e.g., *(Nixon, theme, political corruption)*), forming a new KG $\mathcal{KG}_{\mathcal{I}}$. Similarly, for each user u, we prompt the LLM with their liked items to generate a KG $\mathcal{KG}_{\mathcal{U}}$ describing preferences (e.g., *(user83, fav_setting, post-apocalyptic)*). We then build an augmented KG by merging $\mathcal{G}$, $\mathcal{KG}$, and either or both of $\mathcal{KG}_{\mathcal{I}}$ and $\mathcal{KG}_{\mathcal{U}}$.

Representation Learning. We encode the augmented KG using a KG encoder to learn embeddings $\vec{e_u}$ and $\vec{e_i}$ for users and items, which are fed into a neural network for recommendation.

Problem Definition. Given $\mathcal{KG}_{\text{aug}}$ and model parameters θ, we learn a function $\widetilde{Y}(u, i \mid \mathcal{KG}_{\text{aug}}, \theta)$ to predict user u's interest in item i. We evaluate using a top-k recommendation setting based on predicted scores.

4. Methodology

The augmentation workflow is shown in Figure 1. Starting from the user-item graph $\mathcal{G}$ (Section 3), we use LLMs for *Item Feature Inference* and *User Preference Inference*, generating triples of the form *(user/item, relation, entity)*. Merging these outputs forms the augmented KG used in our KARS. A *Knowledge Graph Encoder* then learns user/item embeddings, which are fed into the *RecSys* module to generate recommendations. We detail each module below.

Dataset	# users	# items	# interactions	sparsity
DBBOOK	5660	6617	129316	99.65%
ML1M	6036	3192	946120	95.09%

Table 1
Statistics of the Datasets

Item Feature Inference Module. To infer item features as KG triples, we use a zero-shot prompting approach with an LLM. Our prompt takes an item's name and returns descriptive triples. The prompt has three parts: a *System Prompt* (instructing the LLM about the task), a *User Prompt* (providing task-specific instance details), and the *Model Output* (the LLM's response). In the System Prompt, we ask the LLM to generate relevant item features for a given domain and specify the output format. In particular, we request both common KG features (e.g., author, topic, genre) and novel ones (e.g., *writing style*, *mood*) typically absent from KGs. This prompt allows the LLM to complete missing knowledge in the original KG, addressing data sparsity on one side, and incorporating new, pre-trained features from the LLM, enriching the data model, on the other side. This process is repeated for all items $i \in \mathcal{I}$, and the combined triples form $\mathcal{KG}_{\mathcal{I}}$, representing all LLM-generated item features.
User Preference Inference Module. Next, a similar process is designed to infer user preferences in the form of KG triples. To this end, we *zero-shot* prompt an LLM in such a way that, given the list of items the user likes, the LLM returns a set of triples encoding her preferences. The structure of the prompt is similar, but rather than incorporating further knowledge about the items, the goal of this part of the augmentation process is to introduce new *edges* in the original KG. In our vision, these edges may improve the quality of the underlying data model, thus improving the performance in a downstream recommendation task as well. Regarding the choice of the elements included in the prompt, we point out again that we mixed features encoded in the original KG (*i.e.,* preferred genre or authors) and more fine-grained characteristics, such as the *mood* of movies liked by the user. Also in this case, the prompt is generated for all the users $u \in U$ based on the items they like, and the triples returned by the LLM are merged in the graph $\mathcal{KG}_{\mathcal{U}}$, which encodes all the user features obtained through the graph augmentation process.
Knowledge Graph Encoder and RecSys Modules. After prompting the LLMs, we build an augmented graph that merges $\mathcal{G}$ and $\mathcal{KG}$ with either one between $\mathcal{KG}_{\mathcal{I}}$ and $\mathcal{KG}_{\mathcal{U}}$, or both of them Based on this data model, a Knowledge Graph Encoder comes into play to learn embeddings representing users and items in the augmented graph. In this work, we used CompGCN [20] as a Graph Encoder. This choice is justified by the competitive performance shown by other models exploiting GCNs for recommendation tasks [8, 16, 23, 17]. More details about this encoder can be found in the original papers [20, 18]. After the encoding, the resulting user and item embeddings are used to train a deep recommendation architecture; in particular, this is trained on a subset of ratings from Y using binary cross-entropy as the loss function. During inference, scores are computed for all test set items, ranked, and the top-k items form the recommendation list

5. Experimental Evaluation

Our experiments aimed to answer this Research Questions: How does each KG configuration contribute to the overall performance of the model?

5.1. Experimental Design

Datasets and Knowledge Graphs. We considered DBBOOK and MovieLens1M (ML1M) for our experiments. We used DBpedia [2] as base KG, by using publicly available mappings. We augmented such KGs according to the methodology previously introduced. More details are on our full paper [18].
Protocol. Graph augmentation relies on training data: positive ratings are used to infer user preferences ($\mathcal{KG}_{\mathcal{U}}$), and training items to infer item features ($\mathcal{KG}_{\mathcal{I}}$). User embeddings are learned on the training

Dataset	DBbook							ML1M						
KG	Precision	Recall	F1	NDCG	Gini	EPC	APLT	Precision	Recall	F1	NDCG	Gini	EPC	APLT
$\emptyset$	0.6858	0.5450	0.5590	0.8781	**0.6458**	0.6162	**0.2279***	0.8022	0.4574	0.4938	*0.9438*	0.7504	*0.6642*	*0.0803*
$\mathcal{KG}$	0.7006	0.5575	0.5720	0.8912	0.6641	0.6318	0.2091	0.7965	0.4550	0.4907	0.9400	0.7576	0.6624	0.0782
$\mathcal{KG}_{\mathcal{I}}$	0.6976	0.5547	0.5688	**0.8958**	0.6815	0.6313	0.1877	0.7995	0.4569	0.4926	0.9421	**0.7261**	**0.6761***	**0.0897***
$\mathcal{KG}_{\mathcal{U}}$	0.6996	0.5557	0.5704	0.8927	0.6792	0.6315	0.1892	*0.8029*	*0.4578*	*0.4942*	0.9435	0.7725	0.6539	0.0707
$\mathcal{KG}_{\mathcal{IU}}$	0.6992	0.5547	0.5699	0.8932	0.6797	0.6307	0.1902	0.7919	0.4534	0.4882	0.9364	0.7514	0.6558	0.0734
$\mathcal{KG}_{\mathcal{I}}^{+}$	*0.7028*	*0.5617*	*0.5745*	0.8905	0.6755	*0.6342*	0.1958	**0.8035***	**0.4585***	**0.4947***	**0.9440***	*0.7489*	0.6629	0.0701
$\mathcal{KG}_{\mathcal{U}}^{+}$	**0.7043***	**0.5619**	**0.5755**	*0.8950*	*0.6640*	**0.6364***	*0.2108*	0.7990	0.4557	0.4919	0.9421	0.7725	0.6512	0.0605
$\mathcal{KG}^{+}$	0.6931	0.5498	0.5646	0.8881	0.6844	0.6246	0.1861	0.8007	0.4570	0.4931	0.9423	0.7795	0.6511	0.0663

Table 2
Ablation studies. Best results are in **bold**, while second-best results are in *italic*; * means the difference with the $\mathcal{KG}$ results is statistically significant, with $p = 0.05$.

set, and top-k recommendations are generated by ranking predicted scores on the test set.
Implementation Details. We use `Llama3-ChatQA-1.5-8B` with 4-bit quantization via *BitsAndBytes* for efficiency. Graph encoding is done using Pykeen's CompGCN, and the recommendation model is implemented in PyTorch (available in our repository). We release more detail on our repository[2].
Experimental Settings. The Graph Encoder is trained for 15 epochs, with embedding size $k = 64$ and 3 GCN layers. Recommendation models are trained for 30 epochs (batch size 512, learning rate 0.01, $\alpha = 0.9$, Adam optimizer). Dropout is set to 0.2 for DBbook and 0.4 for ML1M.
Ablation Studies. To evaluate our method's effectiveness, we compare recommendation performance across eight graph variants: *(1)* CF-only ($\mathcal{G}$), *(2)* original KG ($\mathcal{G}$ + $\mathcal{KG}$), *(3)* item-only LLM graph ($\mathcal{KG}_{\mathcal{I}}$), *(4)* user-only LLM graph ($\mathcal{KG}_{\mathcal{U}}$), *(5)* full LLM-generated KG ($\mathcal{KG}_{\mathcal{I}}$ + $\mathcal{KG}_{\mathcal{U}}$), *(6)* item-augmented KG ($\mathcal{KG}$ + $\mathcal{KG}_{\mathcal{I}}$), *(7)* user-augmented KG ($\mathcal{KG}$ + $\mathcal{KG}_{\mathcal{U}}$), and *(8)* fully augmented KG ($\mathcal{G}$ + $\mathcal{KG}$ + $\mathcal{KG}_{\mathcal{I}}$ + $\mathcal{KG}_{\mathcal{U}}$). We analyze and compare results across these setups.
Evaluation Metrics. We evaluate with ClayRS [12] for reproducibility. Accuracy metrics include Precision, Recall, F1, and nDCG; diversity and novelty are assessed with Gini Index, EPC [19], and APLT. Paired t-tests assess statistical significance.

5.2. Discussion of the Results

Ablation Study. Table 2 reports ablation results across combinations of $\mathcal{KG}$, $\mathcal{KG}_{\mathcal{I}}$, and $\mathcal{KG}_{\mathcal{U}}$, with $\emptyset$ indicating no side information. On DBbook, $\mathcal{KG}_{\mathcal{U}}^{+}$ performed best, highlighting the value of LLM-inferred user preferences in sparse settings. On ML1M, $\mathcal{KG}_{\mathcal{I}}^{+}$ led, suggesting item features are more beneficial in denser datasets—supported by the strong performance of $\emptyset$. Combining all sources ($\mathcal{KG}^{+}$) sometimes hurt performance, likely due to noise from excessive entities and relations. Graph augmentation also improved novelty, especially on DBbook. Notably, LLM-generated KGs ($\mathcal{KG}_{\mathcal{I}}$, $\mathcal{KG}_{\mathcal{U}}$) performed on par with the original KG, underscoring LLMs' potential for knowledge creation. **Overall, RQ2 confirms that graph augmentation improves performance, with optimal gains depending on dataset characteristics.**

6. Conclusions

We proposed a general graph augmentation methodology for KARSs using LLM prompting to infer missing item features and user preferences. Focusing on DBpedia, our approach enriches KGs with LLM-generated knowledge. Experiments, including ablation studies and baselines, validated its effectiveness. While LLM limitations like hallucinations exist, carefully designed prompts helped mitigate them, as supported by our results. Future work will explore adding richer context (e.g., item abstracts or plots) to improve prompt quality.

[2]https://github.com/swapUniba/gal-kars

Acknowledgments

This research is partially funded by the PNRR project FAIR—Future AI Research (PE00000013), Spoke 6—Symbiotic AI, under the NRRP MUR program supported by NextGenerationEU (CUP H97G22000210007), and the PHaSE project — Promoting Healthy and Sustainable Eating through Interactive and Explainable AI Methods, funded by MUR under the PRIN 2022 program - Finanziato dall'Unione europea - NextGeneration EU, Missione 4 Componente 1 (CUP H53D23003530006). in

Declaration on Generative AI

During the preparation of this work, the author did not use any AI tool.

References

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, 2018.
[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pages 722–735. Springer, 2007.
[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26:2787–2795, 2013.
[4] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.
[5] Zefeng Chen, Wensheng Gan, Jiayang Wu, Kaixia Hu, and Hong Lin. Data scarcity in recommendation systems: A survey. *ACM Trans. Recomm. Syst.*, 3(3), March 2025. doi: 10.1145/3639063. URL https://doi.org/10.1145/3639063.
[6] Janneth Chicaiza and Priscila Valdiviezo-Diaz. A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions. *Information*, 12(6):232, 2021.
[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.
[9] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.
[10] Pranjal Kumar. Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260, 2024.
[11] Pasquale Lops, Marco de Gemmis, Giovanni Semeraro, Cataldo Musto, Fedelucio Narducci, and Massimo Bux. A semantic content-based recommender system integrating folksonomies for personalized access. *Web Personalization in Intelligent Environments*, pages 27–47, 2009.
[12] Pasquale Lops, Marco Polignano, Cataldo Musto, Antonio Silletti, and Giovanni Semeraro. Clayrs: An end-to-end framework for reproducible knowledge-aware recommender systems. *Information Systems*, 119:102273, 2023.
[13] Cataldo Musto, Tiziano Franza, Giovanni Semeraro, Marco De Gemmis, and Pasquale Lops. Deep content-based recommender systems exploiting recurrent neural networks and linked open data.

In *Adjunct Publication of the 26th conference on user modeling, adaptation and personalization*, pages 239–244, 2018.

[14] Cataldo Musto, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Context-aware graph-based recommendations exploiting personalized pagerank. *Knowledge-Based Systems*, 216:106806, 2021.

[15] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[16] Giuseppe Spillo, Cataldo Musto, Marco Polignano, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Combining graph neural networks and sentence encoders for knowledge-aware recommendations. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '23, page 1–12, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399326. doi: 10.1145/3565472.3592965. URL https://doi.org/10.1145/3565472.3592965.

[17] Giuseppe Spillo, Francesco Bottalico, Cataldo Musto, Marco De Gemmis, Pasquale Lops, and Giovanni Semeraro. Evaluating content-based pre-training strategies for a knowledge-aware recommender system based on graph neural networks. In *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, pages 165–171, 2024.

[18] Giuseppe Spillo, Cataldo Musto, Matteo Mannavola, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. Gal-kars: Exploiting llms for graph augmentation in knowledge-aware recommender systems. In *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*, pages 73–82, 2025.

[19] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116, 2011.

[20] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.

[21] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 968–977, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330836. URL https://doi.org/10.1145/3292500.3330836.

[22] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019.

[23] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019.

[24] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958, 2019.

[25] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the web conference 2021*, pages 878–887, 2021.

[26] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[27] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 806–815, 2024.

[28] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 12–22, 2024.

[29] Lixiang Xu, Yusheng Liu, Tong Xu, Enhong Chen, and Yuanyan Tang. Graph augmentation empowered contrastive learning for recommendation. *ACM Transactions on Information Systems*, 2024.

[30] Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. Exploring large language models for knowledge graph completion. *arXiv preprint arXiv:2308.13916*, 2023.

[31] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 1294–1303, 2022.

[32] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[33] Qianru Zhang, Lianghao Xia, Xuheng Cai, Siu-Ming Yiu, Chao Huang, and Christian S Jensen. Graph augmentation for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 557–569. IEEE, 2024.

[34] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[35] Yichi Zhang, Zhuo Chen, Lingbing Guo, Yajing Xu, Wen Zhang, and Huajun Chen. Making large language models perform better in knowledge graph completion. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 233–242, 2024.