Knowledge Graph-Enhanced Retrieval Augmented Generation for E-Commerce

Zihao Xu[1], Zhejun Shen[2], Qunzhi Zhou[2] and Petar Ristoski[2]

[1]*Rutgers University, New Jersey, USA*
[2]*eBay Inc., San Jose, USA*

Abstract

Large Language Models (LLMs) have been proven to be highly effective in various language modeling tasks. However, LLMs still suffer from intrinsic limitations when it comes to capturing factual and up-to-date data. This is becoming a bigger challenge in organizations that work with proprietary data, which is updated on daily bases, and cannot be accessed by public LLMs for legal reasons. This often requires re-training proprietary LLMs within the organization, or fine-tuning public LLMs on specific tasks using internal data, which is time-consuming and rather costly. To address these challenges, Retrieval Augmented Generation (RAG) approaches have been introduced, which retrieve relevant knowledge from available data stores, leading to higher accuracy, and easy reuse of pre-trained public LLMs.

In this work we introduce a Knowledge Graph (KG)-enhanced retrieval augmented generation approach, tailored specifically for the e-Commerce domain. We use a relationship-rich inventory-based Knowledge Graph to identify the most relevant knowledge for the given input and task, using entity linking and KG embeddings, which is then injected in the LLM prompt. We combine the power of LLMs for natural language understanding and the power of KGs for quick and easy access to proprietary factual knowledge to generate high-quality results, omitting hallucinations and generic outputs. We evaluate our approach on three e-Commerce tasks, significantly outperforming baseline LLM models, in both zero-shot and instruction-tuned settings.

Keywords

eCommerce, Knowledge Graph, RAG, LLM

1. Introduction

Large Language Models have demonstrated exceptional capability across a variety of tasks, revolutionizing how we interact with machine-generated content. However, when dealing with proprietary data, significant challenges arise, notably due to the dynamic nature of such datasets, domain specificity, and legal restrictions on data accessibility. The daily updates of proprietary data, coupled with its restricted access, predispose public LLMs to inaccuracies and hallucinations, limiting their direct applicability in specialized domains.

To adapt LLMs for such constrained environments, traditional finetuning, including recent advances in knowledge editing [1], has been considered. However, this process tends to be both time-consuming and costly due to the proprietary characteristics of the data involved. Alternatively, Retrieval Augmented Generation [2] offers a promising solution by providing LLMs with timely and domain-specific information, thus enhancing their performance in specialized tasks. This approach not only circumvents the exhaustive demands of model finetuning but also proves more economical and efficient, particularly when modifications are confined to the knowledge database rather than the model itself.

Knowledge Graphs (KGs) are particularly suited for this approach due to their ability to organize structured, complex information about entities and their relationships [3]. They facilitate the integration of comprehensive knowledge into LLMs effectively.

In this work, we introduce a robust KG-enhanced RAG framework specifically tailored for the e-commerce domain. We construct a detailed, relationship-rich inventory-based KG and utilize it to extract pertinent information through entity linking and KG embeddings. This extracted knowledge is subsequently injected into the LLM prompt to generate precise responses, in several e-commerce tasks.

Our contributions are summarized as follows:

- We propose a novel KG-enhanced RAG framework that provides LLMs with seamless access to up-to-date, domain-specific data.
- We validate the effectiveness of our approach through experiments on three e-commerce tasks, demonstrating its superiority in both zero-shot and instruction-tuned settings.

The rest of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we introduce our approach for Knowledge Graph-enhanced retrieval augmented generation. In Section 4, we present an in-depth evaluation of our approach. We conclude with a summary and an outlook on future work.

2. Related Work

To augment Large Language Model with knowledge graph, there are generally two primary steps: extracting relevant subgraphs from the knowledge graph according to the query, and subsequently incorporating this information into the LLM.

Subgraph retrieval frameworks are divided into two types: non-agent-based and agent-based. The non-agent-based approach follows a set schema, including identifying relevant entities within the graph, constructing subgraphs, and pruning the results. Entity resolution can be performed using either rule-based systems or embedding-based representations [4, 5]. Subgraph construction varies from straightforward techniques such as one-hop graph retrieval to more sophisticated methods like Prize-Collecting Steiner Tree [6]. On the other hand, agent-based retrieval is characterized by the use of a decision-making agent, often an LLM, to guide the retrieval process [7, 8, 9]. This agent formulates a retrieval plan based on the initial query. The graph database then executes this plan and provides results. The agent evaluates the outcomes and iteratively refines the plan, engaging in multiple rounds of interaction with the graph database to enhance the retrieval quality.

Incorporating subgraph data into a Large Language Model can be achieved through the use of either hard or soft prompts. Hard prompts, also known as verbalization, involve translating graph information into natural language text, which is then appended to the input prompt of the LLM. KAPING [10] concatenates the subject, relation and object triple and directly appends it to input prompt. GNN-RAG [11] reasons over subgraph and retrieve the answer entities and incorporates question to answer entity paths. Given the proficiency of LLMs in interpreting natural language, no further training is required in this phase, enhancing efficiency. However, recent work [12] shows that plain text may not fully capture the complex structures of graphs. Soft prompts address this by converting subgraph information into a latent representation that is consistent with the LLM's intrinsic framework. The approach of GraphToken [13] involves freezing the LLM's parameters and training a Graph Neural Network to output graph encoding that are compatible with the LLM's embedding. Graph Neural Prompting[14] augments this process with cross-modality pooling and a projection mechanism alongside the GNN encoder. It also introduces a self-supervised entity-linking prediction loss to capture the inter-entity relations and structural nuances of the graph.

While there are multiple approaches in the literature using RAG, and KG-enhanced RAG solutions, to the best of our knowledge, our approach is the first approach in the literature to use KG-enhanced retrieval augmented generation for e-Commerce.

3. Methodology

In this section, we present our approach for Knowledge Graph-enhanced retrieval augmented generation, which is specifically tailored for e-Commerce applications. The architecture of our approach is shown in Figure 1. The framework accepts an e-Commerce task t, described in natural language, and a textual input q, which is usually a product title or a search query, on which the task needs to be executed. For example, in Figure 1 the model is asked to perform "aspect-value pairs extraction and inference" on a short product title "Apple 6.1 inch A17 pro". In the next step we perform entity linking on the input q,

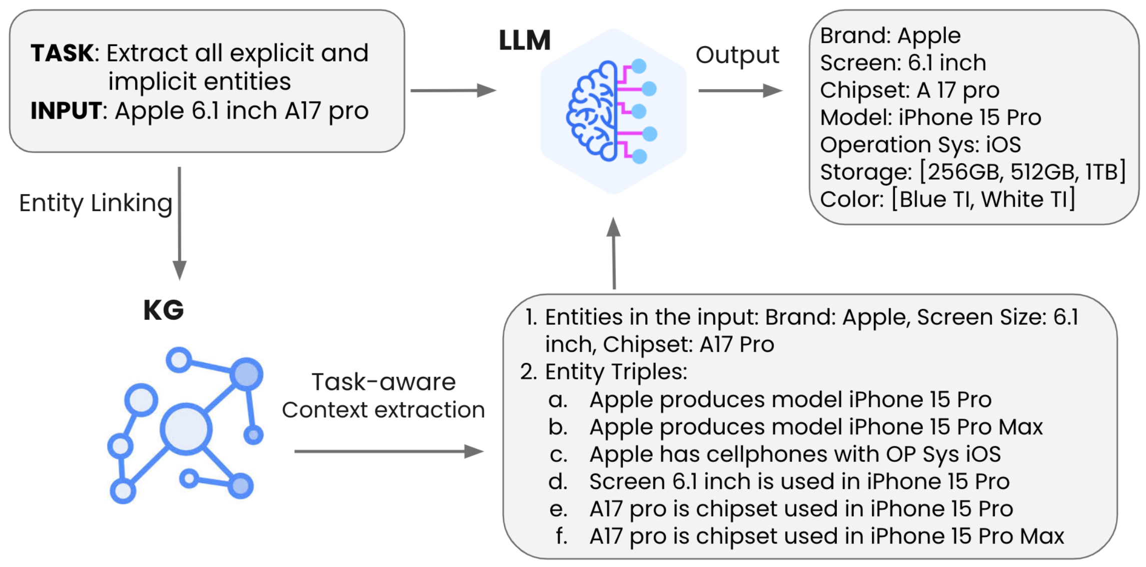
TASK: Extract all explicit and implicit entities
INPUT: Apple 6.1 inch A17 pro
LLM
Output
Brand: Apple
Screen: 6.1 inch
Chipset: A 17 pro
Model: iPhone 15 Pro
Operation Sys: iOS
Storage: [256GB, 512GB, 1TB]
Color: [Blue TI, White TI]
Entity Linking
KG
Task-aware Context extraction
1. Entities in the input: Brand: Apple, Screen Size: 6.1 inch, Chipset: A17 Pro
2. Entity Triples:
a. Apple produces model iPhone 15 Pro
b. Apple produces model iPhone 15 Pro Max
c. Apple has cellphones with OP Sys iOS
d. Screen 6.1 inch is used in iPhone 15 Pro
e. A17 pro is chipset used in iPhone 15 Pro
f. A17 pro is chipset used in iPhone 15 Pro Max

Figure 1: Knowledge Graph-Enhanced Retrieval Augmented Generation Architecture

to identify all KG entities V_q in the input. Based on the extracted entities V_q and the given task t, we extract relevant context C from the KG in natural language format. The context C is then concatenated with the task t and input q to construct the final prompt for the large language model [15]. Finally, the LLM generates the output in the requested format, depending on the task t.

3.1. Inventory-Based Knowledge Graph

A Knowledge Graph is a labeled, directed graph $G = (V, E)$, where V is a set of vertices, and E is a set of directed edges, where each vertex $v \in V$ is identified by a unique identifier, and each edge $e \in E$ is labeled with a label from a finite set of edge labels.

In this work we use a relationship-rich product Knowledge Graph, mined from seller provided data, which captures entities and relationships to model the whole product inventory in eBay Inc. [16]. The KG is mined from millions of product listings based on co-occurring aspect-value pairs in product listings, resulting in a directed weighted graph. More precisely, we generate a node in the graph for each aspect-value pair that occurs in product listings above a user specified threshold. To set the edge weights, for each co-occurring pair of aspect-value pair in at least one product listing, a normalized co-occurrence frequency is calculated. The co-occurrence frequency is normalized by the occurrence of both nodes in each direction, resulting in directed weights.

We use RDF2vec [17] to generate embedding vectors for all entities and relations. More precisely, we perform biased walks on the weighted graph to flatten the graph in sequences that can later be embedded using any language model. This approach is able to capture the neighborhood of each entity in a single vector, which then can be used for similarity calculation or context inference.

3.2. KG Context Extraction

To be able to extract relevant context C for the given task t, we first identify all KG entities in the input q. To do so, we use a proprietary entity linking pipeline [18]. Entity linking identifies textual mentions of named entities in the input, and aligns them to their corresponding entities in the KG. It resolves the lexical ambiguity of textual mentions and determines their concrete meaning. The output of the entity linking pipeline is a set of extracted entities for the given input q, $V_q = \{v_1, v_2, ..., v_n\}$. Once the KG entities are extracted, we can easily access all related information, including information about each entity, as well as their surrounding neighborhood structure and neighboring entities.

In the context C we first add all identified entities E_q using their normalized label in natural language.

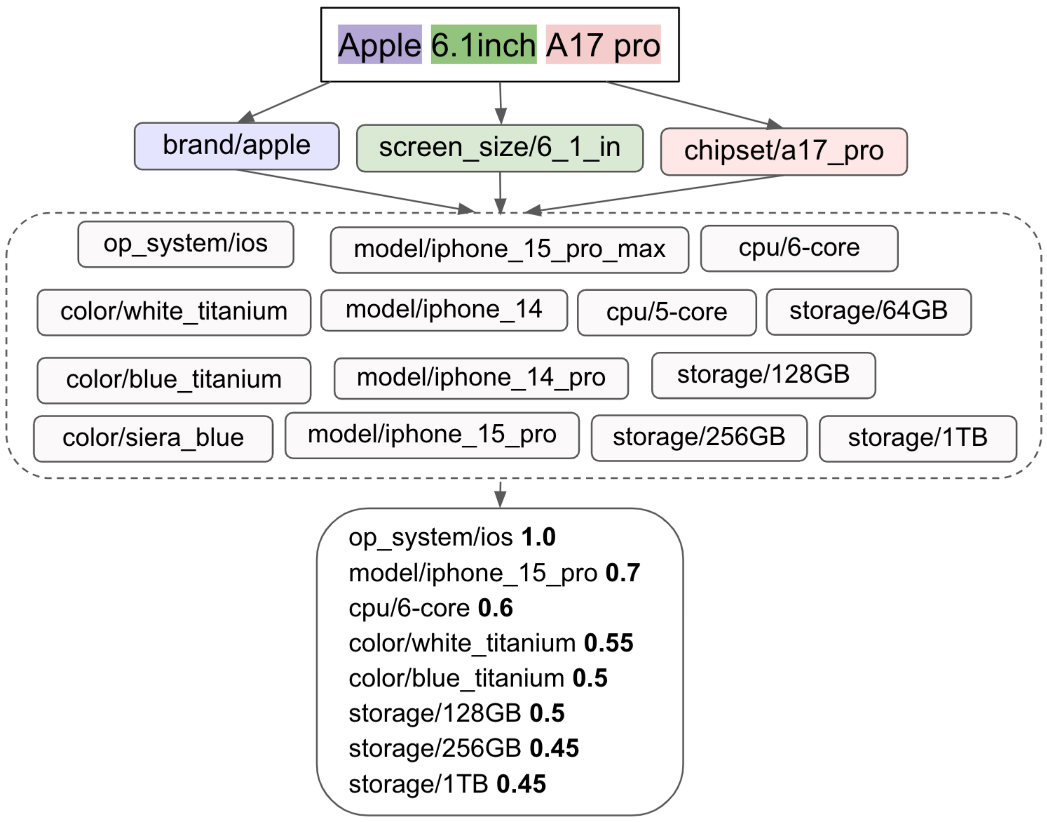

Figure 2: Extracting neighboring entities from the KG.

For example, in Figure 1, we identified "Brand: Apple, Screen Size: 6.1 inch, Chipset: A17 Pro". Furthermore, we are able to attach two different types of context, (i) neighboring entities, and (ii) semantically similar entities.

3.2.1. Extracting Neighboring Entities from KG

To identify relevant context for the given task t, we explore the direct neighbours of the extracted set of entities V_t. To do so, for each entity $v_t \in V_t$, we retrieve the direct neighbouring entities U_t, including the edge weights, both incoming and outgoing, resulting in a set of triples $U_t = \{(u_1, e_{vu1}, e_{uv1}), (u_1, e_{vu2}, e_{uv2}), ...(u_n, e_{vun}, e_{uvn})\}$, where u_n is the neighbouring entity, e_{vun} is the outgoing edge weight, and e_{uvn} is the incoming edge weight. Once all the neighboring entities are extracted, we iterate over the complete list of entities U and aggregate the edge weights, for both outgoing and incoming edges, the outgoing score for each neighbouring entity u is calculated as $w_{uo} = \sum_{i=1}^{n} e_{vui}/|U|$, and the incoming score is calculated as $w_{uin} = \sum_{i=1}^{n} e_{uvi}/|U|$. Then the final score for each entity u is calculated as $w_u = (w_{uo} + w_{uin})/2$. All the entities are sorted descending by their score, and the top-K entities are selected.[1] Then for each neighboring entity we generate a factual knowledge in natural language. Figure 2 shows the extraction of neighbouring entities for the previous example input. In the first step we extract the neighbouring subgraph, which is then aggregated and only the top-K entities are returned as the final context. Then for each neighboring entity we generate a fact, e.g., "Brand Apple produces cellphones with iOS operating system".[2] From the example we can see that the most relevant knowledge is surfacing to the top, e.g., given the input we can infer that the model is "iPhone 15 Pro", with operating system "iOS", with "hexa-core" CPU, and multiple options for the colors and storage capacity.

[1]K depends on the context window used in the LLM.
[2]Note that the KG is divided by category, which allows us to construct rules for improved verbalizing of the triples, e.g., in this case the category "cellphones" is used.

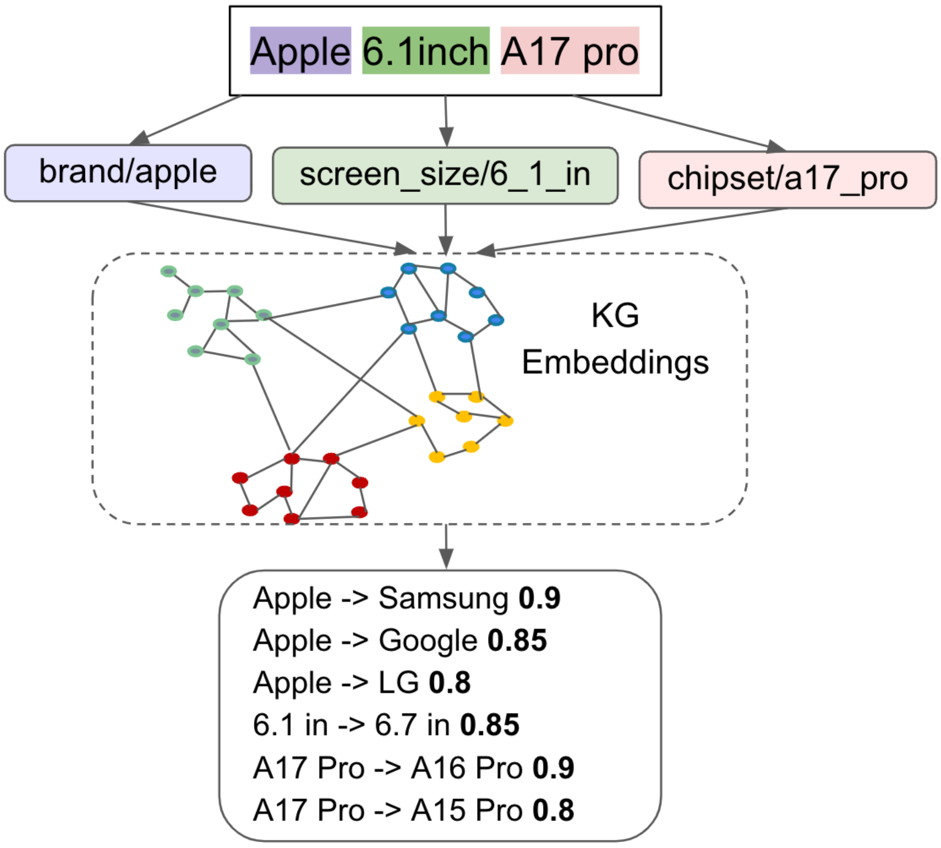

Figure 3: Extracting semantically similar entities using KG embeddings.

3.2.2. Extracting Semantically Similar Entities from KG

In many e-commerce tasks, such as query expansion or rewriting, it is crucial to identify semantically similar entities to the entities in the input. To do so, we use the previously built KG embeddings to calculate cosine similarity between the extracted set of entities V_t and the remaining entities in the KG. For each extracted entity v_t we identify the top-M semantically similar entities from the graph $U_s = \{u_1, u_2, ..., u_m\}$.[3] Then for each entity we verbalize the set of similar entities and generate the final context. For example, Figure 3 shows that for the previously extracted "Brand: Apple" we generate the following context "Brand Apple is similar to: Samsung, Google and LG". This context is appended to the previously extracted context C.

3.3. LLM Prompting

The standard use of LLMs for most of the tasks, is through prompting [15]. In our approach, we concatenate the input task t, the textual input q and the KG context C, which could be a combination of the previously described context retrieval approaches, to form the final prompt P. The final prompt P is then provided as an input to the LLM L. Such prompts can be used with most of the open source models, such as Llama[4] and Mistral[5], or paid APIs, such as ChatGPT[6].

[3]The number of entities depends on the LLM context window size.
[4]https://huggingface.co/meta-llama
[5]https://docs.mistral.ai/getting-started/models/
[6]https://openai.com/chatgpt/

Table 1
Aspect-value pairs extraction and inference results.

Model	Precision	Recall	F-Score
0-shot LLM	33.48%	8.37%	12.59%
0-shot KG-RAG LLM	**57.59%**	**31.47%**	**39.49%**
Tuned LLM	70.80%	62.65%	65.48%
Tuned KG-RAG LLM	**72.55%**	**64.73%**	**67.53%**

4. Experiments

We evaluate our approach on three e-commerce applications: (i) aspect-value pairs extraction and inference from product titles, (ii) product title generation, and (iii) query reformulation. On all three tasks, we compare 4 different settings: (i) zero-shot LLM generation, (ii) zero-shot KG-RAG generation, (iii) instruction-tuned LLM generation, and (iv) instruction-tuned KG-RAG generation. As a base LLM we are using Meta-Llama-3-8B-Instruct.[7] To train the models in a instruction-tune setting, we use low-rank adaptation (LoRA) [19].

4.1. Aspect-Value Pairs Extraction and Inference from Product Titles

When listing new products on most of the e-commerce platforms, besides title and description, the sellers are asked to provide product specifics in the format of aspect-value pairs. While having detailed product specifics is crucial for surfacing an item to the buyers, and having positive transaction metrics, requesting the sellers to fill out product specifics is the most common reason for the sellers to abandon the listing flow. Primarily there are three main reasons: (i) sellers are not familiar with all the product specifics they are asked to fill out (e.g. model numbers), (ii) the product specifics are redundant to what the seller has already provided in the product title or description, and (iii) the product specifics are obvious and could be inferred from the product title or description (e.g. if the seller already specified they are selling "iPhone", the brand of the product can be easily inferred to "Apple"). To ease the listing process, most e-commerce platforms assist the sellers to fill out the product specifics, using aspect-value extraction and inference, using different NLP and LLM-based solutions.

To evaluate our proposed approach, we annotated a random set of 10,000 listings from eBay Inc. inventory, using an independent labeling agency. For each listing, the human judges were shown the title, image and full description of the listings, then they were instructed to extract and infer all aspect-value pairs. The aspect-value pairs with inter-rater agreement of at least 60% of the annotators were considered as ground-truth. The final input for the task is the seller provided listing title, and the expected output is a list of aspect-value pairs.

For both, zero-shot and instruction-tuned models, we use only the title as input, and we task the models to extract and infer all the aspect-value pairs. For the zero-shot models we use the whole dataset as test dataset, while for the instruction-tuned models we use 80%-20% train-test split. For the KG-RAG LLM models, to generate the context we use the "KG neigboring entities" context, as explained in Section 3.2. To evaluate the performance of the models, we use Precision, Recall and F-Score. The results are shown in Table 1. From the results we can observe that the zero-shot KG-RAG LLM approach significantly outperforms the zero-shot LLM model. After instruction-tuning the models, the performance difference is shrinking, however the tuned KG-RAG LLM still significantly outperforms the tuned LLM model without KG context. It is apparent that the KG context we are extracting is highly relevant for the input products, and we are able to extract and infer higher number of aspect-value pairs (recall), with higher precision, compared to the baseline LLM models.

[7]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

Table 2
Product title generation results.

Model	BLEU	Jaccard
0-shot LLM	50.30%	23.54%
0-shot KG-RAG LLM	**56.05%**	**27.39%**
Tuned LLM	58.82%	29.39%
Tuned KG-RAG LLM	**64.45%**	**32.99%**

4.2. Product Title Generation

When listing new products, sellers are requested to add a title that summarizes the most important specifics of the product. On the eBay platform, sellers usually start the listing flow by issuing a search query which allows them to browse the existing inventory for potential matches, which will allow them to copy an existing product, instead of generating a new product from scratch. In the cases where the sellers don't find a match, they need to proceed to the next step where they have to provide a full title, description and item specifics. Given that they already provided a search query, we are trying to assist them with the title generation, i.e., we use the short search query to generate a buyer-attractive title that contains the most relevant information. For example, given the search query "124300" (which is a wristwatch reference number), ideally we would like to generate a title like "Rolex Oyster Perpetual 124300 Black Dial 41mm Stainless Steel".

To evaluate the models, we generate a dataset from the eBay inventory, i.e., we randomly sample 5,000 listing titles, for which we extract the initial seller search query and the final listing title. We make sure that all the tokens from the search query are present in the final title, and that the initial search query is less than 5 tokens, and the final title is larger than 5 tokens. The final input for the task is the initial short seller provided title, and the output is the final listing title.

For both, zero-shot and instruction-tuned models, we use the search query as input, and we task the models to generate the final title. For the zero-shot models we use the whole dataset as test dataset, while for the instruction-tuned models we use 80%-20% train-test split. For the KG-RAG LLM models, to generate the context we use the "KG neigboring entities" context, as explained in Section 3.2.

To evaluate the performance of the models, we use evaluation metrics commonly used in generation and translation tasks, i.e., BLEU score and Jaccard score. The results are shown in Table 2. The zero-shot KG-RAG model significantly outperforms the zero-shot LLM model, on both metrics, and the tuned KG-RAG model significantly outperforms the tuned LLM model. As in the previous task, the added KG context provides relevant information to the LLM to construct richer title, which is more attractive to the buyers.

4.3. Query Reformulation

The main task of an e-commerce search engine is to semantically match the user query to the product inventory and retrieve the most relevant items that match the user's intent. This task is not trivial as often there can be a mismatch between the user's intent and the product inventory for various reasons. To bridge the semantic gap between the user's intent and the available product inventory query rewriting approaches are used. Such approaches use a combination of token dropping, replacement and expansion. In this work, we focus on token replacement for query reformulation, which has been proven to be of great importance for low-inventory recovery, as well as recommendation. For example, if the user is searching for "Nike sneakers" and we cannot retrieve enough items from the inventory, instead of showing empty result page to the user, we can rewrite the query in a way that we still show relevant results to the user, such as "Adidas sneakers" or "Puma sneakers". In this case the reformulation was done by pivoting the brand entity, but it can be done on different types of entities, or multiple entities at the same time.

We evaluate the models on a 10,000 random sample used in [20]. The datasets is compiled from user search logs from eBay. The datasets consist of source and target user query pairs. To identify

Table 3
Query reformulation results.

Model	Recall@5
0-shot LLM	4.64%
0-shot KG-RAG LLM	**39.16%**
Tuned LLM	50.17%
Tuned KG-RAG LLM	**63.31**%

such query pairs, we track user sessions in which a user first issued a query, the source query, which retrieved less than 100 results, and the user didn't click on any item, then within the same session the user reformulated the query, the target query, and then clicked and/or purchased some of the resulting products. For each of the source-target query pairs we identify the entity type on which the replacement took place using our entity resolution approach.

For both, zero-shot and instruction-tuned models, we use the source query and the entity type on which we want to perform the replacement as input, and we task the models to generate 5 rewrites, conditioned on the entity type. For the zero-shot models we use the whole dataset as test dataset, while for the instruction-tuned models we use 80%-20% train-test split. For the KG-RAG LLM models, to generate the context we use the "semantically similar entities" context, as explained in Section 3.2.

To evaluate the model performance, we use Recall@5, i.e., we compare if any of the top-5 rewrites generated by the model are exact match with the target query in the ground truth data. The results are shown in Table 3.

The zero-shot KG-RAG model significantly outperforms the zero-shot LLM model, with a rather big margin. The tuned KG-RAG model also significantly outperforms the tuned LLM model. The results show that the KG context is providing highly-relevant information about the similarities between the entities to the LLM, which is able to correctly identify the entity in the input and replace it with the most similar and relevant entity from the KG context.

5. Conclusion

In this work, we have presented a robust, knowledge-graph enhanced Retrieval-Augmented Generation framework for application in the e-Commerce domain. This framework is designed to be task-agnostic. We conducted evaluations on three high-priority e-Commerce tasks. By injecting the up-to-date and plain-text knowledge directly to the prompt, zero-shot KG-enhanced RAG significantly outperforms the 0-shot LLM. The results are even comparable with those of fine-tuned models. This shows that combining the LLM capabilities for natural language understanding and the KG capabilities for easy access to up-to-date factual data generates high-quality results.

Moving forward, we plan to further evaluate this framework across additional e-Commerce tasks, such as recommendation, smart filtering, and AI shopping agents. Another future direction of work is to expand the approach to different languages.

References

[1] S. Wang, Y. Zhu, H. Liu, Z. Zheng, C. Chen, et al., Knowledge editing for large language models: A survey, arXiv preprint arXiv:2310.16218 (2023).

[2] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, Retrieval-augmented generation for large language models: A survey, arXiv preprint arXiv:2312.10997 (2023).

[3] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (CSUR) 54 (2021) 1–37.

[4] K. Soman, P. W. Rose, J. H. Morris, R. E. Akbas, B. Smith, B. Peetoom, C. Villouta-Reyes, G. Cerono, Y. Shi, A. Rizk-Jackson, S. Israni, C. A. Nelson, S. Huang, S. E. Baranzini, Biomedical knowledge graph-optimized prompt generation for large language models, 2024. URL: https://arxiv.org/abs/2311.17330. arXiv:2311.17330.

[5] Z. Xu, M. J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang, Z. Li, Retrieval-augmented generation with knowledge graphs for customer service question answering, ArXiv abs/2404.17723 (2024). URL: https://api.semanticscholar.org/CorpusID:269449459.

[6] X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, B. Hooi, G-retriever: Retrieval-augmented generation for textual graph understanding and question answering, arXiv preprint arXiv:2402.07630 (2024).

[7] L. LUO, Y.-F. Li, R. Haf, S. Pan, Reasoning on graphs: Faithful and interpretable large language model reasoning, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=ZGNWW7xZ6Q.

[8] J. Jiang, K. Zhou, W. X. Zhao, Y. Song, C. Zhu, H. Zhu, J.-R. Wen, Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph, ArXiv abs/2402.11163 (2024). URL: https://api.semanticscholar.org/CorpusID:267751414.

[9] J. Jiang, K. Zhou, Z. Dong, K. Ye, X. Zhao, J.-R. Wen, StructGPT: A general framework for large language model to reason over structured data, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 9237–9251. URL: https://aclanthology.org/2023.emnlp-main.574. doi:10.18653/v1/2023.emnlp-main.574.

[10] J. Baek, A. F. Aji, A. Saffari, Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, in: ACL 2023 Workshop on Matching Entities, 2023.

[11] C. Mavromatis, G. Karypis, Gnn-rag: Graph neural retrieval for large language model reasoning, ArXiv abs/2405.20139 (2024). URL: https://api.semanticscholar.org/CorpusID:270123131.

[12] B. Fatemi, J. Halcrow, B. Perozzi, Talk like a graph: Encoding graphs for large language models, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=IuXR1CCrSi.

[13] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, J. Halcrow, Let your graph do the talking: Encoding structured data for llms, 2024. URL: https://arxiv.org/abs/2402.05862. arXiv:2402.05862.

[14] Y. Tian, H. Song, Z. Wang, H. Wang, Z. Hu, F. Wang, N. V. Chawla, P. Xu, Graph neural prompting with large language models, in: AAAI 2024, 2024. URL: https://www.amazon.science/publications/graph-neural-prompting-with-large-language-models.

[15] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, ACM Computing Surveys 55 (2023) 1–35.

[16] P. Ristoski, S. Kandasamy, A. Matiushkin, S. Kamath, Q. Zhou, Wisdom of the sellers: Mining seller data for ecommerce knowledge graph generation, in: European Semantic Web Conference, Springer, 2023, pp. 195–199.

[17] P. Ristoski, H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: International Semantic Web Conference, Springer, 2016, pp. 498–514.

[18] Q. Zhou, Z. Wu, J. Degenhardt, E. Hart, P. Ristoski, A. Mandal, J. Netzloff, A. Mandalam, Leveraging knowledge graph and deepner to improve uom handling in search., in: ISWC (Posters/Demos/Industry), 2021.

[19] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685 (2021).

[20] S. Farzana, Q. Zhou, P. Ristoski, Knowledge graph-enhanced neural query rewriting, in: Companion Proceedings of the ACM Web Conference 2023, 2023, pp. 911–919.