Towards a Self-Explainable Recommender System using Large Language Models and Knowledge Graphs

Antony Seabra, Claudio Cavalcante, Sergio Lifschitz

[1]Departamento de Informatica - PUC-Rio

amedeiros,cfrag,sergio@inf.puc-rio.br

Resumo. *Este artigo propõe uma abordagem inovadora para aprimorar sistemas de recomendação por meio da integração de Grafos de Conhecimento (Knowledge Graphs – KGs) e Modelos de Linguagem de Grande Escala (Large Language Models – LLMs), utilizando triplas no formato Resource Description Framework (RDF) e engenharia de prompts. O sistema aproveita o conhecimento estruturado dos KGs para fundamentar as recomendações geradas pelos LLMs com explicações interpretáveis. Avaliamos o sistema em três cenários: geração de recomendações de produtos para usuários individuais, identificação de candidatos a descontos promocionais e seleção de combinações ideais de produtos. Os resultados demonstram que o método proposto gera sugestões contextualmente relevantes, acompanhadas de justificativas claras e compreensíveis para seres humanos. Também discutimos os principais desafios enfrentados durante o desenvolvimento e apresentamos estratégias eficazes para superá-los. De modo geral, nossos achados destacam o potencial dos recomendadores explicáveis baseados em LLMs para superar os sistemas tradicionais em termos de eficácia e transparência.*

1. INTRODUCTION

Recommender systems have become an indispensable tool for a wide range of companies, from e-commerce giants to content streaming platforms, helping to personalize user experiences and drive customer engagement. By analyzing user preferences and behavior, these systems can suggest products, services, or content that users are most likely to appreciate, thereby enhancing satisfaction and increasing sales. The evolution of these systems has seen a significant shift towards more sophisticated approaches, with recent advancements in artificial intelligence (AI) playing a crucial role.

In particular, Large Language Models (LLMs) have seen rapid development in recent years, revolutionizing natural language processing (NLP) and expanding the capabilities of AI across various domains. These models, trained on vast amounts of text data, can understand and generate human-like text, making them valuable for tasks that require nuanced understanding and contextual reasoning. However, integrating LLMs into recommender systems presents unique challenges and opportunities, especially when combined with structured knowledge representations such as Knowledge Graphs (KGs).

Knowledge Graphs (KGs) offer a structured and semantically rich representation of information by organizing entities and their relationships in the form of RDF triples. This format enables the capture of intricate connections between users, products, and their attributes, making KGs highly valuable for enhancing both the precision and explainability of recommender systems. However, a key challenge lies in how to present

this structured knowledge in a way that LLMs can effectively interpret and leverage during generation. This paper investigates the intersection of KGs and LLMs to address three central questions: (P1) How can a Knowledge Graph be presented to an LLM in a way that maximizes its utility? (P2) How can a recommender system be constructed using RDF triples extracted from the KG and integrated with an LLM? (P3)e How can the LLM provide explainable recommendations that transparently convey the reasoning behind each suggestion? By exploring these questions, the study aims to advance explainable, knowledge-enhanced recommendation strategies.

Through a detailed exploration of these questions, we propose a novel approach that integrates RDF triples into the context of an LLM and employs prompt engineering to guide the model's recommendations. The rest of the paper is structured as follows: Section 2 presents a technical background. Our methodology is outlined in Section 3. Section 4 describes the proposed architecture for our recommender system, and Section 5 presents the evaluation we conducted. Section 6 discusses related work in the field, and Section 7 concludes with insights from our work and suggests directions for future research in this area.

2. BACKGROUND

2.1. Recommender Systems

According to [Aggarwal et al. 2016], Recommender Systems are essential components of online platforms, offering personalized suggestions by analyzing user preferences and behaviors. The main techniques include collaborative filtering, content-based filtering, and hybrid approaches. Collaborative filtering relies on user-item interaction data to identify similarities, providing effective recommendations even without item metadata, though it struggles with the cold start problem and data sparsity [Schafer et al. 2007]. In contrast, content-based filtering recommends items based on their attributes and a user's past behavior, performing well in structured environments but often resulting in limited diversity [Lops et al. 2011].

Hybrid systems aim to overcome the limitations of each method by combining them, improving both accuracy and recommendation diversity. These systems adapt to the context using techniques like blending and switching strategies, and are particularly effective in mitigating cold start scenarios. However, they can be complex to implement and computationally expensive [Zhang et al. 2019]. With the rise of deep learning, newer approaches such as neural collaborative filtering and knowledge graph integration have further boosted performance and explainability [Wang et al. 2019]. Despite these advances, challenges remain regarding transparency, fairness, and resource demands, highlighting the need for balanced research efforts focused on developing accurate, diverse, interpretable, and efficient recommender systems.

2.2. Knowledge Graphs

Knowledge Graphs (KGs) are powerful tools for representing structured knowledge by linking entities—such as people, places, and concepts—through semantically meaningful relationships [Hogan et al. 2021]. They provide a unified and interconnected view of information, supporting efficient querying, inference, and data integration across heterogeneous sources. The creation of KGs involves steps like entity and relationship extraction

from unstructured data [Shen et al. 2014], as well as graph embedding techniques that convert graph structures into low-dimensional vector representations to support machine learning applications [Nickel et al. 2015]. These capabilities enable KGs to support complex analytical tasks, including link prediction, semantic search, and entity resolution, making them valuable assets in AI systems [Ji et al. 2021].

In recommender systems, KGs offer significant advantages by enriching the recommendation process with contextual and relational knowledge. Rather than relying solely on user-item interactions, KG-enhanced systems can incorporate domain-specific knowledge, such as attributes of products or semantic connections between items and users [Zhang et al. 2019]. This allows for more precise, diverse, and personalized recommendations, as seen in systems that model relationships between movies, genres, actors, and viewer preferences. Additionally, KGs improve explainability by enabling traceable reasoning paths behind recommendations, which increases transparency and user trust. They also help mitigate the cold start problem by using semantic similarity to recommend newly introduced items. As a result, the integration of Knowledge Graphs enhances both the performance and interpretability of recommender systems.

2.3. Large Language Models

Large Language Models (LLMs) have revolutionized Natural Language Processing by combining powerful architectures—such as Transformers—with advanced training methods like Reinforcement Learning with Human Feedback (RLHF), enabling machines to generate coherent, human-like text [Vaswani et al. 2017]. Unlike earlier models like RNNs and LSTMs, Transformers use attention mechanisms to process entire input sequences in parallel, improving scalability and the handling of long-range dependencies [Pascanu et al. 2013]. More recently, LLMs have been extended through context augmentation, which incorporates external knowledge from structured and unstructured sources, such as Knowledge Graphs and RDF triples. This augmentation enables models to produce more accurate, domain-relevant, and explainable responses for tasks like personalized recommendations and decision support.

Prompt Engineering plays a crucial role in leveraging the full capabilities of LLMs by shaping their outputs through carefully designed instructions and embedded context [Seabra et al. 2024]. While instructions define task, tone, and format, inclusion of contextual information, especially in the form of structured data like RDF triples, allows LLMs to reason over complex semantic relationships. This is particularly valuable for question-answering and explainable AI applications, where the model must not only retrieve relevant information but also provide interpretable justifications. By embedding knowledge directly into prompts, developers can guide LLMs to produce outputs that are both semantically rich and closely aligned with user intent, enhancing their reliability and transparency in real-world scenarios [Wang et al. 2023].

3. RELATED WORK

There is a growing trend underscoring the intersection of Recommender Systems and Natural Language Processing, specially with LLMs serving as a powerful tool for advancing recommendation strategies. The emergence of LLMs has opened new frontiers in the recommender systems domain, as they possess the ability to comprehend and generate

human-like text, which has led to a growing number of studies exploring their potential in enhancing recommendation systems [Balloccu et al. 2024].

Knowledge Graphs (KGs) have gained attention for their ability to encode structured, semantic information, which can be invaluable in enhancing the reasoning capabilities of LLMs. Recent studies, like [Pan et al. 2024], have explored integrating KGs with LLMs to improve the quality of responses in various tasks, including question answering, entity extraction, and knowledge graph reasoning. Approaches typically involve either using KGs as input context for LLMs or leveraging LLMs for dynamic KG construction. According to [Pan et al. 2024], KGs can enhance LLMs by providing external knowledge for inference and interpretability. The synergy between KGs and LLMs has shown promising results in capturing rich, domain-specific knowledge and enhancing explainability, particularly in systems requiring complex reasoning.

On the other hand, numerous techniques have emerged to enhance the extraction abilities of LLMs, improving their effectiveness in various applications like question answering, knowledge retrieval, and reasoning tasks. Prompt engineering, Retrieval-Augmented Generation (RAG), GraphRAG and Text-to-SQL are among of these popular techniques. Prompt engineering has been increasingly recognized for its potential to significantly improve the performance of LLMs by instructing them to behave differently from their default. [White et al. 2023] demonstrated how carefully designed prompts can enable more precise responses from LLMs across a range of tasks, underscoring the importance of prompt design in leveraging model capabilities. According to the authors, prompt patterns significantly enrich the capabilities that can be created in a conversational LLM. Indeed, this approach is essential for guiding LLMs to understand and respond to queries more effectively, by encapsulating the query within a context that the model is more likely to comprehend and respond to accurately.

[Giray 2023] states that, by employing prompt engineering techniques, academic writers and researchers can unlock the full potential of language models, and that this discipline opens up new avenues for improving AI systems and enhancing their performance in a range of applications, from text generation to image synthesis and beyond. The authors presents the prompt components that can be manipulated by engineers to guide text generation of an LLM. These componentes include an instruction, a context, an input data and an output indicator. Instruction outlines what the LLM is expected to do, providing clear directions to guide the model's response. The context gives background information necessary for the model to generate relevant and informed responses. However, despite these advances, there remains a notable gap in the literature regarding systematic methods to explain the outputs generated by LLMs, particularly when they are augmented with external knowledge sources like Knowledge Graphs. Most existing works focus on improving accuracy or retrieval, while few address how to make the reasoning and generation processes of LLMs transparent and interpretable to end-users.

4. METHODOLOGY

Our proposal in this study is to use LLM's context and Prompt Engineering to build a recommender system based on a Knowledge Graph that integrates data on products, features, categories, brands, sales and users. Figure 5 shows a segment of this KG showing relationships between a "Notebook" and other entities such as user "David", brand "HP",

products ”Printer” and ”Router”, and feature ”portable”. The KG structures the relationships between these entities, which will enable the generation of a context to be presented to an LLM.

There are various types of relationships that exist within the KG used in our recommender system. The following table summarizes them. Each row represents a specific relationship type, connecting different types of entities. The relationship ”buy” indicates that a user has purchased a particular product. The relationship ”mention” originating in ”User” captures instances where a user has mentioned specific features of a product in their comments or reviews. It provides insight into what aspects of a product are important to users.

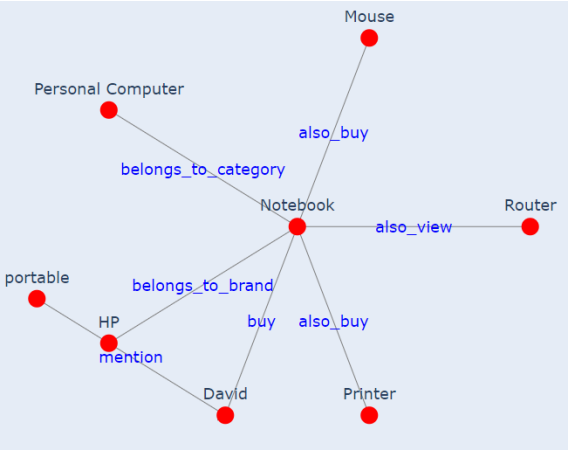

Figure 1. Consumer behaviour graph

The relationship ”mention” originating in ”Product” shows which features are associated with specific products based on user comments and reviews. It helps in understanding the attributes and characteristics commonly linked to products. The ”also_buy” relationship indicates that users who bought one product also bought another product. It is useful for identifying complementary products and making bundle recommendations. The ”also_view” relationship signifies that users who viewed one product also viewed another product. It helps in recommending products that are often considered together by users.

Table 1. Summary of Product Relationships and Associated Purchase probabilities.

Entity Type 1	Relationship	Entity Type 2
User	buy	Product
User	mention	Feature
Product	mention	Feature
Product	also_buy	Product
Product	also_view	Product
Product	belongs_to_category	Category
Product	belongs_to_brand	Brand

Finally, the ”belongs_to_category” and ”belongs_to_brand” relationships helps in organizing products and enabling category-based and brand-based recommendations.

One key-aspect of our methodology is that relationships in the KG are assigned weights, which influence the recommendation outcomes by prioritizing certain connections over others. These weights are derived from the significance of the relationships as determined by domain knowledge and data analysis. The table below presents these weights.

Table 2. Summary of Product Relationships and Associated Purchase Probabilities.

Relationship	Min Occ.	Purchase Prob.
also_buy	5	High
also_view	1	Medium
belongs_to_brand	-	Medium
belongs_to_category	-	Low

Our recommender system incorporates explainable AI principles, ensuring that the rationale behind each recommendation is transparent to and interpretable by users. By leveraging the relationships and their corresponding weights in the KG, the system can provide detailed explanations for its suggestions. For instance, a recommendation might be justified based on the strong association between a product and its brand, or the positive user comments it has received, or both. This explainability enhances user trust and satisfaction, as users can understand why certain products are being recommended.

Regarding to the extraction and transformation of data from the KG, we use RDF triples as the fundamental data units, comprising three components: a subject, a predicate, and an object. These triples encapsulate the semantic relationships between entities, forming the backbone of the knowledge representation. To integrate these RDF triples effectively within the context of the LLM, we transform them into a structured format that the LLM can readily interpret and utilize during inference. This transformation involves reformatting the RDF triples into a natural language or structured template that preserves the original semantic relationships while making the information accessible to the LLM. The following example illustrates an RDF triple and its corresponding formatted version:

In the final step, we leverage Prompt Engineering to steer the LLM in generating targeted recommendations. When RDF triples are formatted for inclusion in a prompt, they can subsequently be appended to questions directed at the LLM. The prompt itself presents the triplets variable alongside instructions on interpreting the relationships within these triplets, a structured approach to guide the assistant's understanding and response generation. It offers a concrete dataset for analysis and instruction on how to interpret the relationships.

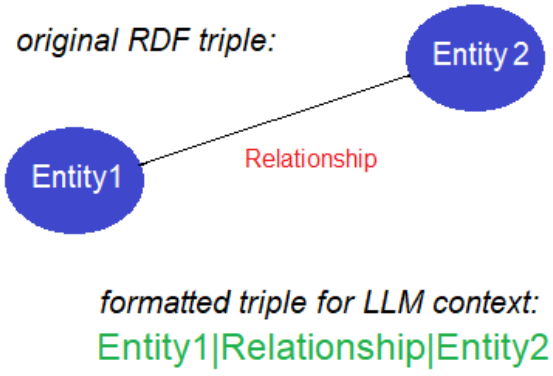

Figure 2. Formatted triple for LLM context

In addition to incorporating the formatted RDF triples, the prompts also include specific instructions for the LLM on how to interpret and weight the relationships represented in these triples. This ensures that the LLM not only understands the entities and their connections but also prioritizes certain relationships based on their relevance to the user's query. Furthermore, the user's question is integrated into the prompt, guiding the LLM to focus on the specific needs and preferences of the user. By combining these

elements—formatted triples, interpretative instructions, and the user query—the prompt provides a comprehensive framework that enables the LLM to generate highly tailored and contextually rich recommendations. This approach ensures that the LLM's outputs are not only aligned with the knowledge graph but also finely tuned to the nuances of the user's request.

5. ARCHITECTURE

The architecture of the application is structured around three main components: the backend layer for data acquisition and preparation, the integration layer with language models, and the user interface layer. The backend layer is responsible for the data aquisition and processing to prepare the formatted triples. To efficiently manage and process this data, we employed Apache Spark in conjunction with the GraphFrames library. Data is read from the CSV files to construct appropriate lists and then to transform these lists in Spark Dataframes, which are then used to construct a graph within the GraphFrame framework. This graph structure allows us to represent the data as RDF triples, where each edge in the graph corresponds to a triple consisting of a subject, predicate, and object.

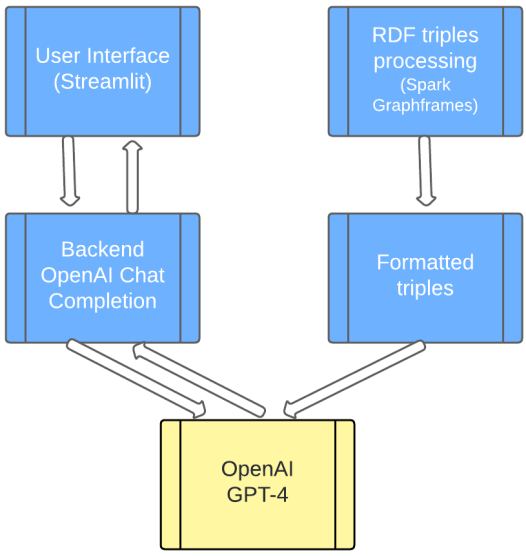

Figure 3. Recommender System Architecture

Once the data is prepared, the challenge is to format this information in a manner that can be seamlessly integrated into a prompt for the LLM. This involves creating a textual representation of the RDF triples that is both comprehensible to the LLM and capable of providing the necessary context for answering a given question. The formatting process includes the conversion of RDF triples into natural language sentences or structured statements that retain the semantic relationships encoded in the RDF format. This step ensures that the rich semantic information contained within the Knowledge Graph is preserved and made accessible to the LLM.

The Integration Layer with Language Models handles the interaction between the prepared data and the Large Language Model. It incorporates the formatted RDF triples into the context fed to the LLM, ensuring that the semantic relationships captured in the knowledge graph are effectively utilized during inference. Additionally, this layer implements Prompt Engineering techniques, where specific prompts are crafted to guide the LLM in interpreting and prioritizing relationships within the triples, as well as responding to the user's query with accurate and contextually rich recommendations.

The user interface layer is designed to provide a seamless interaction between the user and the recommender system. It not only facilitates user input but also displays

the recommendations generated by the LLM, ensuring that they are transparent and explainable. The user interface serves as the final point where the system's integration of knowledge graph data and language model reasoning is presented in an explainable user-friendly manner.

6. EVALUATION

The dataset used in our evaluation is a proprietary one, consisting of approximately 5,200 entities and 16,600 relationships. There are relationships in the dataset such as: "buy" relationships indicating purchases made by users, "mention" relationships reflecting product attributes mentioned by users, "also_buy" relationships denoting products that are frequently bought together, "also_view" relationships showing products that are often viewed together, "belongs_to_category" relationships classifying products into specific categories, and "belongs_to_brand" relationships linking products to their respective brands. To assess the explainability of the proposed recommender system, we evaluated it using the following three questions: (1) What products could David buy?, (2) Which product categories should be prioritized for a discount? and (3) Which two products would be the most suitable candidates for a bundle discount? Each of these questions was submitted to the system, and the results were analyzed with a focus on the system's ability to provide an explainable recommendation.

6.1. What products could David buy?

For this query, the system was asked to recommend potential products for David based on his previous purchases and mentions, as well as his preferences indicated by the RDF triples. The system incorporated formatted triples and relevant prompts to identify products that align with David's purchasing history, and presents the graph show below, indicating products and associated probabilities

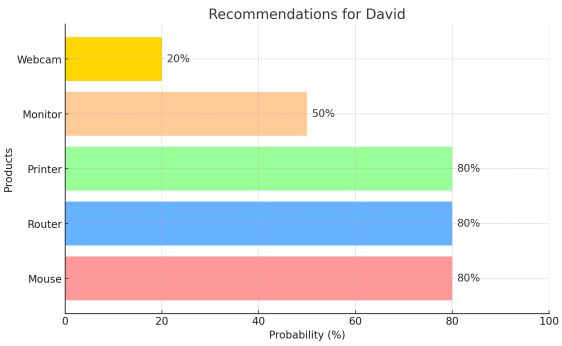

Figure 4. Recommendations for a particular user

The recommendation explains that, analyzing customer behavior patterns, the system identified that products commonly purchased alongside a Notebook, such as a Mouse, Router, and Printer, hold a high probability (80%) of being appealing to David. The strong connection between these items is reinforced by the 'also_buy' relation, which shows that these products are often bought together with a Notebook. Additionally, the Router is recommended with the same high probability due to its frequent co-viewing with a Notebook, as indicated by the 'also_view' relation. Furthermore, considering brand loyalty, the system suggests that David might be interested in purchasing a Monitor from the same brand (HP) as the Notebook, assigning it a medium probability (50%) under the 'belongs_to_brand' relation. Lastly, the system identifies a Webcam as a product in the same category as the Notebook (Personal Computer), although with a lower probability (20%), suggesting it as a less likely but still relevant option. We present other explainable recommendations in the following table.

User	Recommendation
Sophia	Sophia bought a 'USB Drive'. There is a high probability she would buy a 'Router' because both are linked by the 'also_buy' relationship.
Eva	Eva bought a 'Headset'. There is a high probability she would buy a 'Tablet' or 'Smartphone' because these products are linked by the 'also_buy' relationship.
Chris	Chris bought a 'Webcam'. There is a high probability he would buy a 'Keyboard' due to the 'also_view' relationship.
Liam	Liam bought a 'Mouse'. There is a high probability he would buy a 'Notebook' because both are linked by the 'also_buy' relationship.
Alice	Alice bought a 'Tablet'. There is a high probability she would buy a 'Headset' or 'Keyboard' as they are connected by the 'also_buy' relationship.

Table 3. Product Recommendations for users

6.2. Which product categories should be prioritized for a discount?

The recommender system suggests prioritizing discounts on products within the 'Input Device' category to increase sales. This category, which includes items like Mice, Keyboards, Webcams, Smartphones, Headsets, and Tablets, has a medium probability of leading to additional purchases when discounted. The medium probability indicates that customers who have purchased one product in this category are somewhat likely to buy another, especially if brands like Logitech, Samsung, and Sony are considered.

On the other hand, categories such as 'Storage Device', 'Output Device', and 'Personal Computer' exhibit low probabilities for additional purchases. Products in these categories, such as USB Drives, SSDs, Monitors, Printers, and Notebooks, typically have longer lifecycles or higher price points, which reduces the likelihood of repeat purchases or cross-category purchases.

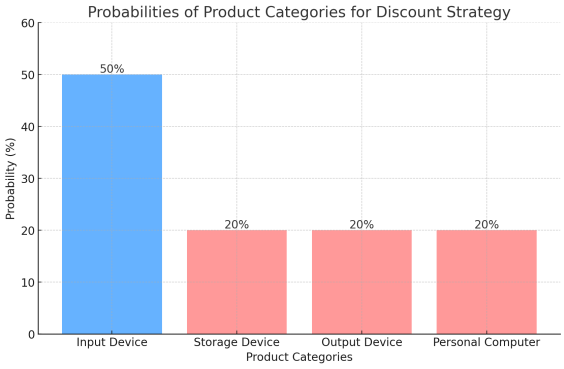

Figure 5. Recommendations for product categories

Figure 6 shows the probabilities for each product category in the context of a discount strategy. The 'Input Device' category stands out with a medium probability (50%), indicating it as the most promising category for increasing sales through discounts. The other categories 'Storage Device', 'Output Device', and 'Personal Computer', all have lower probabilities (20%), suggesting they are less likely to benefit from discounting.

6.3. Which two products would be the most suitable candidates for a bundle discount?

The recommender system identified four pairs of products as suitable candidates for a bundle discount, based on the analysis of relationships such as 'also_buy', 'also_view', 'belongs_to_brand', and 'belongs_to_category'. Here are the recommended pairs:

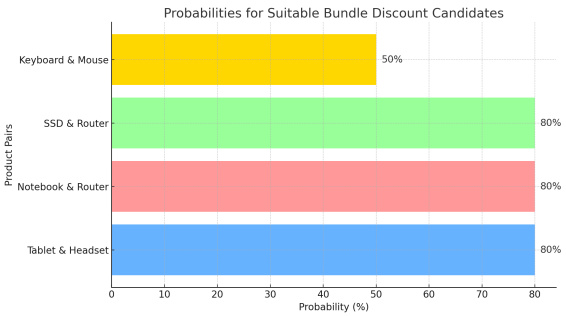

Figure 6. Recommendations for bundles

Tablets and Headsets are frequently bought together, as indicated by the also_buy relationship, making them ideal candidates for a bundle discount. Similarly, Notebooks and Routers show strong association through both also_view and also_buy links, suggesting that customers interested in one are likely to consider the other, reinforcing their bundling potential. Additionally, the combination of SSDs and Routers is supported by purchase data showing that buyers of Routers often acquire SSDs as well. Lastly, although the direct purchase correlation between Keyboards and Mice is moderate, they share the same brand (Logitech) and belong to the 'Input Device' category, indicating that bundling them could strategically increase cross-selling effectiveness. These insights support the system's data-driven bundling recommendations aimed at boosting sales and enhancing customer satisfaction. Figure 6 shows a bar graph representing the probabilities for each pair of products identified as suitable candidates for a bundle discount. The graph shows that the Tablet & Headset, Notebook & Router, and SSD & Router pairs all have a high probability of 80% for cross-purchasing, making them strong candidates for bundling. The Keyboard & Mouse pair has a slightly lower probability of 50%, but bundling them could still encourage additional sales due to their shared brand and category.

7. CONCLUSIONS

This article highlighted the pivotal role of integrating Knowledge Graphs (KGs) with Large Language Models (LLMs) to enhance explainability through prompt engineering and the semantic structuring of relationships. By leveraging the rich, structured information encoded in KGs—particularly through RDF triples and entity-relation semantics—LLMs can generate context-aware and interpretable outputs grounded in explicit knowledge. Prompt engineering plays a crucial role in this integration, guiding the model to access and reason over relevant relationships and attributes within the KG. This synergy enables not only more accurate responses but also the ability to provide human-understandable justifications, a key requirement in domains such as decision support, legal analysis, and personalized recommendations.

Future work should explore the application of these techniques across more complex and large-scale KGs to assess scalability, performance, and generalizability. Investigating how LLMs interpret and reason over deeper ontological structures and relation hierarchies can provide insights into their capabilities and limitations in semantic reasoning. As token limits expand and models become more capable of handling large contextual inputs, combining LLMs with detailed graph-based knowledge will become increasingly feasible and impactful. This direction promises to advance the development of explainable AI systems by bridging symbolic reasoning with generative models, fostering transparency and trust in automated decision-making processes.

References

Aggarwal, C. C. et al. (2016). *Recommender systems*, volume 1. Springer.

Balloccu, G., Boratto, L., Fenu, G., Malloci, F. M., and Marras, M. (2024). Explainable recommender systems with knowledge graphs and language models. In *European Conference on Information Retrieval*, pages 352–357. Springer.

Giray, L. (2023). Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., et al. (2021). Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2015). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., and Wu, X. (2024). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*, pages 291–324. Springer.

Seabra, A., Cavalcante, C., Nepomuceno, J., Lago, L., Ruberg, N., and Lifschitz, S. (2024). Contrato360: uma aplicação de perguntas e respostas usando modelos de linguagem, documentos e bancos de dados. In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 155–166. SBC.

Shen, W., Wang, J., and Han, J. (2014). Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, M., Wang, M., Xu, X., Yang, L., Cai, D., and Yin, M. (2023). Unleashing chatgpt's power: A case study on optimizing information retrieval in flipped classrooms via prompt engineering. *IEEE Transactions on Learning Technologies*.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38.