Semantic Prompting over Knowledge Graphs for Next-Generation Recommender Systems

Antony Seabra[a], Claudio Cavalcante[b] and Sergio Lifschitz[c]
Departamento de Informatica, PUC-Rio, Brazil

Keywords: Recommender Systems, Knowledge Graphs, Large Language Models (LLMs, Semantic Prompt Generation, RDF Triples, Natural Language Interfaces.

Abstract: This paper presents a novel recommender system framework that integrates Knowledge Graphs (KGs) and Large Language Models (LLMs) through dynamic semantic prompt generation. Rather than relying on static templates or embeddings alone, the system dynamically constructs natural language prompts by traversing RDF-based knowledge graphs and extracting relevant entity relationships tailored to the user and recommendation task. These semantically enriched prompts serve as the interface between structured knowledge and the generative capabilities of LLMs, enabling more coherent and context-aware suggestions. We validate our approach in three practical scenarios: personalized product recommendation, identification of users for targeted marketing, and product bundling optimization. Results demonstrate that aligning prompt construction with domain semantics significantly improves recommendation quality and consistency. The paper also discusses strategies for prompt generation, template abstraction, and knowledge selection, highlighting their impact on the robustness and adaptability of the system.

1 INTRODUCTION

Recommender systems play a key role in tailoring digital experiences across domains such as e-commerce, media streaming, and online services. By leveraging user profiles, contextual signals, and historical interactions, these systems aim to suggest items that align with user interests, thus boosting engagement and driving decision-making. Traditional recommendation approaches—ranging from collaborative filtering to content-based methods—have evolved significantly with the integration of semantic knowledge and natural language technologies.

Recent advances in Large Language Models (LLMs) have transformed the field of natural language processing, enabling models to interpret, generate, and reason over text with remarkable fluency. This progress has opened new avenues for building intelligent, conversational recommendation interfaces. However, LLMs alone lack domain-specific grounding, which can lead to generic or inconsistent suggestions when applied to structured decision-making scenarios.

Knowledge Graphs (KGs) offer a powerful mechanism to enrich recommendation processes with domain semantics. By organizing information into entities and relationships using formal representations such as Resource Description Framework (RDF) triples, KGs capture intricate, structured knowledge about products, users, and their interrelations. The integration of KGs with LLMs has the potential to combine the expressiveness of natural language with the precision of structured data.

In this work, we propose a hybrid recommendation framework that bridges KGs and LLMs through dynamic semantic prompt generation. Instead of statically encoding knowledge into embeddings or manually defining rules, our system dynamically traverses RDF graphs to extract relevant information, which is then used to formulate natural language prompts tailored to the recommendation task at hand. These prompts guide the LLM in generating contextually aligned and semantically grounded recommendations.

The central research questions addressed in this paper are:

P1. How can semantic knowledge from a Knowledge Graph be dynamically transformed into prompts that effectively guide an LLM?

[a] https://orcid.org/0009-0007-9459-8216
[b] https://orcid.org/0009-0007-6327-4083
[c] https://orcid.org/0000-0003-3073-3734

Seabra, A., Cavalcante, C. and Lifschitz, S.
Semantic Prompting over Knowledge Graphs for Next-Generation Recommender Systems.
DOI: 10.5220/0013741700003985

In *Proceedings of the 21st International Conference on Web Information Systems and Technologies (WEBIST 2025)*, pages 394-403
ISBN: 978-989-758-772-6; ISSN: 2184-3252

P2. How can the alignment between task-specific goals and KG-derived semantics improve the quality of recommendations generated by LLMs?

P3. What are the implementation strategies and architectural components required to integrate semantic prompting into a web-based recommender system?

To address these questions, we design and evaluate a recommendation pipeline that operationalizes RDF triples as input for LLM-guided reasoning via prompt engineering. The system is validated through three application scenarios involving product suggestion, promotional targeting, and bundling strategies.

The remainder of the paper is organized as follows: Section 2 provides background on recommender systems, semantic technologies, and LLM integration. Section 3 details our methodology for semantic prompt generation. Section 4 introduces the system architecture and implementation. Section 5 presents the experimental evaluation. Section 6 reviews related work, and Section 7 concludes with final remarks and directions for future research.

2 BACKGROUND

2.1 Recommender Systems

Recommender systems have become a crucial component of many online platforms, offering personalized suggestions to users based on their preferences and behaviors. Over the years, various types of recommender systems have been developed, each with unique strengths and weaknesses. The three main types are collaborative filtering, content-based filtering, and hybrid approaches. Each of these methods employs different techniques to generate recommendations and addresses different aspects of the recommendation problem (Ricci et al., 2010).

Collaborative filtering is one of the most widely used techniques in recommender systems. It works by analyzing user behavior and preferences, typically through user-item interaction matrices, to find similarities between users or items (Su and Khoshgoftaar, 2009). The two primary approaches within collaborative filtering are user-based and item-based filtering. User-based filtering recommends items to a user based on the preferences of similar users, while item-based filtering suggests items similar to those the user has previously liked. The strengths of collaborative filtering include its ability to provide recommendations without needing explicit content information and its effectiveness in leveraging the wisdom of the crowd. However, it suffers from the cold start problem, where new users or items with insufficient interactions are challenging to recommend accurately, and it can struggle with sparsity in the user-item interaction matrix (Schafer et al., 2007).

Content-based filtering, on the other hand, relies on the features of the items themselves to make recommendations (Lops et al., 2011). This approach builds user profiles based on the attributes of items they have previously interacted with and recommends new items that share similar characteristics. Content-based filtering is particularly effective in domains where item features are well-defined and structured, such as in recommending movies based on genres, actors, and directors. One of the main advantages of content-based filtering is its ability to handle the cold start problem more effectively for new items, as long as their features are known. However, it has limitations in terms of recommendation diversity, as it tends to suggest items that are too similar to those the user has already seen, potentially leading to a narrow user experience (Aggarwal et al., 2016).

Hybrid recommender systems combine the strengths of collaborative filtering and content-based filtering to overcome their individual limitations (Burke, 2002). By integrating multiple recommendation strategies, hybrid systems can provide more accurate and diverse suggestions. These systems can use various methods to combine recommendations, such as switching between techniques based on the context, weighting the contributions of different methods, or blending their outputs. Hybrid systems can address the cold start problem more effectively by using content-based approaches for new items and collaborative filtering for items with sufficient interaction data. They also tend to provide a better balance between relevance and diversity in recommendations. However, hybrid systems can be more complex to implement and require more computational resources, making them potentially more expensive to deploy and maintain (Zhang et al., 2019).

In recent years, the development of deep learning and advanced machine learning techniques has further enhanced the capabilities of recommender systems. Techniques such as neural collaborative filtering, graph-based recommendation, and the use of knowledge graphs have shown promising results in improving recommendation accuracy and explainability (He et al., 2017) and (Wang et al., 2019). These advanced methods leverage large-scale datasets and complex models to capture intricate patterns in user behavior and item characteristics. While they offer significant improvements in performance, they also introduce challenges related to model interpretability and the need for substantial computational resources. As

recommender systems continue to evolve, balancing accuracy, diversity, explainability, and resource efficiency remains a key focus for researchers and practitioners in the field (Wang et al., 2020).

2.2 Knowledge Graphs

Knowledge Graphs (KGs) are structured representations of knowledge that connect entities, such as people, places, and concepts, through relationships or edges. These graphs consist of nodes representing entities and edges depicting the relationships between them (Hogan et al., 2021). The primary purpose of KGs is to provide a comprehensive and interconnected view of knowledge, allowing for efficient querying and inference. Knowledge Graphs leverage semantic information to create meaningful connections and are widely used in various applications, including search engines, natural language processing, and AI systems. Their ability to integrate and organize vast amounts of heterogeneous data makes them valuable tools for managing complex information landscapes (Paulheim, 2017).

The construction of Knowledge Graphs involves several key processes, such as entity extraction, relationship extraction, and graph embedding. Entity extraction identifies and categorizes entities from unstructured data sources, while relationship extraction identifies the connections between these entities (Shen et al., 2014). Graph embedding techniques then transform the graph structure into low-dimensional vector spaces, enabling machine learning algorithms to process and analyze the data effectively. Knowledge Graphs can be manually curated, automatically generated, or created using a combination of both approaches (Nickel et al., 2015). The rich, interconnected nature of KGs enables advanced data analysis, supporting tasks like link prediction, entity resolution, and semantic search (Ji et al., 2021).

In the context of recommender systems, Knowledge Graphs enhance recommendation quality by providing additional contextual information and relationships between items. By integrating KGs, recommender systems can move beyond simple user-item interactions and incorporate richer data about item attributes, user preferences, and domain knowledge (Zhang et al., 2019). For instance, a movie recommendation system can leverage a KG to understand relationships between actors, directors, genres, and user ratings, allowing it to generate more nuanced and accurate recommendations. The semantic relationships captured in KGs enable the system to make inferences and discover hidden patterns, leading to improved recommendation diversity and relevance (Cao et al., 2019).

Furthermore, Knowledge Graphs facilitate explainability in recommender systems by providing transparent and interpretable insights into the recommendation process. When a recommendation is made, the system can trace the reasoning through the KG, offering explanations such as ”This movie is recommended because it shares similar themes with movies you have previously enjoyed and features an actor you frequently watch” (Wang et al., 2018). This transparency enhances user trust and satisfaction, as users can understand the rationale behind the recommendations. Additionally, KGs help address the cold start problem by leveraging the semantic relationships to recommend new items based on their attributes and connections within the graph. As a result, integrating Knowledge Graphs into recommender systems not only improves overall accuracy but also boosts user engagement and trust through explainable AI.

2.3 Large Language Models

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP) with their ability to understand and generate human-like text. At the heart of the most advanced LLMs is the Transformers architecture, a deep learning model introduced in the seminal paper *Attention Is All You Need* by (Vaswani et al., 2017). Transformers leverage a mechanism called *attention*, which allows the model to weigh the influence of different parts of the input data at different times, effectively enabling it to focus on relevant parts of the text when making predictions.

Prior to Transformers, Recurrent Neural Networks (RNNs) and their variants like Long Short-Term Memory (LSTM) networks were the standard in NLP. These architectures processed input data sequentially, which naturally aligned with the sequential nature of language. However, they had limitations, particularly in dealing with long-range dependencies within text due to issues like vanishing gradients (Pascanu et al., 2013). Transformers overcome these challenges by processing all parts of the input data in parallel, drastically improving the model’s ability to handle long-distance relationships in text.

Chat models, a subset of LLMs, are specialized in generating conversational text that is coherent and contextually appropriate. This specialization is achieved through the training process, where the models are fed vast amounts of conversational data, enabling them to learn the nuances of dialogue. ChatGPT, for instance, is fine-tuned on a dataset of conversational exchanges and it was optimized for dialogue

by using Reinforcement Learning with Human Feedback (RLHF) - a method that uses human demonstrations and preference comparisons to guide the model toward desired behavior (OpenAI, 2023a).

The transformative impact of LLMs, and particularly those built on the Transformers architecture, has been profound. By moving away from the constraints of sequential data processing and embracing parallelization and attention mechanisms, these models have set new standards for what is possible in the realm of NLP. With the ability to augment generation with external data or specialize through fine-tuning, LLMs have become not just tools for language generation but platforms for building highly specialized, knowledge-rich applications that can retrieve information in a dialogue-like way, find useful information and generate insights for decision making.

The ability to augment the generation capabilities of LLMs using enriched context from external data sources is a significant advancement in AI-driven systems. An LLM context refers to the surrounding information provided to a LLM to enhance its understanding and response generation capabilities. This context can include a wide array of data, such as text passages, structured data, and external data sources like Knowledge Graphs. Utilizing these external data sources allows the LLM to generate more accurate and relevant responses without the need for retraining. By providing detailed context, such as product attributes, user reviews, or categorical data, the model can produce insights that are tailored and contextually aware.

2.4 Prompt Engineering

One key aspect of providing contexts to LLMs is the ability of designing and optimizing prompts to guide LLMs in generating the answers. This is what is called Prompt Engineering. Its main goal is to maximize the potential of LLMs by providing them with instructions and context (OpenAI, 2023b).

In the realm of Prompt Engineering, instructions are the crucial first steps. Through them, engineers can detail the roadmap to an answer, outlining the desired task, style and format for the LLM's response (White et al., 2023). For instance, To define the style of a conversation, a prompt could be phrased as "Use professional language and address the client respectfully" or "Use informal language and emojis to convey a friendly tone". To specify the format of dates in answers, a prompt instruction could be "Use the American format, MM/DD/YYYY, for all dates".

On the other hand, as mentioned earlier, context refers to the information provided to LLMs alongside the core instructions. The most important aspect of a context is that it can provide information that supports the answer given by the LLM, and it is very useful when implementing question-answering systems. This supplemental context can be presented in various formats. One particularly effective format is RDF triples, which represent information as subject-predicate-object statements. RDF triples are a standardized way of encoding structured data about entities and their relationships, making them ideal for embedding precise information into prompts. By including RDF triples in a prompt, we can clearly convey complex relationships and attributes in a format that the LLM can easily process, leading to more accurate and relevant responses. According to (Wang et al., 2023), prompts provide guidance to ensure that ChatGPT generates responses aligned with the user's intent. As a result, well-engineered prompts greatly improve the efficacy and appropriateness of ChatGPT's responses.

3 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 3 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.

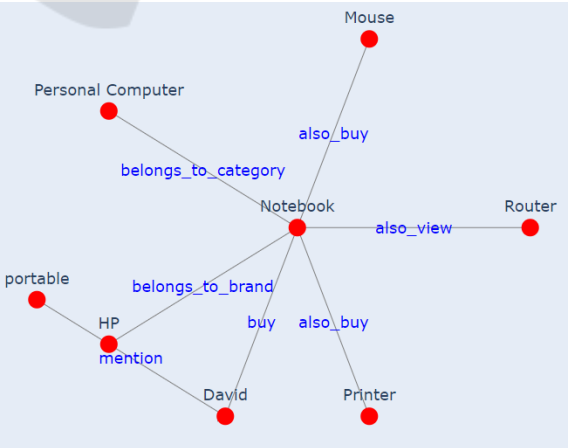

Figure 1: Consumer behaviour graph.

There are various types of relationships that ex-

ist 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.

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

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. 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:

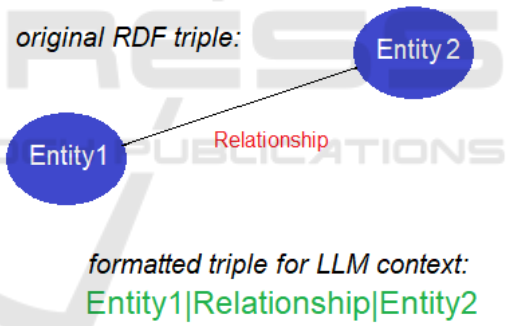

Figure 2: Formatted triple for LLM context.

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.

```
messages = [
  {"role": "system",
  "content": "You are a helpful assistant."},
  {"role": "user",
  "content": f"Consider the following
```

```
relationships represented as triplets:
\n{formatted_triplets}"},
{"role": "user",
"content": "Consider that if a person has
bought a product that participates in a
relation 'also_buy' with other products,
there is a high probability for this
person to buy these other products."},

...
```

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.

4 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. The data used in this study was sourced from CSV files, which contain the entities and the relationships between them.

To efficiently manage and process this data, we employed Apache Spark in conjunction with the GraphFrames library. The data processing begins with initializing a Spark session, providing the foundation for creating a graph-based structure, where vertices represent entities, and edges represent the relationships. 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.

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.

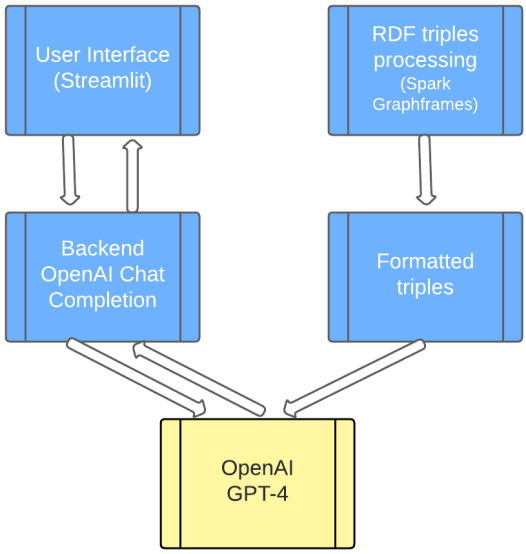

Figure 3: Recommender system architecture.

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 enable seamless interaction between the user and the recommender system, serving as a bridge between user intent and semantically grounded recommendations. It not only facilitates the input of natural language queries but also presents the output of the system in a transparent and interpretable manner. By surfacing recommendations that are generated through the reasoning capabilities of the LLM and grounded in structured knowledge graph data, the interface en-

sures that semantic relationships and decision rationale are clearly conveyed to the user.

5 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 how effectively the system leveraged semantic knowledge to generate contextually relevant and semantically coherent recommendations. Particular attention was given to the system’s ability to incorporate structured domain relationships into the output, ensuring that the suggested items aligned with the underlying meaning and intent of each query.

5.1 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

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

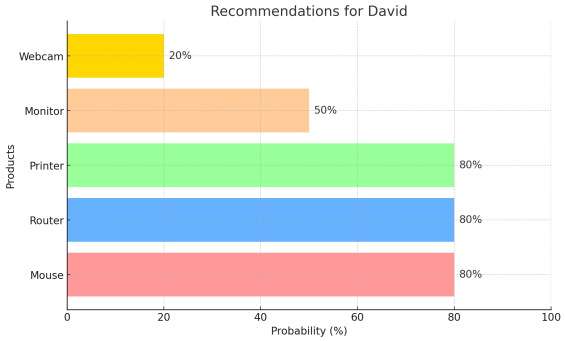

Figure 4: Recommendations for a particular user.

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.

Table 3: Product Recommendations for users.

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.

5.2 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 prob-

ability 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.

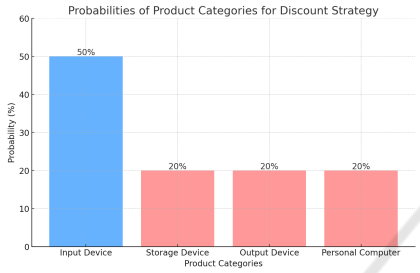

Figure 5: Recommendations for product categories.

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.

5.3 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:

- Tablet and Headset: The 'also_buy' relationship indicates that customers who purchase a Tablet often also buy a Headset. This pair is highly likely to benefit from a bundle discount.
- Notebook and Router: Both 'also_view' and 'also_buy' relationships suggest that customers who are interested in a Notebook are also likely to want a Router, making this pair a strong candidate for bundling.
- SSD and Router: Data shows that customers who purchase a Router also frequently buy SSDs, making this pair another potential bundle option.
- Keyboard and Mouse: These products are linked by both brand (Logitech) and category ('Input Device'). While the probability of purchasing one after the other may be medium to low, bundling them could increase this likelihood.

The following figure 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.

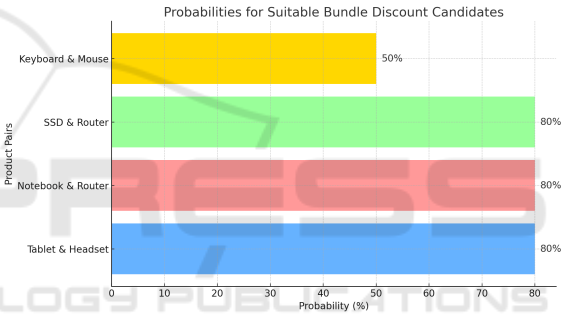

Figure 6: Recommendations for bundles.

6 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 (Zhao et al., 2023) and (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) and (Zhu et al., 2023), have explored integrating KGs with LLMs to improve the quality of responses in various tasks, including question answering, entity extraction, and knowledge graph reason-

ing. 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, harnessing their capabilities across various domains, 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. The input data refers to the actual data fed into the model for processing, like a question, an image or a set of data points. And the output indicator tells the model how to format its response and what type of output is expected.

7 CONCLUSIONS

This article has elucidated the critical role of techniques such as Prompt Engineering, Retrieval-Augmented Generation (RAG) and text-to-SQL in enhancing the functionality and applicability of LLMs in accessing and integrating external data sources. The Python scripts utilized in our analyses are openly accessible at (Seabra, 2024). These methodologies are fundamental in interacting with and leverage vast, dynamic external knowledge repositories, without the need of retraining a model.

Notably, these approaches have been applied with notable success to various data-intensive environments, including documents, knowledge graphs, and databases. By enabling LLMs to dynamically query and retrieve relevant information from these structured and unstructured data sources, the techniques enhance the model's ability to generate informed and contextually accurate outputs. This synergy not only maximizes the utility of existing data but also expands the potential applications of LLMs across different sectors, including business intelligence, legal advisement, and academic research.

The promising results obtained from these techniques underscore the potential for data interaction and retrieval. However, to fully ascertain their effectiveness and scalability, future work should focus on testing these methodologies across more voluminous and diverse data sets, encompassing extensive documents, knowledge graphs (KGs), and expansive databases. Such rigorous testing is essential to validate the robustness and adaptability of the strategies employed, ensuring that they maintain high levels of accuracy and efficiency when scaled.

Moreover, exploring these techniques in larger, more complex data environments will also shed light on their limitations and the potential need for refinement or adaptation. As mentioned in the paper, the continuous expansion of token limits in LLMs marks a significant trend in the evolution of artificial intelligence technologies. As these limits grow, developers are empowered to work with increasingly larger blocks of text in a single submission, enabling a deeper and more comprehensive analysis of data. This future exploration will not only bolster the confidence in deploying these techniques in real-world scenarios but also pave the way for their optimization and potential customization to specific domains or data types, ultimately enhancing the utility and impact of LLMs across various sectors.

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.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12:331–370.

Cao, Y., Wang, X., He, X., Hu, Z., and Chua, T.-S. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161.

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

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

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.

OpenAI (2023a). Chatgpt fine-tune description. https://help.openai.com/en/articles/6783457-what-is-chatgpt. Accessed: 2024-03-01.

OpenAI (2023b). Chatgpt prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering. Accessed: 2024-04-01.

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.

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.

Ricci, F., Rokach, L., and Shapira, B. (2010). Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer.

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. (2024). Github repository. https://github.com/antonyseabramedeiros/qasystems. Accessed: 2024-04-01.

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.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009(1):421425.

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, H., Zhang, F., Xie, X., and Guo, M. (2018). Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844.

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, P., Shi, T., and Reddy, C. K. (2020). Text-to-sql generation for question answering on electronic medical records.

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.

Zhao, Z., Fan, W., Li, J., Liu, Y., Mei, X., Wang, Y., Wen, Z., Wang, F., Zhao, X., Tang, J., et al. (2023). Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.

Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H., and Zhang, N. (2023). Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *arXiv preprint arXiv:2305.13168*.