BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 05/18 pp407–424
DOI: 10.26599/BDMA.2025.9020080
Volume 9, Number 2, April 2026

ItemRAG: Retrieval-Augmented Generation with Item-Based Knowledge Computing for E-Commerce Product Question Answering

Changliang Xu, Yukun Kang, Quan Feng, Jinghua Hua, Piji Li, Feiran Wu, Hu Wei, Xiang Chen*, Sheng-Jun Huang, and Songcan Chen

Abstract: The integration of Large Language Models (LLMs) into e-commerce platforms has significantly enhanced user experience through personalized recommendations and automated customer support. However, existing Retrieval-Augmented Generation (RAG) frameworks face challenges when applied to e-commerce product Question Answering (QA), such as handling extensive product catalogs, ensuring timely knowledge updates, and maintaining efficient retrieval performance. In this paper, we propose ItemRAG, a novel framework that combines RAG with item-based knowledge computing to address these challenges. ItemRAG decouples QA templates from specific products by leveraging a dynamic knowledge graph, enabling efficient updates and reducing the size of the knowledge base. The framework includes state analysis to capture user intent and context, grouped indexing for efficient retrieval, and knowledge computing to dynamically generate accurate answers. Experimental results demonstrate that decoupled-based ItemRAG significantly outperforms the Coupled-based RAG approaches (CoupledRAG) in retrieval accuracy and generation quality, achieving higher precision, recall, F1-score, and factual correctness. Our work highlights the efficacy of integrating the knowledge graph with RAG to enhance LLM-based e-commerce customer service systems.

Key words: retrieval augmentation; knowledge computing; e-commerce customer service; Large Language Models (LLMs)

1 Introduction

The advent of Large Language Models (LLMs) has revolutionized various domains by enabling advanced natural language understanding and generation capabilities. In the e-commerce sector, LLMs have been increasingly applied to enhance the online shopping experience. Applications range from personalized product recommendations to automated content creation and intelligent customer support. For

• Changliang Xu and Yukun Kang are with Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, Hangzhou 310024, China. E-mail: xuchangliang@ucas.ac.cn; kangyukun23@mails.ucas.ac.cn.
• Quan Feng is with Hunan Vanguard Group Corporation Limited, Changsha 410137, China. E-mail: jgxyfq@126.com.
• Jinghua Hua is with Muyu Works Ltd., Hangzhou 310024, China. E-mail: jinghua@ppt.video.
• Piji Li, Xiang Chen, Sheng-Jun Huang, and Songcan Chen are with College of Computer Science and Technology and MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China. E-mail: pjli@nuaa.edu.cn; xiang_chen@nuaa.edu.cn; huangsj@nuaa.edu.cn; s.chen@nuaa.edu.cn.
• Feiran Wu and Hu Wei are with Alibaba Group, Hangzhou 311121, China. E-mail: feiran.wufr@alibaba-inc.com; kongwang@alibaba-inc.com.
* To whom correspondence should be addressed.
Manuscript received: 2025-01-17; revised: 2025-05-23; accepted: 2025-06-27

example, RecMind[1] utilizes LLMs to develop autonomous recommender agents that interpret user behavior for context-aware recommendations, improving user engagement and conversion rates. Similarly, LLaMA-E[2, 3] and EcomGPT[4] fine-tune LLMs for e-commerce-specific tasks like generating product descriptions, crafting advertisements, answering general product-related queries, streamlining content creation, and improving marketing efficiency.

Despite these advancements, the application of LLMs for e-commerce customer service[5], particularly in product Question Answering (QA), remains underexplored. Existing research primarily focuses on recommendation systems and content generation, leaving a gap in leveraging LLMs to enhance customer support interactions. Effective customer service is crucial for user satisfaction and retention in e-commerce platforms, necessitating solutions that can provide accurate and context-aware responses to user queries.

Retrieval-Augmented Generation (RAG)[6–10] has emerged as a promising approach to enhance the factual accuracy of LLM-generated responses by incorporating external knowledge sources[11–18]. Consequently, it has made large-scale, automated, and highly accurate e-commerce QA systems feasible. Traditionally, e-commerce systems respond to user queries by retrieving and displaying a set of QA pairs, without generating new content. Moreover, e-commerce platforms typically construct QA pairs by coupling product information with predefined QA templates (to guide responses and ensure guideline compliance).

With the advancement of RAG, a natural evolution of this approach is to restructure these QA pairs into a knowledge base that serves as the retrieval source. Leveraging the text comprehension and generation capabilities of LLMs, this framework enables the system to precisely extract highly relevant QA pairs and generate responses in a more fluent manner, enhancing user experience and engagement. However, when applied to e-commerce QA, RAG approaches that construct QA pairs through coupling mechanisms (referred to as CoupledRAG) encounters significant limitations:

(1) Challenges in managing the size of the knowledge base. As depicted in Fig. 1, the coupling process between n QA templates and m products results in $n \times m$ QA pairs, reflecting a Cartesian product relationship. Unfortunately, e-commerce platforms often manage extensive product catalogs, inevitably resulting in an explosive number of potential QA pairs. This not only consumes considerable storage resources but also degrades retrieval efficiency and accuracy.

(2) Challenges in updating the knowledge base. Product attributes in e-commerce are frequently updated due to changes in pricing, availability, promotions, and other factors. In CoupledRAG, any modification to a product attribute necessitates updating all its associated QA pairs, making the knowledge base maintenance labor-intensive and inefficient.

(3) Constraints in embedding model performance. To accurately match user queries with relevant QA pairs, product identifiers (e.g., product IDs) are often included in the queries. However, these identifiers

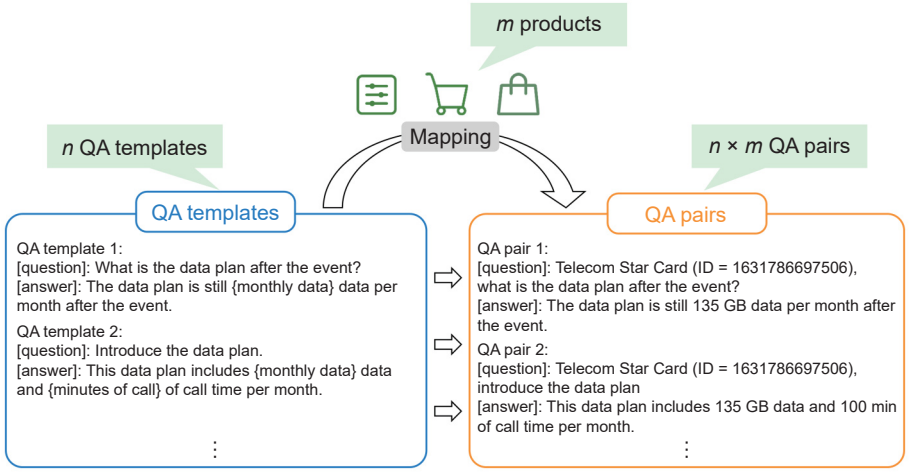


Fig. 1 Coupling process of QA pairs for CoupledRAG.

typically consist of non-semantic alphanumeric strings, which can impair the performance of embedding models that rely on semantic similarity, leading to decreased retrieval accuracy.

To address these challenges, we propose ItemRAG, a dynamic e-commerce QA framework that integrates RAG with item-based knowledge computing. Knowledge computing is a core module of ItemRAG. It involves dynamically retrieving product knowledge, selecting appropriate QA templates, and composing complete QA pairs to address user queries. The key innovation of ItemRAG is the decoupling of QA templates from specific products by leveraging a graph representation of product data. This decoupled architecture offers several advantages:

(1) Efficient knowledge base management. By storing QA templates and product information separately, ItemRAG reduces the storage complexity from $O(n \times m)$ in CoupledRAG to $O(n+m)$, thus improving storage efficiency and allowing product attributes to be updated without affecting the QA templates.

(2) Enhanced retrieval mechanism. The framework employs a grouped indexing strategy for the vector store, allowing efficient retrieval of relevant QA templates based on product categories and user intent.

(3) Dynamic knowledge computing. ItemRAG generates responses by populating QA templates with real-time product information obtained from the knowledge graph, ensuring that answers are accurate and up-to-date.

(4) Improved answer generation. By decoupling QA templates and product information, ItemRAG mitigates the issues caused by non-semantic product identifiers and leverages semantic-rich embeddings for better retrieval performance, thereby further enhancing the quality of generated responses.

Our contributions can be summarized as follows:

- We present a decoupled architecture that separates QA templates from product information, enabling efficient knowledge base updates and reducing storage requirements.
- We develop a state analysis and knowledge computing mechanism that dynamically generates accurate and context-aware answers to user queries.
- We conduct extensive experiments to evaluate the performance of ItemRAG against CoupledRAG approaches, demonstrating significant improvements in retrieval accuracy and generation quality.

The remainder of this paper is organized as follows: In Section 2, we review related work on LLM applications in e-commerce and RAG-based customer service systems. Section 3 discusses the limitations of CoupledRAG frameworks in the context of e-commerce QA. In Section 4, we detail the ItemRAG methodology, including the construction of the knowledge base, retrieval process, and answer generation techniques. Section 5 presents experimental results comparing ItemRAG with baseline methods and analyzes the performance gains. Finally, Section 6 concludes the paper and suggests directions for future research.

2 Related Work

2.1 LLMs for e-commerce application

LLMs are increasingly applied in the e-commerce domain to enhance various aspects of the online shopping experience. RecMind[1] leverages LLMs as an autonomous recommender agent by augmenting their ability to understand and interpret user behaviors. This approach enables more personalized and context-aware recommendations, improving the overall user experience and potentially driving higher conversions. Another notable work is LLaMA-E[2], which is built upon the LLaMA[3] model and fine-tuned specifically for e-commerce authoring tasks. These tasks include generating advertisements, product descriptions, and providing general product-related QA. By utilizing LLMs for content creation and customer support, LLaMA-E demonstrates the potential of language models to streamline and automate various aspects of e-commerce operations. EcomGPT[4] is a pioneering work in the development of instruction-tuned LLMs for e-commerce applications. It introduces the EcomInstruct dataset, which is used to fine-tune the model for a wide range of e-commerce tasks. Retail-GPT[5] is an open-source RAG-based chatbot for e-commerce product recommendations and shopping cart assistance. Despite these advancements, there remains a significant gap in the literature regarding the application of LLMs for e-commerce customer service. Existing works primarily focus on recommendation systems[19–24] and content generation, while the potential of LLMs to enhance customer support and provide personalized assistance remains largely unexplored.

2.2 RAG for e-commerce service

RAG[25] is a pattern for ensuring factual generation of responses in LLMs, aiming to avoid common pitfalls such as hallucinations and lack of provenance. With the advent of RAG and LLMs in recent periods, multiple studies also focus on utilizing LLMs for e-commerce customer assistance. BSharedRAG[6] trains a domain-specific backbone model and two LoRA modules to minimize retrieval and generation losses, improving performance compared to separate RAG frameworks. Herrero-Vidal et al.[7] proposed a new entity resolution type for variant product relationships, learning from e-commerce website structures. Veturi et al.[8] introduced an end-to-end framework using RAG-enabled LLMs to generate response suggestions for customer service agents, evaluating various retrieval, embedding, and prompting strategies. REAPER[9] is a retrieval plan generator for complex queries that minimizes latency while maintaining response quality compared to single or multi-agent systems. Moreover, SCRABLE[10] is an adaptive automation system for customer review that uses RAG and LLMs with self-optimizing prompts and an LLM-based judging mechanism. Xu et al.[11] built a Knowledge Graph (KG) from historical issues to preserve internal structure and relationships, parsing consumer queries and retrieving relevant subgraphs to generate answers. Motivated by prior research, we propose integrating RAG with item-based knowledge computing to effectively retrieve relevant information. By leveraging LLMs, we aim to generate precise, context-aware responses to customer queries, ultimately improving user experience and satisfaction.

3 Preliminary of Product QA System with RAG-Based Vector Retrieval

3.1 CoupledRAG pipeline

We compare ItemRAG with the baseline CoupledRAG. The workflow of CoupledRAG in the e-commerce QA setting is depicted in Fig. 2a.

Preparation stage. Based on conversations between users and customer service, common questions and answers are summarized to form QA templates. Product information is further added to these QA templates to create QA pairs.

Indexing stage. Instead of executing a document chunking, CoupledRAG uses an embedding model directly to convert the QA pairs to vectors which are then maintained in a vector store.

Retrieval stage. CoupledRAG converts the user query to a vector, which is used to retrieve the most relevant QA pairs in the vector store.

Generation stage. QA pairs are first filtered and reranked by CoupledRAG, then combined with the user query to form a prompt, which is submitted to an LLM to generate the final response.

3.2 Limitations of CoupledRAG

Although the CoupledRAG approach has certain advantages in efficient retrieval and improving generation, it also exhibits some limitations in e-commerce QA.

(1) Excessive knowledge base size. CoupledRAG combines QA templates with product information to form QA pairs, with each QA pair associated with a specific product. However, the number of products is typically in the tens of thousands, leading to a rapid expansion of the vector store size. An excessively large vector store inevitably affects the accuracy and speed of vector retrieval, which is unacceptable for e-commerce scenarios that require timely and accurate responses.

(2) Difficult knowledge base updates. In CoupledRAG, a product is associated with multiple QA pairs. When modifying a product attribute, it is necessary to synchronously update all its associated QA pairs, which is highly inefficient. In the context of e-commerce, adjusting product attributes (such as price) based on sales strategies is commonplace. However, the CoupledRAG framework struggles to achieve atomic updates of the knowledge base, which can affect normal product sales.

(3) Limited embedding model performance. To match the QA pairs in the knowledge base during retrieval, information that uniquely identifies the product, such as the product ID, needs to be added to the user query. Product IDs consist of many similar letters and numbers, a structure that lacks semantic information, which can negatively impact embedding models that rely on semantic understanding.

The fundamental reason for the limitations of CoupledRAG lies in the tight coupling between QA templates and specific products. To overcome these issues, we propose an e-commerce QA framework that combines the knowledge graph and RAG based on the concept of decoupling.

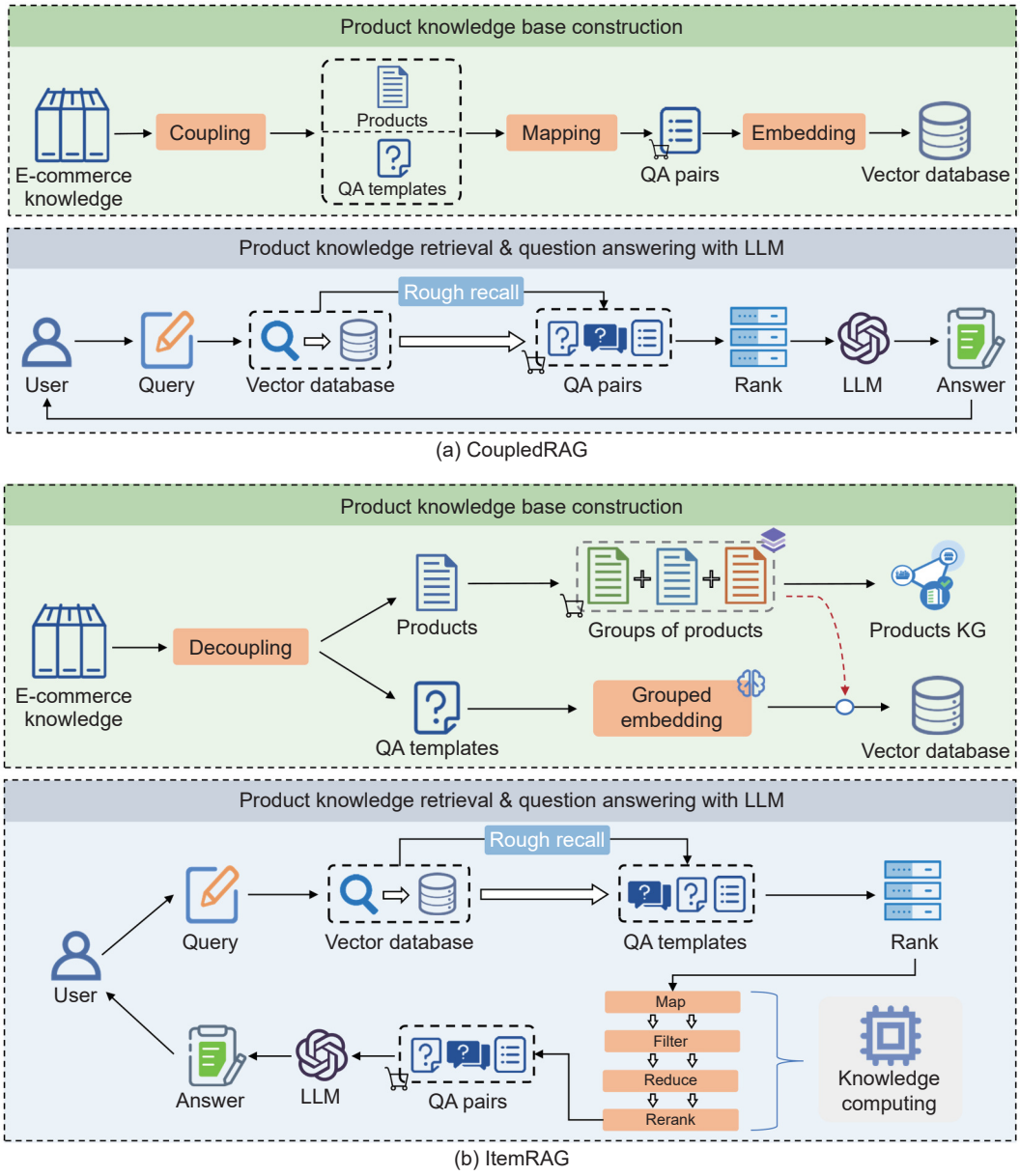


Fig. 2 CoupledRAG pipeline vs. ItemRAG pipeline.

4 Method

In this section, we present ItemRAG, a dynamic e-commerce QA framework designed to efficiently retrieve and process product information. Figure 2b illustrates the workflow of ItemRAG, which consists of three stages: knowledge base construction, product knowledge retrieval, and LLM-powered question answering.

4.1 Knowledge base construction

One of the main advantages of ItemRAG lies in its ability to decouple QA templates from specific products. This is achieved by storing QA templates in a vector store and product data in a knowledge graph.

4.1.1 Vector store construction

The construction of the vector store consists of two parts: knowledge inheritance and grouped indexing. Knowledge inheritance is a conceptual approach that connects QA templates to product categories, while grouped indexing builds the vector store for QA templates based on the corresponding categories.

Knowledge inheritance. Given a set of products $P = \{p_1, p_2, \ldots, p_m\}$, we group them into broader categories $C = \{C_1, C_2, \ldots, C_m\}$, where each $C_i = \{c_{i1}, c_{i2}, \ldots, c_{ij}\}$ is a sequence of j hierarchical categories

assigned to product p_i. In our dataset, each product is represented by a chain of four hierarchical classifications, ranging from the most specific to the most general (i.e., $j = 4$). For example (see Fig. 3), "Telecom Star Card (ItemID=163178...)" is first classified as "Telecom Star Card", then abstracted to "Telecom Card", and ultimately assigned to the top level "Telephone Card".

Grouped indexing. Suppose we have a set of QA templates $T = \{t_1, t_2, \ldots, t_n\}$ and their corresponding metadata $M = \{m_1, m_2, \ldots, m_n\}$, where each t_i is paired with m_i. We implement grouped indexing during the vector store construction,

$$V = \text{embed}(T, M) \tag{1}$$

where V denotes the vector store. The embed function transforms each QA template t_i into a vector while enriching it with its metadata m_i. Metadata serves to specify the scope and applicability of a QA template, enabling more flexible and targeted utilization. For example, consider a QA template with the question: "Please introduce the Telecom Star Card". This represents a general inquiry pertaining to the "Telecom Star Card" category, and thus its metadata should be labeled as "Telecom Star Card". In contrast, if the QA template question is: "What differentiates this product (Telecom Star Card, ItemID=163178...) from other items within the Telecom Star Card category?", it reflects a product-specific query. Accordingly, the metadata should be defined as "Telecom Star Card ItemID=163178...", capturing its focus on a distinct item within the broader category.

It is important to note that the metadata was derived through collaboration with multiple e-commerce experts, who manually curated it based on domain knowledge. In addition, it can be extended as needed to capture additional contextual dimensions, such as purchase stage, age group, geographic region, and other task-relevant attributes.

The combination of knowledge inheritance and grouped indexing enables ItemRAG to flexibly leverage QA templates to address user queries at both macro and micro levels. Moreover, this methodology not only suits our telephone card scenario but is universally generalizable across all e-commerce products.

4.1.2 Product knowledge graph construction

Graph structure definition. To construct the product knowledge graph, we use the Resource Description Framework (RDF), which represents data in the form of triples (subject, predicate, object). Each element in the triple serves a specific role as follows:

- **Subject:** It represents a specific product or a collection of products, identified by a unique ID.
- **Predicate:** It denotes the relationship between the subject and the object, such as "hasElement" or "hasCategory".
- **Object:** It represents the value of the attribute or another resource, identified by a literal value.

Graph construction. Based on the graph structure, we construct a product knowledge graph G with the product set P and the category set C,

$$G = \text{createRDF}(P, C) \tag{2}$$

where createRDF $(\cdot)$ function serves as a concise abstraction that defines the data required to construct a product graph, namely, a set of products and their associated categories. Each product comprises multiple attributes, such as ID, monthly data and minutes of call. Internally, createRDF $(\cdot)$ encodes a sequence of RDF triples instantiated from predefined patterns, which are systematically inserted into the RDF graph. As illustrated in Fig. 4, to capture the unique characteristics of the e-commerce domain, we design six well-defined triple pattern sets that guide the graph construction process:

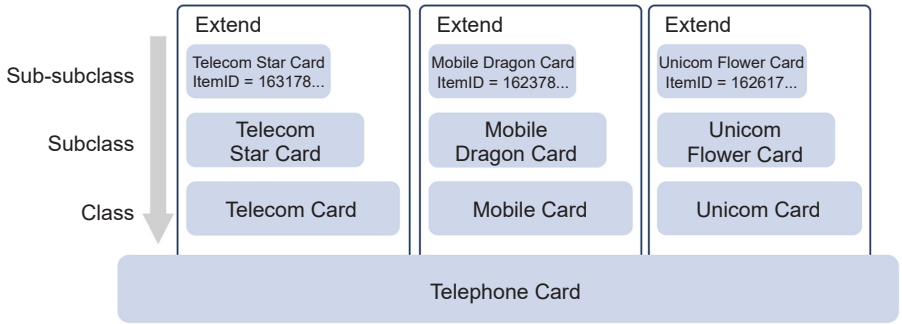


Fig. 3 Example of knowledge inheritance.

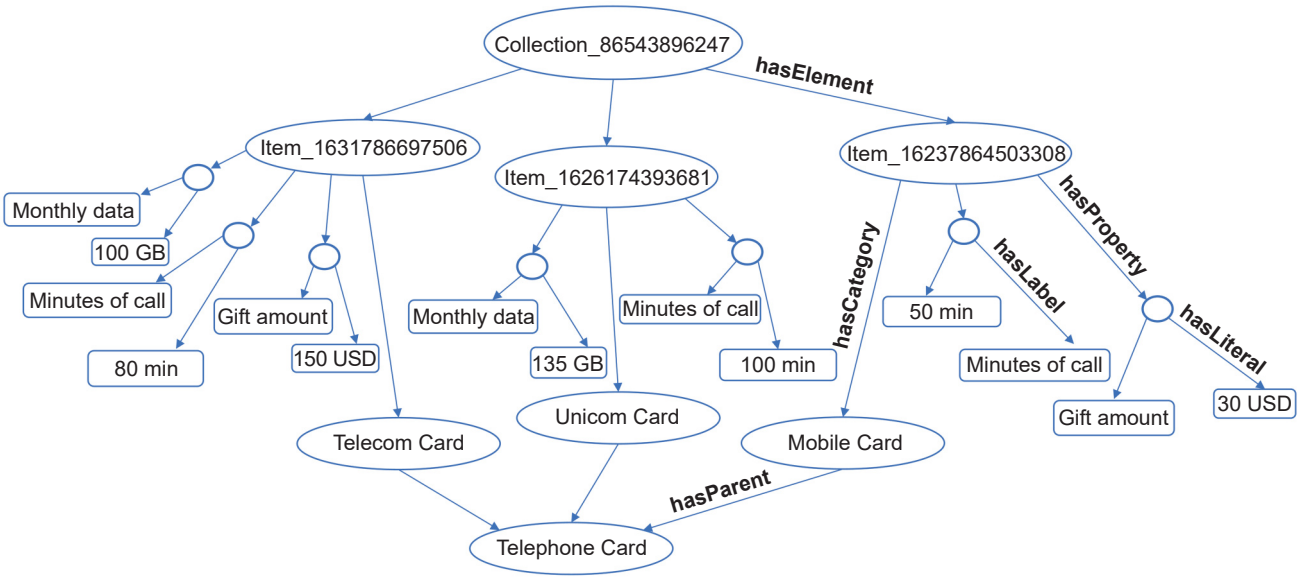


Fig. 4 Example of product graph.

- **Collection and Item:** The product detail page is treated as a Collection, and the products listed on the page are referred to as Items. Each Collection and Item is assigned a unique identifier. The relationship is represented by the RDF triple (Collection, hasElement, Item), where hasElement is a predicate indicating that the Collection contains the Item.
- **Item and BlankNode:** Each item has various properties. To accommodate properties that require additional structure beyond simple literals, we use a BlankNode. The relationship is represented by the RDF triple (Item, hasProperty, BlankNode), where hasProperty is a predicate indicating that the Item has properties represented by the BlankNode.
- **BlankNode and Literal:** The BlankNode used to represent a property can be further described by a literal value. The relationship is represented by the RDF triple (BlankNode, hasLiteral, Literal), where hasLiteral is a predicate indicating that the property represented by the BlankNode has the literal value.
- **BlankNode and Label:** The BlankNode also has a label to provide descriptive information about the property. The relationship is represented by the RDF triple (BlankNode, hasLabel, Label), where hasLabel is a predicate indicating that the property represented by the BlankNode has the label.
- **Item and Category:** Each Item can be grouped into a category. The relationship is represented by the RDF triple (Item, hasCategory, Category), where hasCategory is a predicate indicating that the Item belongs to the specified category.
- **Subcategory and parent-category:** Categories have hierarchical relationships, where one category is a subcategory of another. The relationship is represented by the RDF triple (CategoryA, hasParent, CategoryB), where hasParent is a predicate indicating that CategoryA is a subcategory of CategoryB.

4.2 Product knowledge retrieval

The retrieval process proceeds as follows: Based on the user query, ItemRAG retrieves relevant QA templates. These QA templates are then converted into RDF graphs and integrated with the product graph to extract the attributes of the user's browsing product. Subsequently, the attributes are populated into the QA templates to form QA pairs. Finally, these QA pairs are filtered, reduced, and reranked to generate the retrieval results.

To support personalized and context-aware retrieval, ItemRAG extracts key information which comprises user's query Q, browsing Item I, and purchase stage S (pre-sale, in-sale, or after-sale).

4.2.1 Vector retrieval

ItemRAG queries the vector store V with the user's Item I and query Q to retrieve the most relevant QA templates,

$$T' = \text{coarseRecall}\,(V, Q, I) \tag{3}$$

Specifically, $T' = \{t_1, t_2, \ldots, t_k\} \subseteq T$ denotes the set of the top k most relevant QA templates. The coarseRecall $(\cdot)$ function calculates the vector similarity between a user's query and QA templates

within a specific range, returning the most relevant results. Suppose the user is browsing an Item, such as the "Telecom Star Card (ID=163178...)", which belongs to the "Telecom Star Card" category. In this case, the function will not retrieve QA templates related to other categories, such as "Mobile Card" or "Union Card".

In addition to retrieving QA templates specifically related to the "Telecom Star Card", ItemRAG further performs recall within a broader set of QA templates that span from "Telecom Star Card" to "Telecom Card" and "Telephone Card" (following the 4-tier categorization described in grouped indexing). This comprehensive approach is necessary because the user's query may map to a category-level QA template (e.g., "Please introduce the Telecom Star Card".) or require an product-level QA template (e.g., "What differentiates this product (Telecom Star Card, ItemID=163178...) from other items within the Telecom Star Card category?"). Since the query type cannot be determined a priori, coarseRecall $(\cdot)$ guarantees that every relevant QA template is considered.

4.2.2 Knowledge computing

Knowledge computing plays a central role in transforming QA templates into high-quality QA pairs. After retrieving QA templates, ItemRAG queries the product graph to extract relevant attributes and populate them into the QA templates. Then it filters out irrelevant QA pairs, merges those with identical intent, and reorders the consolidated results. These four steps correspond to the map, filter, reduce, and rerank.

(1) Step 1

Map: The map phase comprises three core tasks: translating retrieved QA templates into RDF graphs; integrating these RDF graphs with the product graph; querying the unified graph to extract product attributes and instantiate the QA templates.

Convert: As illustrated in Fig. 5, QA templates are converted into RDF graphs,

$$F = \text{convert}(T') \tag{4}$$

Specifically, $F = \{f_1, f_2, \ldots, f_k\}$, where f_i represents an RDF graph based on a QA template $t_i \in T'$. Each of them consists of four primary triples: (Template, hasQuestion, Question), (Template, hasAnswer, Answer), (Template, hasRelated, Category), and (Template, hasVariable, Variable). The hasQuestion specifies the question defined by the QA template, while hasAnswer denotes its corresponding answer. The hasVariable identifies the product attributes that need to be filled into the QA template. The hasRelated triple is derived directly from the QA template's metadata and encodes the specific product hierarchy level to which the QA template applies.

Integrate: Since each QA template is linked to a specific product or category, its RDF graph can be seamlessly integrated with the product graph,

$$K = \text{integrate}(G, F) \tag{5}$$

where K denotes the conjunctive graph used for product information retrieval. The integrate $(\cdot)$ function identifies shared entities between the product graph G and the QA template graphs F. It then establishes connections between these entities using RDF triple operations, resulting in a unified, merged graph. As illustrated in Fig. 6, retrieval is conducted within this graph according to the presented architecture.

Query: ItemRAG extracts product-relevant attributes from the conjunctive graph and fills the QA templates based on these values,

$$R = \text{query}(K, I) \tag{6}$$

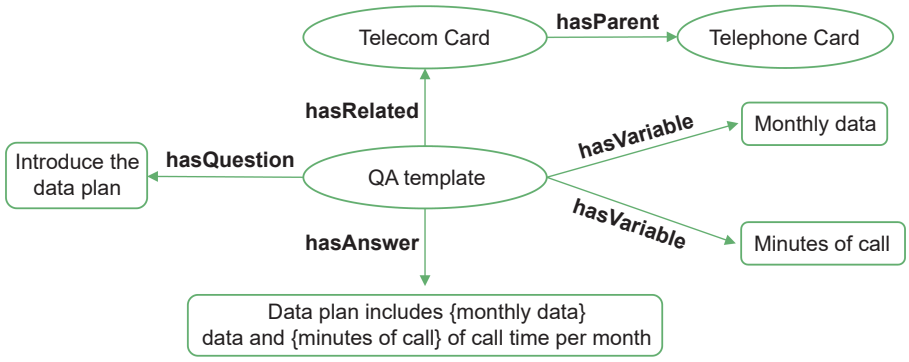


Fig. 5 RDF graph structure of QA template.

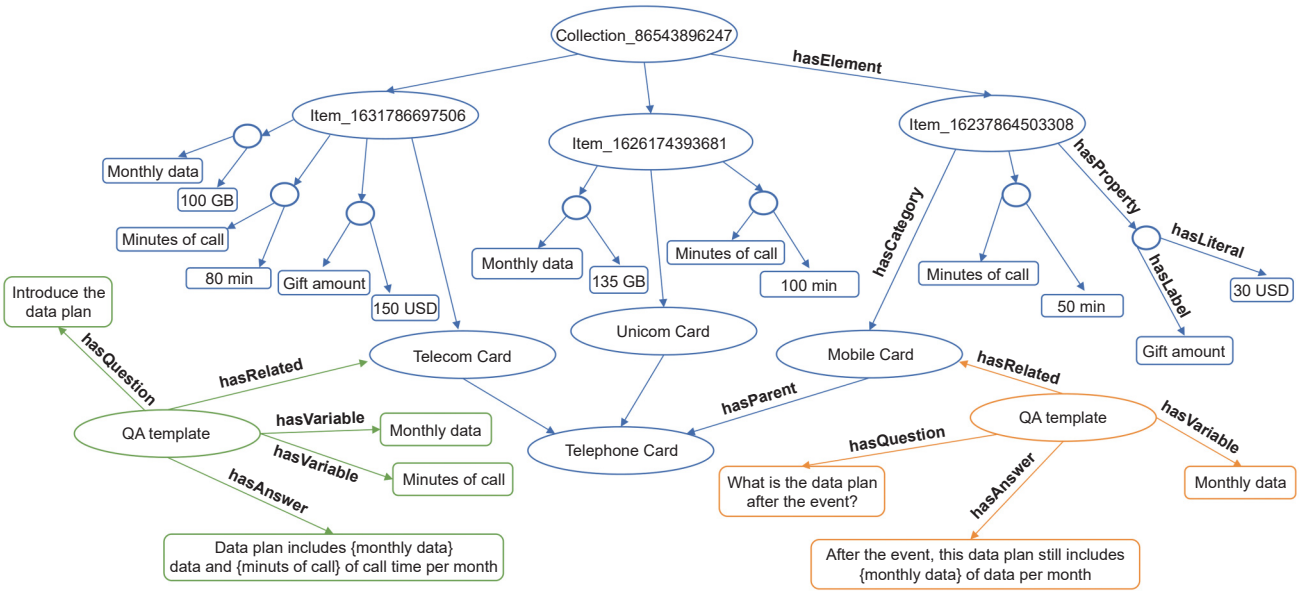


Fig. 6 Conjunctive graph involved in ItemRAG.

Using SPARQL Protocol and RDF query language, the query $(\cdot)$ function retrieves product attributes from the conjunctive graph K using the ID of item I, and populates them into the QA templates to form complete QA pairs $R = \{r_1, r_2, \ldots, r_k\}$. It is important to note that R may contain both successful and failed QA pairs. A failure occurs when the product lacks an attribute required by a QA template placeholder.

In most cases, extracting the user's browsing product and its attributes is sufficient to answer general questions. However, the carefully designed structure of our product graph enables deeper reasoning. For example, when a user asks "whether a cheaper alternative is available", ItemRAG can leverage the CollectionID to retrieve all Items within the same Collection and perform attribute comparisons, thereby supporting recommendation-style queries.

(2) Step 2

Filter: The filtering process primarily relies on predefined rules to refine the QA pairs generated during the map stage,

$$R' = \text{filter}\,(R) \tag{7}$$

Specifically, $R' \subseteq R$ denotes the filtered set of QA pairs. Two primary filtering mechanisms are employed:

- **Population filtering:** During the map stage, some QA templates may fail to be populated due to missing product attributes. These failed instances are explicitly marked and filtered out during the Filter stage.
- **Metadata filtering:** In addition to product categories, supplementary metadata, such as that produced in purchase stage (e.g., pre-sale, in-sale, and after-sale), can be assigned to QA templates during vector indexing. Although the retrieval process may return QA templates from multiple purchase stages, the user occupies only one specific status at any given time. Accordingly, QA templates are filtered based on the user's current purchase stage to maintain contextual relevance.

(3) Step 3

Reduce: At this stage, ItemRAG aggregates and ranks the remaining QA pairs to return the final retrieval results,

$$R'' = \text{rerank}\,(\text{reduce}\,(R')) \tag{8}$$

where $R'' \subseteq R'$ denotes the final retrieved content. The reduce $(\cdot)$ function aggregates QA pairs that share the same question but differ in their answers, and retains a single representative QA pair among them. Specifically, to meet the need for response diversity, as illustrated in Fig. 7, e-commerce platforms often prepare multiple answers for the same question. However, to reduce hallucinations in LLMs, we preserve only the QA pair with the highest vector similarity score.

(4) Step 4

Rerank: The preceding aggregation step may disrupt the original ordering of QA pairs. Therefore, after the reduce operation, we perform a lightweight rerank step to restore sequence coherence by sorting QA pairs according to their vector similarity scores.

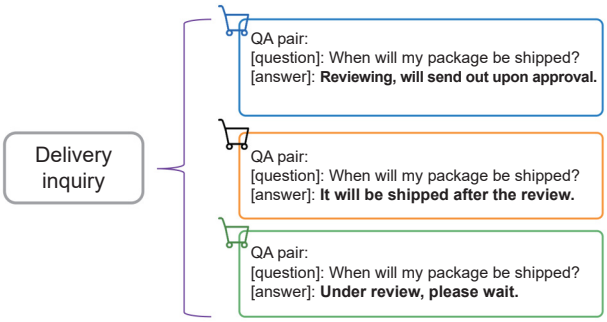


Fig. 7 QA pairs belonging to the same question.

4.3 LLM-powered question answering

After identifying the optimal embedding strategy, retrieval technique, and retrieval thresholds, we evaluate a selected prompting technique to ensure that LLMs generate grounded factual responses,

$$A = \text{generate}\,(L,\ Q,\ R'') \tag{9}$$

where the generate $(\cdot)$ function constructs a prompt by combining the user query Q with the retrieved content R'', and passes it to the LLM L to generate response A. For Chinese text generation across all tasks, we employ the Qwen2.5 series of LLMs, as they provide a clear path to production in terms of enterprise licenses, compared to other LLMs available at the time of our research.

5 Experiment

5.1 Setup

5.1.1 Baseline

We adopt CoupledRAG as our baseline. Constructing the CoupledRAG paradigm involves three key design dimensions: the mapping strategy for QA pairs, the degree of product information integration, and the overall architecture of the RAG pipeline.

(1) QA mapping strategies

The QA mapping strategy determines the cardinality of product associations for each QA pair, which directly impacts the scalability and efficiency of the system. Two primary strategies are explored in this work:

- **One-to-One:** Each QA pair corresponds to a single product, with the product explicitly specified in the question. This mapping strategy is straightforward and intuitive but suffers from a combinatorial explosion when scaling to a large number of products.
- **One-to-Many:** A single QA pair corresponds to all products, with the product explicitly specified in the answer. This approach extends the One-to-One mapping strategy and effectively mitigates the issue of combinatorial explosion. However, it results in significant information redundancy, which may strain the in-context learning capabilities of LLMs.

(2) Product information integration

The integration of product information dictates which attributes of each product are represented in QA pairs, thereby influencing the quality of the knowledge base and the downstream performance of the model. Specifically, we investigate two levels of product information integration:

- **With-ID:** QA pairs include both the product name and its ID. While the ID serves as a unique identifier, product IDs are often composed of alphanumeric strings that lack semantic meaning, potentially diminishing the performance of embedding models.
- **With-ID-Attr:** QA pairs incorporate the product name, ID, and additional attributes (e.g., monthly data allowance and promotional pricing). This approach combines the uniqueness of product IDs with semantically rich information, enhancing the expressiveness of the knowledge base.

(3) Overall RAG architecture

While the QA mapping strategy and product information integration focus on the indexing phase, the overall RAG architecture governs the orchestration of its retrieval and generation stages. In this study, we investigate two representative RAG methodologies:

- **NaiveRAG[25]:** This approach follows a "retrieve-then-generate" process, which is simple and efficient. However, it lacks an evaluation and reflection on the quality of both retrieval and generation, making it difficult to ensure the reliability of the final output.
- **Self-RAG[26]:** The rationale for selecting Self-RAG includes the following key points: First, it is a representative approach that introduces dynamic evaluation and self-reflection within the RAG framework. Second, Self-RAG primarily focuses on optimizing the retrieval and generation processes, with minimal constraints on the form of data indexing. This flexibility allows for the seamless integration of the same indexing format used in NaiveRAG, specifically the coupled QA pairs stored as vectors, without incurring significant modification costs.

5.1.2 Knowledge base

Figures 8 and 9 respectively illustrate the knowledge base constructions and associated user-query schemas

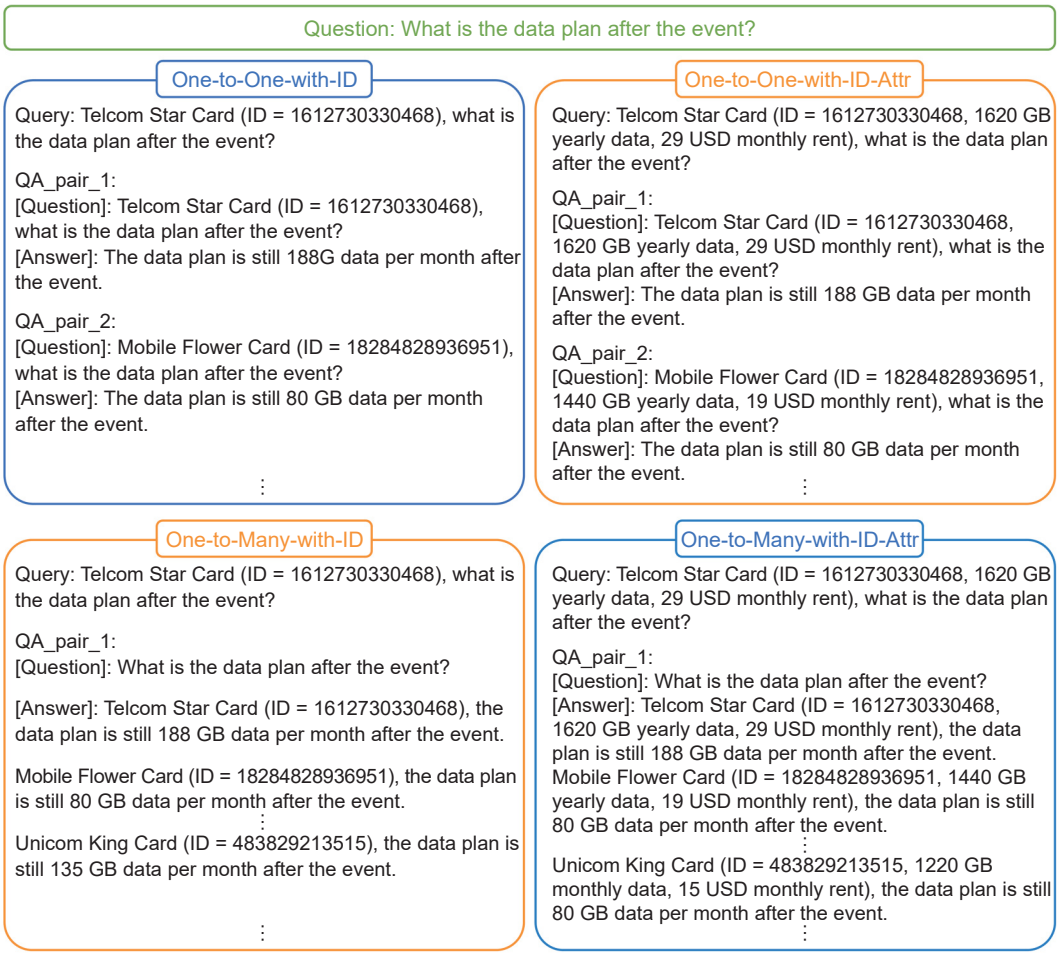


Fig. 8 Construction cases of CoupledRAG knowledge bases.

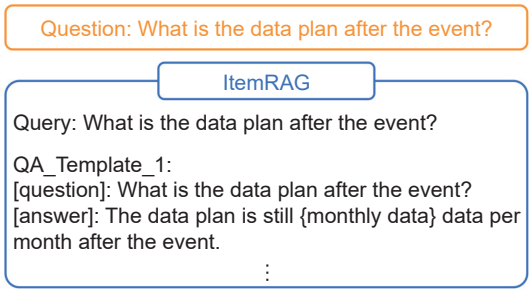


Fig. 9 Construction case of ItemRAG knowledge base.

of CoupledRAG and ItemRAG. We also analyze the size and average length of QAs described above (QA pairs for CoupledRAG and QA templates for ItemRAG). A summary of the results is presented in Table 1. Evidently, the ItemRAG configuration is the most compact, as evidenced by its minimal knowledge base footprint and the shortest average knowledge-unit length.

5.1.3 Datasets

While QA in domains like law and medicine has been extensively studied, the e-commerce domain presents unique challenges, leading to the specialized field of Product Question Answering (PQA)[27]. Several datasets have been developed for PQA research, each with its limitations: the Amazon dataset[28] consists of product reviews, QA pairs, and product information crawled from Amazon webpages. Despite its large size, it contains mostly user-generated content that is

Table 1 Comparison between different construction methods of knowledge base. Size denotes the number of QA pairs, and AvgLen indicates their average length. ItemRAG requires storing significantly fewer QA pairs, and those pairs are substantially shorter on average, than any of the competing methods.

Method	Size	AvgLen
One-to-One-with-ID	24 550	83.08
One-to-One-with-ID-Attr	24 550	99.90
One-to-Many-with-ID	599	3059.21
One-to-Many-with-ID-Attr	599	3748.35
ItemRAG	599	61.16

redundant, noisy, outdated, and lacks document similarity annotations, making it unsuitable for RAG tasks. SubjQA[29] explores the relationship between subjectivity in product reviews and PQA across six domains. However, its focus on subjective questions limits its diversity and coverage, rendering it inadequate for RAG tasks requiring broader background information. semiPQA[30] investigates PQA on semi-structured data, but its specific data structure requirements restrict its applicability to non-semi-structured data.

In summary, existing datasets suffer from various issues such as redundancy, noise, and data format constraints. To address these limitations, we have chosen to create our TeleCardQA dataset. We have established partnerships with multiple telephone card service providers and collected real conversations between their customer service representatives and users. These conversations serve as the data source for our knowledge base and user queries. We carefully selected 536 QA templates and randomly sampled 5 products from the collected merchandise for each QA template. By matching these QA templates with the sampled products, we ultimately constructed a test dataset comprising 2680 user queries. This approach ensures that our dataset is diverse, relevant, and representative of real-world user inquiries in the telephone card domain, making it well-suited for evaluating the performance of our proposed system.

5.1.4 Evaluation metrics

Retrieval ranking metrics. We employ two standard metrics to evaluate the retrieval performance of the E-Commerce Product QA system: Mean Reciprocal Rank (MRR) and hit rate.

(1) MRR measures the ability of the system to rank the most relevant document at the top of the retrieval list. Specifically, MRR is defined as the average of the reciprocal ranks of the first relevant QA pair across a set of queries. The reciprocal rank is the multiplicative inverse of the rank position of the first relevant QA pair. Formally, MRR is computed as follows:

$$\mathrm{MRR} = \frac{1}{|S|}\sum_{i=1}^{|S|}\frac{1}{\mathrm{rank}_i} \tag{10}$$

where rank_i refers to the rank position of the first relevant QA pair for the i-th query, and S denotes the total number of queries in the test set.

(2) Hit rate (Hit@K), on the other hand, evaluates the system's ability to retrieve relevant QA pairs within the top K results of the ranked list. It measures the proportion of queries for which at least one relevant QA pair appears among the top K retrieved results. Mathematically, Hit@K is defined as

$$\mathrm{Hit@}K = \frac{1}{|S|}\sum_{i=1}^{|S|} I(\mathrm{rank}_i \leqslant K) \tag{11}$$

where $I(\cdot)$ is the indicator function, which returns 1 if the condition is true (i.e., $\mathrm{rank}_i \leqslant K$) and 0 otherwise.

Together, these two metrics provide a comprehensive evaluation of the system's retrieval effectiveness. MRR emphasizes the precision of the system by assessing its ability to rank the most relevant QA pair at the top, while Hit@K evaluates its recall-like ability to surface relevant QA pairs within a defined retrieval window. These complementary metrics enable a thorough assessment of both ranking quality and retrieval completeness for the E-Commerce Product QA system.

Answer generation metrics. To evaluate the similarity between the generated answers and the reference answers, we compute the number of overlapping words and subsequently calculate Precision, Recall, and F1-score. These metrics are used to measure the syntactic similarity between the generated and reference answers. However, in the context of e-commerce, relying solely on word overlap is insufficient to accurately assess the correctness of responses generated by LLMs. For example, consider the user query: "How much data does this plan include?" The reference answer is: "Hello, this plan includes 80 GB data per month". Two generated responses are "Greetings, with this plan, you get 80 GB data each month" and "This plan includes 10 GB data per month". While the second response achieves higher Precision, Recall, and F1-score due to greater lexical overlap, its key information is incorrect. This highlights the limitations of traditional word-overlap-based metrics in evaluating the factual correctness of generated answers.

To address this issue, we introduce a novel arbitration-based evaluation framework. Specifically, we leverage a third-party LLM, Llama3.1-70B, to assess whether the key information in the generated answer aligns with that in the reference answer. The model outputs "Y" if the key information is consistent and "N" otherwise. Based on this evaluation, we define a new metric, K_{acc}, which is calculated as the proportion of "Y" responses over the total number of

queries. This approach ensures a more accurate evaluation of the factual consistency of generated answers, which is crucial in the e-commerce domain.

5.1.5 Implementation details

During the retrieval phase, we employ three widely used embedding models: Bge-Large-Zh[31], Gte-Large-Zh[32], and Dmeta-Embedding-Zh[33] as the backbone for knowledge retrieval. NaiveRAG executes a single retrieval operation while Self-RAG extends this by assessing the relevance of retrieved content to the original query and filtering out irrelevant information, utilizing Qwen2.5-32B-Instruct-GPTQ-Int4 (referred to as Self-RAG(32B)) and Qwen2.5-72B-Instruct-GPTQ-Int4 (referred to as Self-RAG(72B)) for evaluation. ItemRAG performs a category-based coarse recall within the vector store, followed by querying the product graph to fetch items the user has viewed.

In the generation stage, we use Qwen2.5-32B-Instruct-GPTQ-Int4 (refered to as Qwen2.5-32B) and Qwen2.5-72B-Instruct-GPTQ-Int4 (refered to as Qwen2.5-72B) as foundational generative engines. Under NaiveRAG, the model synthesizes the final answer directly from retrieved knowledge. Self-RAG introduces a validation step: it evaluates the generated content for alignment with retrieved evidence and effectiveness in addressing the user's query, thereby deciding whether to trigger an additional retrieval cycle or to finalize its response. By contrast, ItemRAG constructs explicit QA pairs by combining product-specific knowledge with QA templates; these QA pairs are then subjected to filter, reduce, and rerank before yielding the conclusive answer.

To clarify any potential ambiguity, it is essential to note that Self-RAG is implemented in two configurations: one in which Qwen2.5-32B is used for both retrieval and generation, and another where Qwen2.5-72B is utilized for both stages.

5.2 Experimental results

5.2.1 Retrieval evaluation

In this section, we evaluate the retrieval performance of the proposed ItemRAG method in comparison to CoupledRAG. The results are detailed in Table 2. Specifically, we compare the performance of ItemRAG and CoupledRAG across various embedding models and select the BGE model—where ItemRAG and CoupledRAG achieve their best performance—for further analysis.

Superior performance of ItemRAG. ItemRAG significantly outperforms all baseline configurations. This indicates that ItemRAG is highly effective in retrieving the correct knowledge entries relevant to user queries. The superior performance can be attributed to ItemRAG's tailored approach in handling item-specific knowledge, which better aligns with the complexities of e-commerce knowledge bases where precise product information is crucial.

Performance of baseline configurations. Under the One-to-One mapping strategy, the inclusion of product attributes in the QA pairs (One-to-One-with-ID-Attr) provides additional semantic information, enabling more accurate retrieval compared to its counterpart

Table 2 Retrieval performance on different embedding models. The best results are highlighted in bold.

(%)

Method	Knowledge base	Bge-Large-Zh				Gte-Large-Zh				Dmeta-Embedding-Zh			
		MRR	Hit@1	Hit@3	Hit@5	MRR	Hit@1	Hit@3	Hit@5	MRR	Hit@1	Hit@3	Hit@5
NaiveRAG	One-to-One-with-ID	68.04	57.98	77.53	83.95	49.38	36.86	60.33	70.00	56.32	42.94	67.94	77.94
	One-to-One-with-ID-Attr	81.12	75.11	86.82	89.73	56.18	42.16	69.44	77.87	71.95	60.22	83.47	89.55
	One-to-Many-with-ID	80.90	75.22	86.67	89.36	58.85	55.59	62.46	63.24	63.05	55.59	70.44	73.95
	One-to-Many-with-ID-Attr	81.01	74.55	87.20	90.14	74.80	70.26	79.32	80.78	66.09	56.82	74.58	80.22
Self-RAG (32B)	One-to-One-with-ID	59.33	50.89	67.57	72.08	43.45	33.47	52.79	59.14	41.63	30.59	51.52	59.85
	One-to-One-with-ID-Attr	67.10	62.38	71.67	73.65	45.01	34.73	54.47	60.63	52.33	42.50	62.23	67.61
	One-to-Many-with-ID	77.24	71.56	83.02	84.81	72.41	65.59	79.44	81.97	54.76	47.16	62.27	65.67
	One-to-Many-with-ID-Attr	75.00	68.39	81.75	83.65	72.44	65.48	79.73	82.05	56.4	48.24	64.4	68.20
Self-RAG (72B)	One-to-One-with-ID	66.74	56.75	76.34	81.86	49.12	37.76	59.55	67.68	47.83	34.51	59.85	70.03
	One-to-One-with-ID-Attr	78.84	72.79	84.44	87.57	52.77	40.78	63.88	71.56	62.02	49.96	73.8	81.04
	One-to-Many-with-ID	76.42	71.11	81.67	83.65	70.63	63.54	77.87	80.37	53.84	46.45	61.00	64.36
	One-to-Many-with-ID-Attr	71.52	65.14	77.68	80.44	69.52	62.61	76.60	79.32	54.10	45.74	62.61	66.26
ItemRAG	Decoupling	**92.87**	**92.26**	**93.44**	**93.52**	**92.49**	**92.43**	**92.60**	**92.65**	**89.07**	**88.73**	**89.41**	**89.57**

with only product IDs (One-to-One-with-ID). In comparison, the benefits of One-to-Many-with-ID-Attr configuration are relatively limited. In some cases, it even underperforms compared to the One-to-Many-with-ID setup. We attribute this to the redundancy in One-to-Many QA pairs, which introduces additional complexity and poses challenges to the model performance. Moreover, incorporating additional attributes further extends the text length, yielding only minimal benefits and potentially even introducing negative effects.

This observation highlights the importance of incorporating semantic information in the form of product attributes to enhance the expressiveness of the knowledge base and improve retrieval performance, provided that the text does not become excessively lengthy to the point of impairing model performance.

In summary, the experimental results demonstrate the superior retrieval performance of ItemRAG compared to the CoupledRAG configurations. The analysis of the CoupledRAG reveals the importance of incorporating semantic information in the form of product attributes and highlights the effectiveness of the One-to-Many mapping strategy in mitigating the combinatorial explosion issue while maintaining retrieval performance.

5.2.2 Generation evaluation

We evaluate the answer generation performance using Qwen2.5-32B and Qwen2.5-72B as the base models for generating responses, while employing Llama3.1-70B as the arbitration model to assess the factual accuracy of the generated answers. The evaluation results are presented in Table 3.

Effectiveness across models. ItemRAG demonstrates exceptional performance in the larger LLM (Qwen2.5-72B), surpassing all baseline methods. When applied to the smaller-scale LLM (Qwen2.5-32B), the performance of ItemRAG remains largely unaffected, while baseline methods exhibit varying degrees of performance fluctuations. This highlights the strong generalization ability of ItemRAG across models of different scales.

Coupling vs. decoupling. Decoupled-based ItemRAG outperforms all coupled-based baseline configurations, showcasing the advantages of decoupling in knowledge base management and retrieval optimization, ultimately improving the quality of generated responses.

Meanwhile, there is a notable phenomenon that Self-RAG performs slightly inferior to NaivRAG. We observe that the dataset is derived from real-world scenarios, which include challenging user queries, such as “Why is the general data plan so limited?”. In response to such questions, e-commerce platforms typically provide tactful and ambiguous answers. However, Self-RAG iteratively evaluates whether the LLMs’ output adequately addresses the user’s question. Within the LLMs’ cognitive framework, vague responses are perceived as failing to provide a satisfactory answer, leading to repeated iterations and a decline in output quality. This observation further highlights the substantial gap between the cognitive processes of LLMs and human reasoning.

5.3 Analysis

5.3.1 Analysis of ItemRAG performance gain

Decoupling of QA templates and product information. ItemRAG decouples QA templates from

Table 3 Generation performance on different LLMs. ItemRAG consistently surpasses all baselines across every model scale, and its data are highlighted in bold.

(%)

Method	Knowledge base	Qwen2.5-32B				Qwen2.5-72B			
		Precision	Recall	F1-score	K_{acc}	Precision	Recall	F1-score	K_{acc}
NaiveRAG	One-to-One-with-ID	92.29	91.12	91.48	90.22	94.30	92.88	93.29	91.67
	One-to-One-with-ID-Attr	92.35	91.38	91.70	90.22	93.48	92.35	92.68	90.82
	One-to-Many-with-ID	79.11	87.43	82.47	83.76	79.14	87.94	82.86	84.40
	One-to-Many-with-ID-Attr	68.66	87.32	74.99	72.38	62.60	89.06	72.04	66.67
Self-RAG	One-to-One-with-ID	85.75	83.71	83.58	76.06	91.28	87.90	88.93	84.60
	One-to-One-with-ID-Attr	83.53	82.87	81.67	76.16	90.88	87.81	88.77	84.60
	One-to-Many-with-ID	84.21	81.24	81.25	67.17	74.55	77.02	74.28	68.88
	One-to-Many-with-ID-Attr	83.09	80.49	80.32	65.44	68.47	72.61	68.68	62.93
ItemRAG	Decoupling	**97.23**	**97.32**	**97.12**	**96.26**	**98.27**	**98.06**	**98.12**	**97.05**

specific products by dynamically computing product information to generate responses. In e-commerce, where there are vast numbers of products with diverse attributes, tightly coupled approaches like One-to-One and One-to-Many mappings introduce significant challenges. The One-to-One mapping strategy suffers from a combinatorial explosion of QA pairs, which negatively impacts vector-based retrieval efficiency. On the other hand, the One-to-Many mapping strategy leads to excessive redundancy in QA pairs, potentially exceeding the context window size of LLMs. Even when the context window is not exceeded, lengthy input text can cause LLMs to struggle with the "lost in the middle" phenomenon, where critical information is overlooked.

Precise information retrieval through product knowledge graph. ItemRAG achieves precise and reliable information retrieval by leveraging a product knowledge graph to query product-specific information. By organizing products into a knowledge graph, ItemRAG naturally represents One-to-Many relationships between products and their attributes through the graph's data structure. This design enables accurate question answering for general user queries, such as "Introduce this data plan". Furthermore, in the product knowledge graph, similar types of product nodes are linked to a common node, making it highly effective for product recommendation scenarios. For instance, when a user asks "Is there a cheaper data plan?", ItemRAG can query the knowledge graph to retrieve all products of the same type, enabling a direct comparison and yielding the optimal recommendation.

5.3.2 Comparison of inference time cost

We conduct a comparative analysis of the inference time costs for various retrieval-generation frameworks. Our findings reveal that ItemRAG exhibits shorter inference time compared to the CoupledRAG-based retrieval paradigms, with both leveraging vLLM[34] for acceleration. As shown in Fig. 10, despite the overhead incurred by querying the knowledge graph, ItemRAG maintains a relatively short average inference time, exceeding the performance of most methods when using Qwen2.5-32B and Qwen2.5-72B. The two One-to-Many configurations exhibit significantly higher time costs due to processing longer QA pairs, as corroborated by Table 1. Similarly, the iterative generation process of Self-RAG introduces a substantially higher time cost, which is unacceptable for e-commerce QA systems that prioritize user experience.

The comparative analysis highlights ItemRAG's efficiency advantages, achieved by leveraging knowledge graphs and optimizing the retrieval process. The competitive inference times underscore its potential for real-world e-commerce applications, where responsiveness and scalability are critical.

6 Conclusion

In this paper, we introduce ItemRAG, a novel framework that addresses the limitations of RAG approaches which adopt CoupledRAG in the context of e-commerce product QA. By decoupling QA templates from specific product information and leveraging a dynamic product knowledge graph, ItemRAG effectively manages the vast and ever-changing product catalogs typical of e-commerce platforms. Our approach tackles three primary challenges faced by CoupledRAG systems: scalability and efficiency,

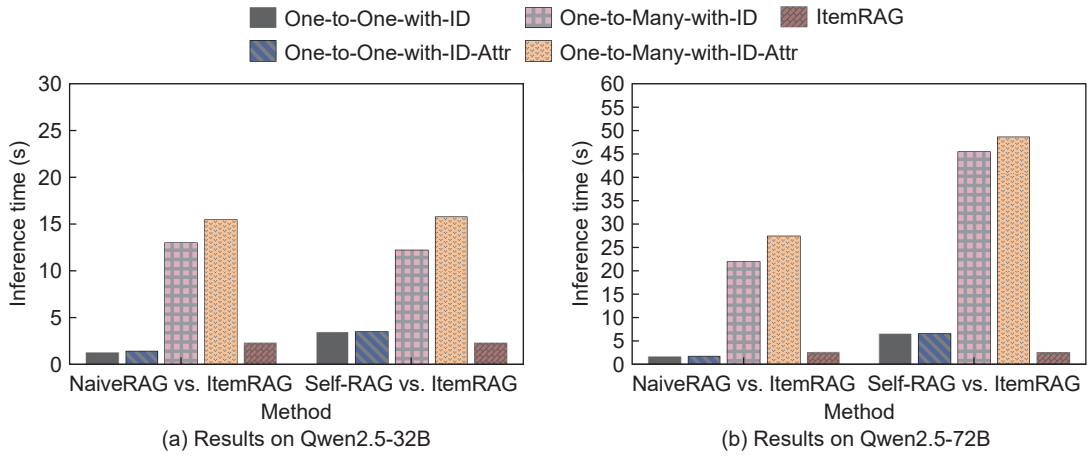


Fig. 10 Inference time comparison.

dynamic knowledge updates and improved embedding performance. We conduct extensive experiments on the TeleCardQA dataset and compare ItemRAG with various configurations of CoupledRAG that differ in QA mapping strategies, product information integration levels, and RAG architectures. The results demonstrate that ItemRAG outperforms the baseline methods across multiple metrics.

While ItemRAG addresses significant challenges in e-commerce QA systems, there are avenues for further research and enhancement: (1) Expanded knowledge graph integration: Incorporating additional data sources, such as customer reviews and real-time inventory status, could enrich the knowledge graph and enable more comprehensive responses; (2) Multilingual support: Extending ItemRAG to handle multiple languages would make the framework applicable to a broader range of e-commerce platforms operating in different regions; and (3) Adaptive learning mechanisms: Implementing feedback loops where the system learns from user interactions could improve the relevance and accuracy of responses over time.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 62506166, U2441285, and 62222605), the Natural Science Foundation of Jiangsu Province (No. SBK20250401456), the China Postdoctoral Science Foundation (No. 2025M774283), and the DiDi GAIA Collaborative Research Funds (No. CCF-DiDi GAIA202507).

References

[1] Y. Wang, Z. Jiang, Z. Chen, F. Yang, Y. Zhou, E. Cho, X. Fan, Y. Lu, X. Huang, and Y. Yang, RecMind: Large language model powered agent for recommendation, in *Proc. Findings of the Association for Computational Linguistics*, Mexico City, Mexico, 2024, pp. 4351–4364.

[2] K. Shi, X. Sun, D. Wang, Y. Fu, G. Xu, and Q. Li, LLaMA-E: Empowering E-commerce authoring with object-interleaved instruction following, in *Proc. 31st Int. Conf. Computational Linguistics*, Abu Dhabi, United Arab Emirates, 2025, pp. 870–885.

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., LLaMA: Open and efficient foundation language models, arXiv preprint arXiv: 2302.13971, 2023.

[4] Y. Li, S. Ma, X. Wang, S. Huang, C. Jiang, H. T. Zheng, P. Xie, F. Huang, and Y. Jiang, EcomGPT: Instruction-tuning large language models with chain-of-task tasks for E-commerce, in *Proc. 38th AAAI Conf. Artificial Intelligence*, Vancouver, Canada, 2024, pp. 18582–18590.

[5] B. A. T. de Freitas and R. de Alencar Lotufo, Retail-GPT: Leveraging retrieval augmented generation (RAG) for building E-commerce chat assistants, arXiv preprint arXiv: 2408.08925, 2024.

[6] K. Guan, Q. Cao, Y. Sun, X. Wang, and R. Song, BSharedRAG: Backbone shared retrieval-augmented generation for the E-commerce domain, in *Proc. Findings of the Association for Computational Linguistics*, Miami, FL, USA, 2024, pp. 1137–1158.

[7] P. Herrero-Vidal, Y. L. Chen, C. Liu, P. Sen, and L. Wang, Learning variant product relationship and variation attributes from E-commerce website structures, arXiv preprint arXiv: 2410.02779, 2024.

[8] S. Veturi, S. Vaichal, R. L. Jagadheesh, N. I. Tripto, and N. Yan, RAG based question-answering for contextual response prediction system, arXiv preprint arXiv: 2409.03708, 2024.

[9] A. Joshi, S. M. Sarwar, S. Varshney, S. Nag, S. Agrawal, and J. Naik, REAPER: Reasoning based retrieval planning for complex RAG systems, in *Proc. 33rd ACM Int. Conf. Information and Knowledge Management*, Boise, ID, USA, 2024, pp. 4621–4628.

[10] G. Azov, T. Pelc, A. F. Alon, and G. Kamhi, Self improving customer review response generation based on LLMs, in *Proc. 7th Workshop on e-Commerce and NLP @ LREC-COLING*, Torino, Italy, 2024, pp. 40–57.

[11] Z. Xu, M. J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang, and Z. Li, Retrieval-augmented generation with knowledge graphs for customer service question answering, in *Proc. 47th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Washington, DC, USA, 2024, pp. 2905–2909.

[12] K. Liang, L. Meng, M. Liu, Y. Liu, W. Tu, S. Wang, S. Zhou, X. Liu, F. Sun, and K. He, A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 9456–9478, 2024.

[13] K. Liang, L. Meng, H. Li, M. Liu, S. Wang, S. Zhou, X. Liu, and K. He, MGKsite: Multi-modal knowledge-driven site selection via intra and inter-modal graph fusion, *IEEE Trans. Multimedia*, vol. 27, pp. 1722–1735, 2025.

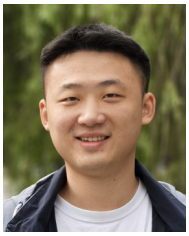
[14] K. Liang, Y. Liu, S. Zhou, W. Tu, Y. Wen, X. Yang, X. Dong, and X. Liu, Knowledge graph contrastive learning based on relation-symmetrical structure, *IEEE Trans. Knowledge Data Eng.*, vol. 36, no. 1, pp. 226–238, 2024.

[15] J. Li, Y. Lei, Y. Bian, D. Cheng, Z. Ding, and C. Jiang, RA-CFGPT: Chinese financial assistant with retrieval-augmented large language model, *Front. Comput. Sci.*, vol. 18, no. 5, p. 185350, 2024.

[16] X. Chen, N. Zhang, X. Xie, S. Deng, Y. Yao, C. Tan, F. Huang, L. Si, and H. Chen, KnowPrompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction, in *Proc. ACM Web Conf.*, Lyon, France, 2022, pp. 2778–2788.

[17] L. Cui, Y. Liu, C. Ouyang, Y. Yu, J. Zhang, Y. Wan, and F. Yang, Bailicai: A domain-optimized retrieval-augmented generation framework for medical applications, *Big Data Mining and Analytics*, doi: 10.26599/BDMA.2024.9020097.

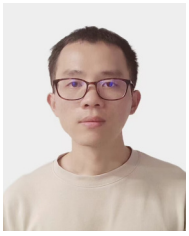
[18] Z. Hu, P. Yang, F. Liu, Y. Meng, and X. Liu, Prompting large language models with knowledge-injection for knowledge-based visual question answering, *Big Data Mining and Analytics*, vol. 7, no. 3, pp. 843–857, 2024.
[19] D. Zou, W. Wei, F. Zhu, C. Xu, T. Zhang, and C. Huo, Knowledge enhanced multi-intent transformer network for recommendation, in *Proc. ACM Web Conf.*, Singapore, 2024, pp. 1–9.
[20] S. Zhao, W. Wei, X. L. Mao, S. Zhu, M. Yang, Z. Wen, D. Chen, and F. Zhu, Multi-view hypergraph contrastive policy learning for conversational recommendation, in *Proc. 46th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Taipei, China, 2023, pp. 654–664.
[21] C. Wang, L. Yang, Z. Liu, X. Liu, M. Yang, Y. Liang, and P. S. Yu, Collaborative alignment for recommendation, in *Proc. 33rd ACM Int. Conf. Information and Knowledge Management*, Boise, ID, USA, 2024, pp. 2315–2325.
[22] Y. Zhang, Z. Liu, Q. Wen, L. Pang, W. Liu, and P. S. Yu, AI agent for information retrieval: Generating and ranking, in *Proc. 33rd ACM Int. Conf. Information and Knowledge Management*, Boise, ID, USA, 2024, pp. 5605–5607.
[23] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, Chat-REC: Towards interactive and explainable LLMs-augmented recommender system, arXiv preprint arXiv: 2303.14524, 2023.
[24] K. Taha, P. D. Yoo, C. Yeun, and A. Taha, Empirical and experimental perspectives on big data in recommendation systems: A comprehensive survey, *Big Data Mining and Analytics*, vol. 7, no. 3, pp. 964–1014, 2024.
[25] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, Retrieval-augmented generation for large language models: A survey, arXiv preprint arXiv: 2312.10997, 2023.
[26] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, Self-RAG: Learning to retrieve, generate, and critique through self-reflection, in *Proc. 12th Int. Conf. Learning Representations*, Vienna, Austria, https://dblp.uni-trier.de/rec/conf/iclr/AsaiWWSH24.html?view=bibtex, 2024.
[27] Y. Deng, W. Zhang, Q. Yu, and W. Lam, Product question answering in E-commerce: A survey, in *Proc. 61st Annu. Meeting of the Association for Computational Linguistics*, Toronto, Canada, 2023, pp. 11951–11964.
[28] J. McAuley and A. Yang, Addressing complex and subjective product-related queries with customer reviews, in *Proc. 25th Int. Conf. World Wide Web*, Montreal, Canada, 2016, pp. 625–635.
[29] J. Bjerva, N. Bhutani, B. Golshan, W. C. Tan, and I. Augenstein, SubjQA: A dataset for subjectivity and review comprehension, in *Proc. Conf. Empirical Methods in Natural Language Processing*, Virtual Event, 2020, pp. 5480–5494.
[30] X. Shen, G. Barlacchi, M. Del Tredici, W. Cheng, and A. Gispert, semiPQA: A study on product question answering over semi-structured data, in *Proc. 5th Workshop on e-Commerce and NLP*, Dublin, Ireland, 2022, pp. 111–120.
[31] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J. Y. Nie, C-Pack: Packed resources for general Chinese embeddings, in *Proc. 47th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Washington, DC, USA, 2024, pp. 641–649.
[32] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, Towards general text embeddings with multi-stage contrastive learning, arXiv preprint arXiv: 2308.03281, 2023.
[33] DMetaSoul/Dmeta-Embedding-Zh, https://huggingface.co/DMetaSoul/Dmeta-embedding-zh, 2024.
[34] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, Efficient memory management for large language model serving with PagedAttention, in *Proc. 29th Symp. Operating Systems Principles*, Koblenz, Germany, 2023, pp. 611–626.



Yukun Kang received the BEng degree in information security (cryptology) from Hainan University, China in 2022. He is currently a master student at Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences (UCAS). His main research interests include LLMs and AI agent.

Quan Feng received the BEng and MEng degrees from Nanchang Hangkong University, China in 2008 and 2011, respectively, and the PhD degree from Nanjing University of Aeronautics and Astronautics, China in 2022. Currently, he is the head of Machine Learning Research Laboratory, Hunan Vanguard Group Corporation Limited, China. His research interests are mainly in machine learning and pattern recognition.

Changliang Xu received the BEng degree from Peking University, China in 2003, and the PhD degree from Princeton University, USA in 2007. He is currently a professor at Hangzhou Institute for Advanced Study, UCAS, China, leading a research group dedicated to the development of AI agents leveraging Large Language Models (LLMs) and multimodal models. He played a pivotal role in the establishment of Alibaba Cloud, founding the team responsible for the development of Alibaba's Big Data Platform (ODPS) and the Machine Learning Platform (PAI). His current research interests are centered around the automation of data mining and content generation using AI agents.



Jinghua Hua received the MEng degree from Zhejiang University of Technology, China in 2013. He is currently the algorithm director at Muyu Works Ltd. His research interests are mainly in LLMs, knowledge graphs, and intelligent dialogue systems.

Feiran Wu received the PhD degree in computer science and technology from Zhejiang University, China in 2016. He is currently an algorithm expert at Alibaba Group, China, leading a research and development team with a primary focus on AI-powered business analytics. His team provides technical support for the business intelligence platform products within Alibaba, serving over 80 000 users. His research interests include interaction-enhanced analysis, visual analysis, and practical application of LLMs.

Xiang Chen received the PhD degree from Zhejiang University, China in 2024. He is currently a professor at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China, leading a research group focused on multimodal and language intelligence technologies. He has served as a program committee member for several prestigious conferences, including ICLR, NeurIPS, SIGIR, and WWW. His current research interests include LLMs and knowledge graphs.

Songcan Chen received the PhD degree from Nanjing University of Aeronautics and Astronautics, China in 1997, where he is currently a professor. He is the director of the Machine Learning Committee of the Chinese Artificial Intelligence Association, the executive director of the Chinese Artificial Intelligence Association, and the executive vice president of the Jiangsu Artificial Intelligence Association. He is also a fellow of the International Association for Pattern Recognition (IAPR) and a fellow of the Chinese Artificial Intelligence Association (CAAI). He has published more than 170 papers in international mainstream academic journals. According to Google Scholar statistics, his papers have been cited more than 20 000 times, with an H-index of 66. He has been selected into the Elsevier China Highly Cited Scholars List for 10 consecutive years from 2014 to 2024. His current research interests include machine learning and data mining.

Piji Li received the BEng and MEng degrees from Shandong University, China in 2009 and 2012, respectively, and the PhD degree from the Chinese University of Hong Kong, China in 2018. He is currently a professor at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China, and leads the Nanjing University of Aeronautics and Astronautics NLP Group. Previously, he was a senior researcher at Tencent AI Lab. His research interests are mainly in LLMs and dialogue systems.

Hu Wei received the MEng degree from Wuhan University, China in 2003. He is currently a PhD candidate in computer science and technology at Zhejiang University, China. He is also a technical director at Alibaba Group, where his team is responsible for building and maintaining the Alibaba Data Infrastructure Platform. This platform focuses on the collection, development, management, and utilization of Alibaba's big data, providing technical support for data-driven business scenarios. His research interests include big data mining and machine learning, big data analytics and visualization, as well as data management and data warehousing.

Sheng-Jun Huang received the BEng and PhD degrees in computer science from Nanjing University, China in 2008 and 2014, respectively, under the supervision of Prof. Zhihua Zhou. He is currently a professor at Nanjing University of Aeronautics and Astronautics, and the dean of College of Computer Science and Technology. His research interests include machine learning and data mining.