Personalized Fashion Recommendation via RAG and SFT-Enhanced LLMs *

1st Xiayang Sun

I. Introduction

Traditional recommendation systems—despite decades of advancement—continue to face fundamental limitations when it comes to capturing the nuanced and dynamic preferences of users. These systems often rely on collaborative filtering or hand-crafted features.

One of the primary challenges lies in data sparsity: users typically engage with only a small subset of available items, leading to incomplete interaction matrices and suboptimal recommendations. The cold-start problem further complicates matters, as systems lack the data necessary to recommend relevant content to new users or for newly added items. Moreover, traditional engines are primarily built upon explicit feedback signals, overlooking the often subtle and implicit nature of consumer preferences. These systems also falter when handling niche or emerging items with limited historical data [1]–[3].

With help of langchain community, we present a personalized fashion recommendation system that integrates instruction-tuned Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG). Using the H&M Personalized Fashion Recommendation dataset, we extract item-level features, convert them into natural language chunks, and embed them with models on Huggingface MTEB-all-mpnet-base-v2, bge-small-en [7] comparison in terms of retrieval speed and accuracy. We also try to improve accuracy, controllability, and latency by selecting a better embedding model and chatbot, applying Qdrant Product Quantization (PQ) [6] instead of Qdrant only.

II. Methodology

The system follows a modular pipeline consisting of four core stages: data preprocessing by semantic chunk construction, embedding, dual-path model optimization (RAG and SFT), and final interaction via prompt engineer on a LLM.

A. Data Preprocessing and Semantic Chunking

We begin by cleaning and organizing the Huggingface **H&M Personalized Fashion Recommendation** dataset [4]. There are 25 columns out of which we'll be using 5 columns: *prod_name, product_type_name, colour_group_name, index_name, detail_desc*. Then all 5 features of each item are linked into a string by "-". For example:
"FLEECE PYJAMA - Pyjama jumpsuit/playsuit - Light Pink - Baby Sizes 50-98 - All-in-one pyjamas in soft ..."

Identify applicable funding agency here. If none, delete this.

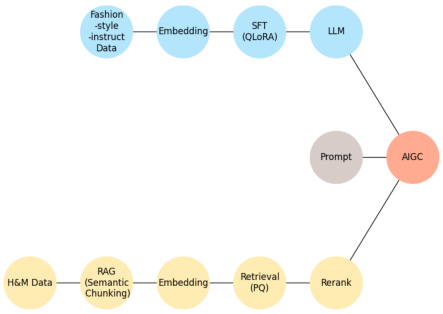

Fig. 1. Technology roadmapping: including RAG retrieval and rerank, QLoRA supervised fine-tunning for chat model and AIGC which consisting of four core stages: semantic chunking, embedding, dual-path model optimization (RAG and SFT), and prompt engineer via an LLM

This transformation facilitates downstream embedding and retrieval by aligning the structure of training data with natural user queries.

B. Embedding

To support efficient retrieval, each chunk description is embedded numerically. Previous study applied all-mpnet-base-v2 which holds a dimension of 768. However, we observed that dimension is not the main factor to effect the performance. So we apply bge-small-en (high MTEB accu- racy and proven production readiness in industry) with 384-dimension [7]. All embeddings are stored and queried within a Qdrant vector database.

C. Parallel Dual Optimization: RAG and SFT

Our system leverages two independent but complementary paths to optimize both retrieval quality and generation alignment:

1) Retrieval-Augmented Generation (RAG): Semantic retrieval is performed by computing similarity between the user's query embedding and the stored fashion item vectors in Qdrant.

2) Supervised Fine-Tuning (SFT): To tailor the LLM's generation capabilities to the fashion domain, we perform QLoRA-based supervised fine-tuning on DeepSeek-LLM-7B-Chat using the neuralwork/fashion-style-instruct dataset in Huggingface [5].

3) Recommendation via LLM chat model: At runtime, the system processes user input through the following stages: Embed the query and retrieve top-k relevant product chunks. Construct a prompt combining the user query and retrieved contexts. Feed the prompt into the fine-tuned DeepSeek-LLM-7B-Chat model. Embed the query and retrieve top-k relevant product chunks.

In step 2, we design a prompt to ensure LLM could retrieve RAG knowledge repository, rerank top-k relevant product chunks in step 1, and recommend top-20 items in a vivid and personalized tongue.

III. Efficiency Promotion

In previous work, the author implemented a personalized recommendation system using a combination of Mistral-7B-Instruct-v0.3 as the generation model and all-mpnet-base-v2 for retrieval and embedding in the RAG pipeline. While our replication revealed several practical issues affecting the system's accuracy, controllability, and latency.

A. Accuracy

First, we observed that the retrieved recommendations often failed to match the specific entities mentioned in the user's query. For instance, a user prompt such as "I want a summer dress in pastel colors" would yield a list of unrelated items like winter coats or men's shirts.

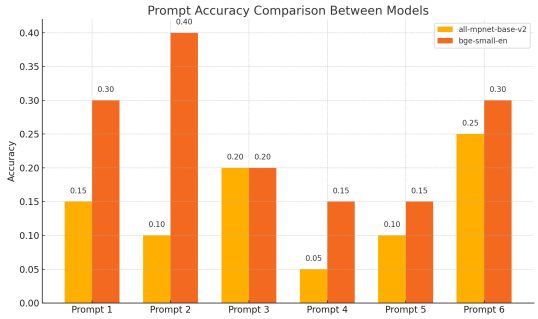

Fig. 2. Accuracy Comparison: all-mpnet-base-v2 vs. bge-small-en. 20 reranked most relevantly by Deepseek are selected from 100 chunks with highest scores after retrieval by embedding. Those with same product type as in query is voted 1, otherwise 0.

The accuracy rate is calculated by counting how many recommended items have the same product type with that in the corresponding prompt. We suggest using bge-small-en which ranks high in MTEB. The plot presents the result of the experiment conducted on 6 prompts, showing that bge-small-en could provide a better embedding function.

B. Controllability

Second, we found that Mistral-7B-Instruct-v0.3—despite being instruction-tuned—demonstrates limited control over prompt-specific generation constraints. The model failed to recognize the required number of recommend items in the query, and it couldn't generate a vivid description about the items, simply citing the description in chunks instead. Two adjustments are applied to handle this situation.

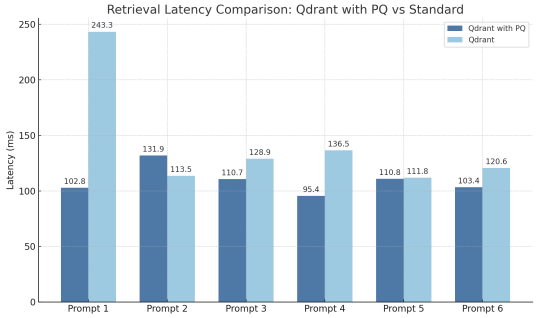

Fig. 3. Retrieval Latency Comparison: Qdrant with PQ vs. Standard Qdrant with bge-small-en

Therefore, we will replace Mistral-7B-Instruct-v0.3 with DeepSeek-LLM-7B-Chat which is one of the most chatbot models in 2025.

Besides, the original prompt is improved (The improvement would be shown in the appendix).

C. Latency

Finally, we could also reduce latency in retrieval by model quantization technique. To reach this target, we introduced Product Quantization (PQ) [6] within Qdrant, which compresses the vector representations by clustering and enables sublinear search performance. This technique resulted in speedups with minimal loss in retrieval accuracy, improving overall system responsiveness and user satisfaction.

We conducted a series of experiments using bge-small-en as the embedding model for retrieval. We compared retrieval times between the standard Qdrant setup (no compression) and the PQ-enabled setup.

We simulated 6 different users' inputs as prompts. For each prompt, we measured the time taken to complete retrieval and ranking on a 10^5-scale database. All experiments were run on a T100 16GB GPU. The results are summarized below:

Overall, the use of PQ resulted in a 5–10% improvement in average retrieval latency, making the system more responsive for end users. However, we also observed occasional fluctuations in retrieval time when PQ was enabled.

IV. Conclusion

This project proposes a unified pipeline for natural language-based fashion recommendation by combining semantic retrieval (RAG) and instruction-tuned generation (SFT) using Langchain and Qdrant. Product Quantization (PQ) is employed to reduce latency with minimal relevance loss. A QLoRA-tuned DeepSeek-LLM-7B-Chat model enhances controllability and expressiveness, especially with constrained prompts (e.g., "top 5 items"). Future improvements include: adding entity-type classifiers to boost category precision, using hybrid retrieval (coarse PQ + fine reranking) and incorporating image embeddings to enrich recommendations in visual domains.

REFERENCES

[1] https://medium.com/@shireenchand/leveraging-llm-fine-tuning-and-rag-for-advanced-recommendation-engines-a3d683e39976.

[2] J. Lin et al., "How Can Recommender Systems Benefit from Large Language Models: A Survey," arXiv preprint arXiv:2306.05817, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2306.05817

[3] https://zhuanlan.zhihu.com/p/694700684.

[4] Fashion Style Instruct — https://huggingface.co/datasets/neuralwork/fashion-style-instruct.

[5] H&M Fashion Dataset — https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations/data.

[6] H. Jégou, M. Douze and C. Schmid, "Product Quantization for Nearest Neighbor Search," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pp. 117-128, Jan. 2011, doi: 10.1109/TPAMI.2010.57.

[7] K. Luo, Z. Liu, S. Xiao, and K. Liu, "BGE Landmark Embedding: A Chunking-Free Embedding Method For Retrieval Augmented Long-Context Large Language Models," arXiv preprint arXiv:2402.11573, Feb. 2024. [Online]. Available: https://arxiv.org/abs/2402.11573

APPENDIX

A. Prompt Optimization

Original Prompt:

Question: {question} Give 5 best recommendations based on these articles: {context}.

Besides recommendations, you need to vividly introduce these items to me one by one based on these articles, and explain how they relate to {question}.

Answer:

Modified Prompt:

You are a fashion recommendation assistant. Based on the following product information and the user's query, select and recommend {num} most suitable fashion items.

You MUST pay extra attention to whether the product category in the user's question matches the product type name in the 2nd attribute (product_type_name) of each item.

The 5th attribute always starts with a phrase that names the product type, such as "Jacket in sweatshirt fabric", "Sweater with buttons", etc.

You should: 1. Only recommend items whose product type in the description matches the user's requested product type. 2. Recommend {num} best matches and briefly explain why each one is suitable, based on the full product description and its alignment with the user's request. 3. Vividly introduce the products in a friendly and engaging tone, explain how they relate to {question}

Here is the user request: {question}

Here are the available fashion items: {context}

Your introduction for each item should be listed one by one.

B. Query (question) List for the experiment

- I want a black cool sweater with shiny decorations making me looks like a movie star.
- Do you have pure white T-shirt with cute patterns at back?
- Please offer me some choices of jackets which are waterproof.
- Jeans for women, tight.
- Please offer me items like: light yellow dress or skirt which suit for summer, best for dating with my boyfriend.
- Jogging sports trousers, easy to take off, light, breathable, sweat absorption.