Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.086762

ARTICLE

Enhancing Personalized Fashion Recommendation by Integrating Large Language Models with Attribute Features

Ti-Lun Miao[1] **and Hsien-Tsung Chang**[1,2,3,*]

[1]Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan, Taiwan
[2]Department of Artificial Intelligence, Chang Gung University, Taoyuan, Taiwan
[3]Center for Artificial Intelligence in Medicine, Chang Gung Memorial Hospital at Linkou, Taoyuan, Taiwan

*Corresponding Author: Hsien-Tsung Chang. Email: smallpig@cgu.edu.tw

Received: 05 June 2026; Accepted: 12 August 2026; Published: 28 August 2026

ABSTRACT: Personalized fashion recommendation requires models that can capture visual compatibility, textual semantics, structured attributes, and user-specific preferences. However, existing multimodal approaches often rely on static word embeddings and shallow text encoders, limiting their ability to represent nuanced fashion descriptions. This study proposes a multimodal recommendation framework enhanced by large language models (LLMs) that integrates visual features, contextual textual representations, and structured attribute features for personalized outfit matching. A Japanese pretrained BERT encoder is used to replace the conventional Word2Vec and convolutional neural network (CNN)-based text pipeline, while GPT-4o is employed to extract fine-grained fashion attributes from product metadata. In addition, Llama-3.3-70B-Instruct is used to estimate semantic similarity among attribute values, enabling attribute-aware compatibility modeling beyond exact matching. Experiments on the IQON3000 dataset, containing 216,791 top-bottom outfit combinations, show that the proposed model achieves an area under the receiver operating characteristic curve (AUC) of 0.8477, outperforming the original Personalized Outfit Recommendation Scheme with Attribute-wise Interpretability based on Bayesian Personalized Ranking (PAI-BPR) baseline. Ranking-based evaluation further demonstrates that the proposed model consistently outperforms PAI-BPR across different candidate-set sizes and places compatible items closer to the top of the recommendation list. These results demonstrate that integrating large language models with multimodal and structured attribute features can effectively improve the accuracy and personalization of fashion recommendation systems.

KEYWORDS: Personalized fashion recommendation; multimodal recommendation; large language models; attribute-based modeling; semantic understanding

1 Introduction

In recent years, one of the most revolutionary breakthroughs in artificial intelligence has been the emergence of Large Language Models (LLMs) [1]. From early rule-based and statistical approaches in natural language processing (NLP) to the subsequent development of neural architectures such as Recurrent Neural Networks (RNNs) [2] and Convolutional Neural Networks (CNNs) [3], and now to large-scale pretraining based on the Transformer architecture [4], language understanding technologies have experienced a qualitative leap. Through large-scale pretraining on massive text corpora, LLMs are able to capture not only linguistic structures but also general world knowledge and reasoning patterns. Compared with task-specific models, LLMs exhibit superior generalization and transfer capabilities, allowing them to be flexible to a

wide variety of downstream applications. This combination of broad linguistic competence and contextual reasoning has positioned LLMs as key enablers in diverse domains.

Within the domain of e-commerce and recommender systems, LLMs have demonstrated exceptional text understanding abilities that enable the deep analysis of product descriptions and user reviews. In particular, the fashion industry provides an ideal testbed for exploring LLM capabilities, as fashion-related texts often carry multi-layered meanings involving aesthetics, emotions, and cultural context. LLMs can accurately interpret complex expressions such as "minimalist chic," "vintage style," or "elegant temperament," and capture subtle stylistic cues and affective tones that go beyond surface lexical similarity. This capacity for nuanced comprehension makes LLMs promising tools for fine-grained product analysis and personalized fashion recommendation, where understanding both content and style is critical.

Despite these advantages, most existing multimodal recommendation systems continue to rely on traditional text feature extraction techniques—such as Word2Vec [5] embeddings combined with CNN encoders—that are limited in their ability to capture deep semantic structures. These conventional approaches often perform adequately for general text classification but struggle when applied to domains with highly specialized or stylistically rich language, such as fashion. Fashion product descriptions typically include information across multiple semantic layers: material (e.g., "soft silk," "stretch cotton"), style (e.g., "French elegance," "street fashion"), usage scenario (e.g., "business formal," "casual vacation"), and even aesthetic tone or emotional impression (e.g., "intellectual charm," "youthful vitality"). When a system must recommend a lower-body garment to complement an upper-body item such as a "gentle style knitted top," it must be able to infer the underlying esthetic concept conveyed by "gentle style" and identify stylistically compatible options, such as a flowing chiffon skirt or high-waisted wide-leg pants. These expressions frequently incorporate cultural and subjective interpretations—terms such as "gentle style" or "salt-based look" represent not just literal meanings but cultural semantics specific to fashion that conventional embeddings do not capture.

With the rapid advancement of LLM technologies—exemplified by models such as BERT [6], GPT [7] and LLaMA [8]—it has become feasible to represent text with rich context-dependent semantics that encompass stylistic, emotional and cultural dimensions. This opens up new opportunities for leveraging LLMs in fashion recommendation tasks, enabling systems to move from shallow word-level similarity toward deeper semantic and conceptual understanding. The central research question of this work is therefore: *How can the semantic understanding power of LLMs be harnessed to more accurately analyze fashion descriptions and generate contextually appropriate outfit recommendations?* Addressing this question requires exploring how different LLM architectures can be effectively integrated into a multimodal recommendation framework and assessing whether these integrations yield measurable improvements in recommendation quality.

To achieve this goal, this study proposes a multimodal fashion recommendation framework that incorporates large language models as advanced text feature extractors. Given a user ID and an upper-body garment (e.g., a T-shirt, shirt, or jacket), the system automatically recommends the most compatible lower-body garment (e.g., pants or skirts) by jointly considering visual compatibility (color harmony, material coherence, and style alignment) and individual user preferences inferred from historical interactions. Fig. 1 illustrates the overall architecture of the system. The framework is designed to integrate LLM-derived textual representations with visual and attribute-level features, forming a tri-modal system that exploits the complementary strengths of each modality. Specifically, GPT-4o is employed to extract fine-grained item attributes—including color, design, pattern, sleeve length, garment length, silhouette, sleeve type, and neckline—from product metadata. LLaMA is then used to estimate semantic similarity between attribute values within the same category. These structured attributes and similarity scores are incorporated into a dedicated attribute module that complements the textual and visual representations.

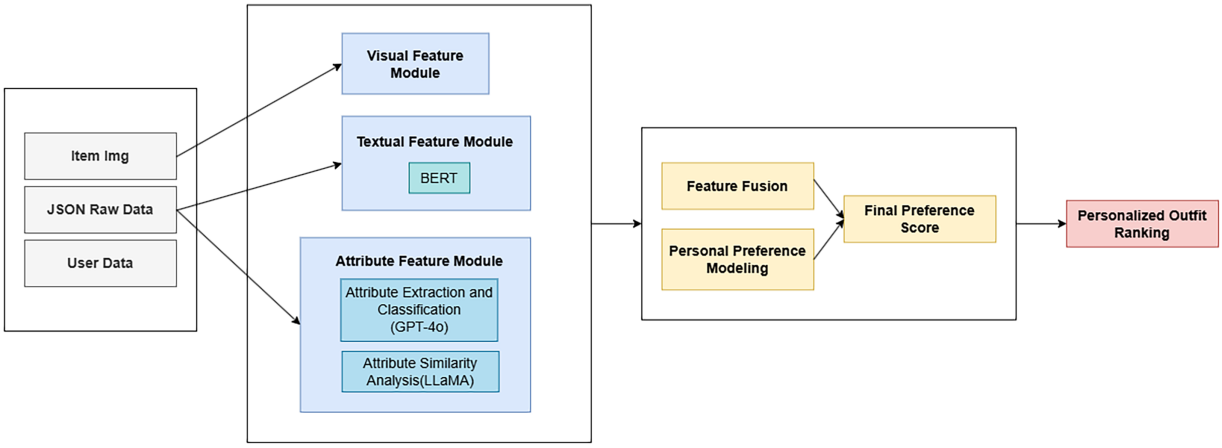

Figure 1: Overall architecture of the proposed LLM-integrated multimodal fashion recommendation system.

Building upon this integration, the objective of this study is to substantially enhance the semantic interpretability and personalization capability of fashion recommendation systems. Using the deep semantic reasoning power of LLMs, the proposed model advances from surface-level lexical matching to concept-level understanding, allowing it to better capture the nuanced semantics of fashion language. The effectiveness of the proposed method is validated in the **IQON3000** data set, demonstrating its ability to outperform the baseline multimodal models in outfit recommendation performance.

The contributions of this work are twofold:

1. **Deep semantic text feature extraction:** This study introduces BERT as a pretrained language encoder to replace traditional Word2Vec and CNN-based text processing methods, upgrading from static 300-dimensional word embeddings to dynamic 768-dimensional contextual representations. This transition significantly improves the system's comprehension of semantic and stylistic nuances in fashion descriptions.
2. **Intelligent attribute analysis and modular design:** By employing GPT-4o and LLaMA for intelligent attribute analysis, the system extracts fine-grained attributes—including color, design, pattern, sleeve length, garment length, silhouette, sleeve type, and neckline—and computes semantic similarity scores between attribute values. A dedicated attribute module is then incorporated as a third modality, complementing the textual and visual representations to improve recommendation accuracy.

Through this integration of large language models and multimodal learning, this study aims to advance the precision, interpretability, and personalization of fashion recommendation systems, contributing to the broader exploration of how LLMs can serve as semantic engines in complex, style-driven domains.

2 Literature Review

This section provides a concise overview of research on multimodal recommendation with a focus on fashion. We first trace the evolution of fusion strategies—from early feature concatenation to mid-level interaction and interpretable, attribute-aware models. We then review text representation advances from Term Frequency–Inverse Document Frequency (TF–IDF) and Word2Vec to contextual encoders (ELMo/BERT) and, most recently, large language models (LLMs) used for recommendation. Finally, we highlight practical issues such as cross-modal alignment, efficiency, and reliability (e.g., latency and hallucinations) that remain open. These gaps motivate our LLM-enhanced, attribute-aware multimodal framework for outfit compatibility and personalized fashion recommendation.

2.1 Architectures and Evolution of Multimodal Systems

A central challenge in multimodal recommendation is how to effectively fuse heterogeneous signals (e.g., text, images, and structured metadata) so as to capture complementary semantics and improve expressiveness and predictive performance. Early fusion strategies concatenated features at the input or representation level. A seminal example is VBPR [9], which incorporated CNN-based visual features of fashion items into a Bayesian personalized ranking framework to better model users' visual preferences. To move beyond simple concatenation, Tensor Fusion Networks (TFN) [10] explicitly modeled higher-order inter-modal interactions via outer products, although at a considerable computational cost; Low-rank Multimodal Fusion (LMF) [11] subsequently reduced this cost through low-rank factorization.

Mid-level (or "bottleneck") fusion mechanisms further improved the trade-off between accuracy and efficiency by enabling controlled cross-modal exchange while preserving modality-specific processing. Multimodal Bottleneck Transformers (MBT) [12] restrict information exchange to a small set of bottleneck tokens, which has proven effective when jointly modeling visual details from images and stylistic cues from text. Building on these trends, PAI-BPR [13] advances personalized and interpretable fashion recommendation by introducing an Attribute Classification Network (ACN) that decomposes items into fine-grained, human-interpretable attributes (texture, style, material, shape, and part-level details). The model uses a 2,048-dimensional attribute vector (plus an 8-dimensional color quantization via k-means) as visual representation and combines Word2Vec+TextCNN text encoders with a multi-layer perceptron (MLP) for non-linear cross-modal interactions, while matrix factorization captures user–item signals with a parameter μ to balance global compatibility and individual preference.

Recent collaborative filtering studies have also explored graph-based architectures to better capture user–item interaction structures. Alshareet and Ben Hamza [14] proposed an adaptive spectral graph wavelet framework for collaborative filtering, where users, items, and their interactions are represented as a bipartite graph. Their method applies an adaptive transfer function and spectral graph wavelets to learn low-dimensional user and item embeddings, allowing the model to capture both local and global graph structures in implicit-feedback recommendation.

This line of work is relevant because it demonstrates the effectiveness of graph-spectral modeling for collaborative filtering. However, its focus differs from the objective of the present study. Spectral graph wavelet methods mainly improve the modeling of user–item interaction topology, whereas our proposed LLM-PAI-BPR framework focuses on personalized top–bottom fashion compatibility prediction by enriching item representations with visual features, contextual textual representations, and LLM-assisted structured attribute features. Therefore, graph-spectral collaborative filtering and multimodal attribute-enhanced fashion recommendation address different but complementary aspects of recommender-system design.

2.2 Characteristics and Challenges of Fashion Recommendation

Fashion recommendation differs from general product recommendation in its heavy reliance on image and text signals that are difficult to discretize with a small set of tags. Beyond item names, critical details include silhouette, cut, color, and fabric. Type-aware embeddings [15] learn distinct subspaces for item categories (e.g., tops, bottoms, shoes) to model outfit compatibility across types. Scenario-dependent preferences (e.g., work, travel, or dating) further require recommendation strategies to adapt to context; few-shot preference modeling [16] helps for cold-start users and newly listed items. Rapidly evolving terminology and trends pose an additional challenge: emerging expressions (e.g., culturally grounded style descriptors) are hard to capture with traditional word embeddings or shallow CNN-based encoders, which limits semantic coverage and reduces recommendation fidelity.

2.3 Evolution of Text Feature Learning

Classical text representations in recommendation relied on frequency-based models such as TF–IDF [17], which ignore context and inter-word semantics. Word2Vec [5] advanced distributional semantics via CBOW/Skip-gram but yields static embeddings that cannot disambiguate word senses across contexts. Deep models then leveraged pretrained embeddings with convolutional neural network (CNN) and bidirectional long short-term memory (BiLSTM) encoders; for instance, Kim [3] demonstrated that CNNs capture salient local n-gram patterns and improve sentence classification. Contextualized representations marked a major shift in 2018: ELMo [18] produced dynamic embeddings from bidirectional long short-term memory networks (LSTMs), and BERT [6], built on the Transformer [4], achieved deep bidirectional context modeling through masked language modeling and next sentence prediction. For fashion text, such contextualization distinguishes nuanced senses (e.g., different eras of "vintage") and provides a stronger foundation for multimodal fusion.

2.4 Large Language Models for Recommendation

LLMs such as GPT and BERT have recently attracted attention in recommendation for their superior semantic understanding of user reviews and product descriptions. Empirical studies report gains from integrating unstructured signals (e.g., social media and review text) into recommenders [19]. To address the latency and deployment costs of large models, knowledge distillation to smaller student models (e.g., SLMRec) has been explored to retain semantic competence while improving efficiency [20]. In multimodal settings, LLMs can act as cross-modal "translators" to alleviate semantic misalignment between vision and language, which is particularly valuable in fashion tasks requiring both visual reasoning and stylistic understanding [21]. Nonetheless, practical issues remain: model size can induce serving delays and, in some cases, hallucinated recommendations; recent work highlights the need for efficiency, reliability, and alignment in LLM-driven recommenders [22].

2.5 Benchmark Evidence of LLM Understanding

Because this study leverages LLMs to process product descriptions and user reviews, it is important to situate their language understanding against standardized benchmarks. BERT, for example, reports strong scores on the General Language Understanding Evaluation (GLUE) benchmark and achieves high F1 on SQuAD v1.1 [6], while GPT-4 and successors attain competitive or near-expert performance on comprehensive benchmarks such as MMLU [23,24]. Although benchmark outcomes do not directly translate to recommendation quality, they indicate that modern LLMs possess robust contextual reasoning and semantic competence, motivating their adoption as textual backbones within multimodal fashion recommendation frameworks.

3 Methodology

This section describes in detail the methodology and techniques employed in this study to enhance the precision of personalized fashion recommendations. We propose a recommendation framework that deeply integrates Large Language Models (LLMs) with a tri-modal feature engineering approach. The section first introduces the dataset used in this study and explains how it is divided into training, validation, and test subsets. We then describe how visual, textual, and attribute-based features are extracted and processed, with particular emphasis on the use of LLMs—such as GPT-4o, LLaMA, and BERT—for understanding textual semantics, extracting product attributes, and computing attribute similarities. Finally, we explain the core architecture of our recommendation model, which jointly considers general outfit compatibility and

individual user preferences. Fig. 2 presents the overall architecture of the proposed recommendation system, detailing the complete processing pipeline from raw data input to final recommendation output.

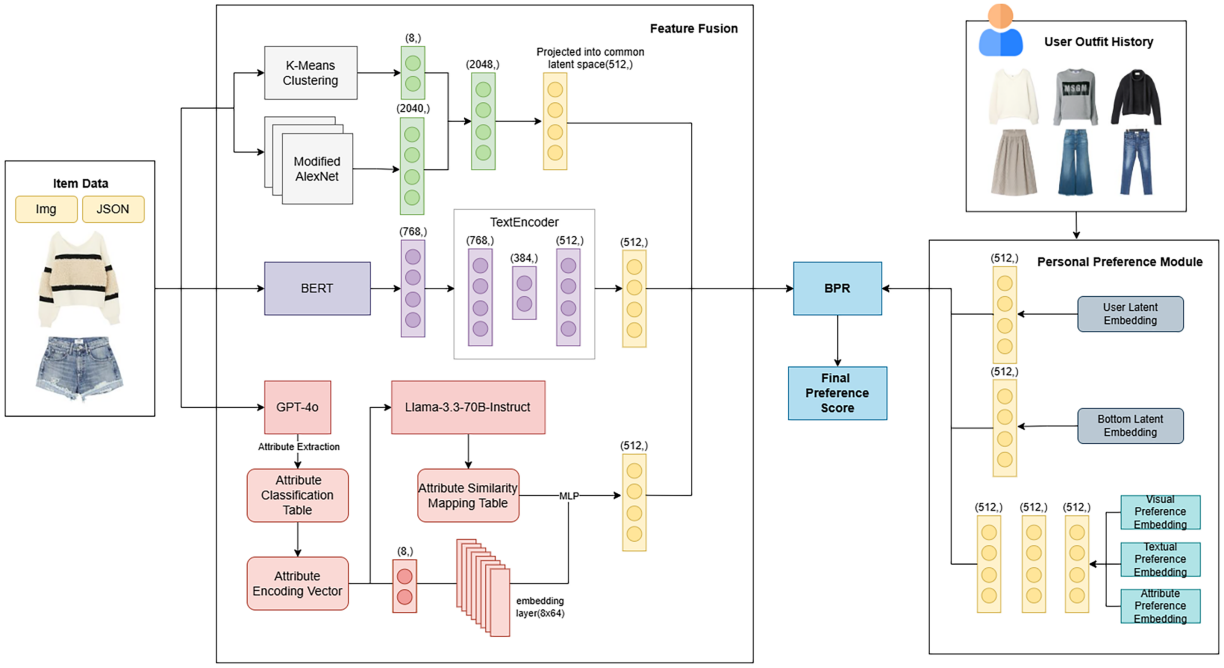

Figure 2: Overall system architecture of the proposed recommendation framework. The pipeline integrates visual feature extraction, BERT-based textual encoding, GPT-4o-based attribute extraction, LLaMA-based attribute similarity modeling, feature fusion, and personal preference modeling.

3.1 Multimodal Feature Engineering

We adopt a tri-modal fusion architecture comprising visual, textual, and attribute features. For the visual modality, following PAI-BPR [13], we employ a modified AlexNet trained for multi-task attribute classification to produce a 2,048-dimensional visual semantic vector.

3.1.1 Dataset

We evaluate our approach on the IQON3000 dataset [25], which is curated from the Japanese fashion social platform IQON. After collection and preprocessing by the original authors— including duplicate removal and quality control—the dataset contains 216,791 top–bottom outfit pairs created by 3568 users. IQON3000 focuses specifically on the outfit recommendation task between tops and bottoms, excluding accessories, shoes, and other categories. For each outfit, the data include item images and a structured JSON metadata file that records multi-dimensional attributes, such as `itemName`, `categorys`, hierarchical `breadcrumb`, `colors`, product `options`, and stylistic `expressions`. These rich textual signals provide a strong semantic basis for our BERT-based feature extraction and downstream LLM-driven analyses.

We adopt the standard train–validation–test split used in prior work. The detailed configuration is shown in Table 1.

Table 1: Dataset split of IQON3000.

Subset	# Samples	Proportion
Training	170,601	78.70%
Validation	23,095	10.65%
Test	23,095	10.65%
Total	216,791	100%

3.1.2 Selection and Application of Large Language Models

This study leverages three large language models at different stages of the pipeline, each serving a distinct role and exploiting complementary strengths.

GPT-4o

We employ GPT-4o [26] for attribute analysis on the IQON3000 dataset, primarily because of its strong multimodal understanding and extensive domain knowledge in fashion. By jointly processing images and textual metadata, GPT-4o identifies fine-grained item attributes, including color, design, pattern, sleeve length, garment length, silhouette, sleeve type, and neckline, and organizes them into a structured attribute taxonomy.

LLaMA

To construct an attribute–similarity lookup table, which requires a large number of application programming interface (API) calls (involving similarity judgments over thousands of attribute-value pairs), we adopt Meta's *Llama-3.3-70B-Instruct* [27] as a cost-effective solution. This model maintains high-quality semantic understanding while being more amenable to large-scale batch processing. Our experiments indicate that Llama-3.3-70B delivers competitive performance for fashion-domain semantic similarity estimation, supporting semantic smoothing of discrete attribute embeddings in the downstream recommendation model.

In our implementation, the similarity scoring prompt is fixed, and the model outputs a numerical score in the range $[0, 1]$, where a higher score indicates stronger semantic proximity. The generation settings are `temperature`=0.3, `top_p`=0.8, and `max_tokens`=10. The resulting similarity table is constructed offline and kept fixed during recommendation model training. Rather than treating these scores as human-annotated ground-truth labels, we use them as an LLM-based semantic prior for smoothing discrete attribute embeddings in the downstream recommendation model.

BERT

For Japanese text encoding, we replace the original CNN+Word2Vec pipeline with `tohoku-nlp/bert-base-japanese` [28]. Pretrained specifically for Japanese, this BERT variant better handles the product descriptions in IQON3000 and, through bidirectional attention, captures richer contextual semantics. This BERT encoder is used as a frozen feature extractor rather than being fine-tuned during recommendation model training.

Taken together, this multi-model strategy exploits the complementary strengths of different LLMs—supporting robust attribute extraction, semantic similarity evaluation, and high-quality text understanding across the key components of our system.

3.1.3 Textual Feature Enhancement

We replace the conventional CNN+Word2Vec pipeline with a Transformer-based BERT encoder for textual feature processing. Unlike Text-CNN, which relies on multiple convolutional kernels and pooling operations, BERT's bidirectional contextual modeling captures sentence-level semantics more comprehensively. This substitution not only improves the quality of semantic representations but also simplifies the overall architecture, avoiding additional convolution/pooling layers and easing integration into the recommendation model.

All product-related text in the IQON3000 dataset is written in Japanese—covering fields such as item names, brand information, and category descriptions. In the original JSON files, Japanese strings are stored as Unicode escape sequences (e.g., `\u30b8\u30e3\u30b1\u30c3\u30c8` represents "ジャケット"), which decode to standard Japanese text during parsing. To accommodate the linguistic characteristics of Japanese (e.g., mixed usage of kana and kanji, relatively flexible word order), we adopt `tohoku-nlp/bert-base-japanese` as our text encoder. Pretrained on Japanese corpora, this model provides more accurate handling of domain-specific fashion vocabulary than multilingual BERT, enabling finer-grained understanding of nuances in product descriptions.

For feature construction, we concatenate multiple textual fields—`itemName`, `categorys`, `colors`, `options`, and `expressions`—after trimming whitespace and removing duplicated phrases, forming a unified semantic description per item. The combined text is then encoded by BERT, and the output of the `[CLS]` token is used as the item's textual semantic representation, yielding a 768-dimensional dense vector. The overall process is illustrated in Fig. 3.

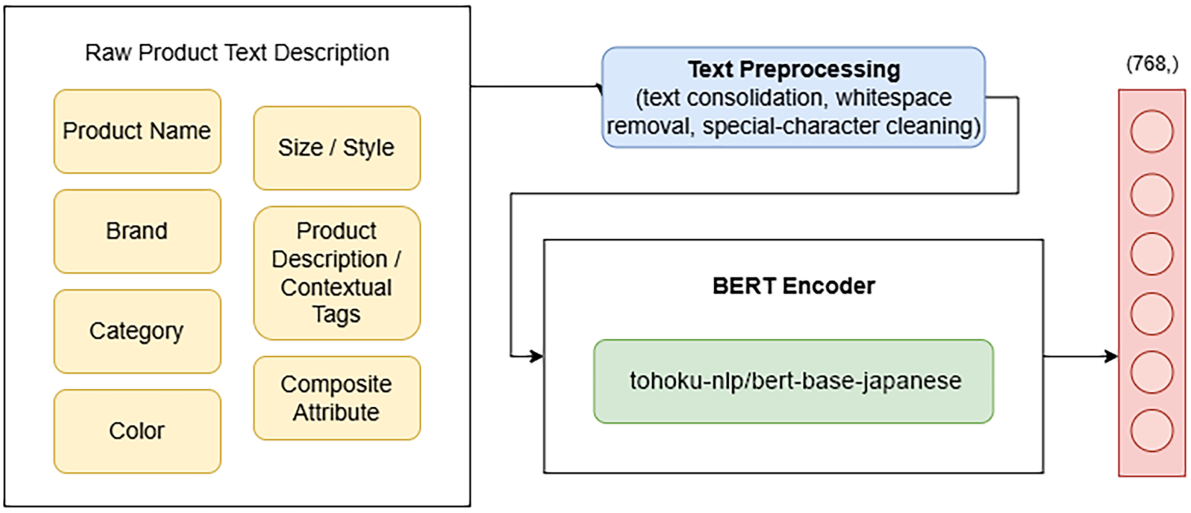

Figure 3: Text feature processing pipeline. Product metadata fields are consolidated into a unified textual description and encoded by a Japanese BERT model to generate a 768-dimensional semantic representation.

3.1.4 Attribute Feature Design

While BERT excels at contextual semantic understanding, the JSON metadata of each product contains numerous discrete attribute tags—such as color codes, size specifications, and style categories. When these structured attributes are verbalized into free-form text, their discrete nature and clear categorical boundaries may be diluted in BERT's continuous semantic space. To preserve and exploit this information, we design a dedicated attribute module that complements BERT-based textual features: (i) when product descriptions are short or missing, structured attributes provide a reliable fallback representation; and

(ii) the explicit categorical structure of attributes improves interpretability, enabling the system to articulate recommendation rationales (e.g., "same color family," "style consistency").

Conventional attribute matching typically checks only for exact equality (e.g., "red" equals "red") and cannot capture fine-grained semantic relations such as the proximity between "pink" and "light pink." We therefore introduce an LLM-based attribute similarity computation to model nuanced relations among attribute values. This allows the system to determine which attributes are compatible even when they are not identical, effectively bridging structured attribute signals with semantic understanding and thereby improving both accuracy and flexibility in recommendation. The overall attribute feature processing pipeline is illustrated in Fig. 4. It consists of GPT-4o-based attribute extraction, attribute vector encoding, and LLaMA-based attribute similarity table construction.

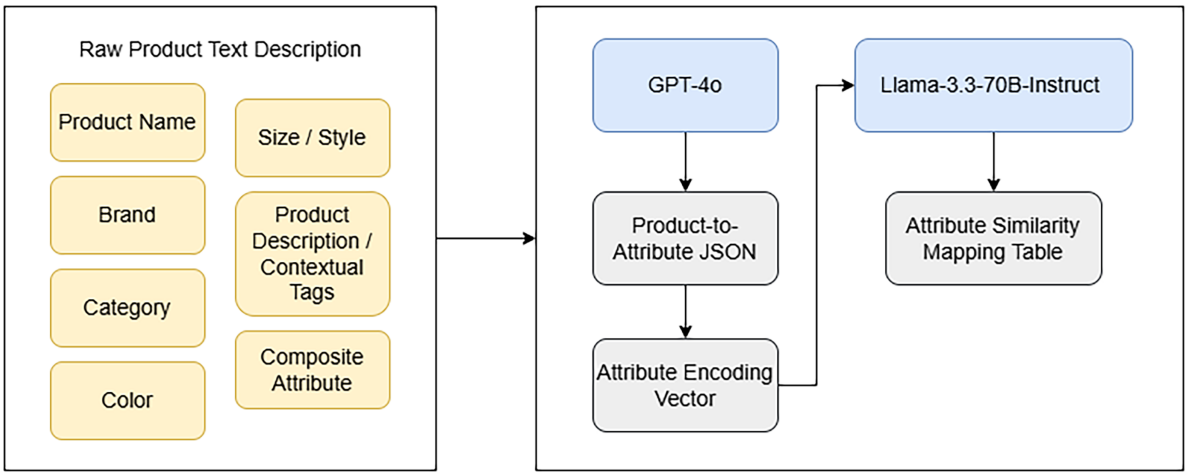

Figure 4: Attribute feature processing pipeline. Product metadata are first converted into structured attribute JSON outputs using GPT-4o, then encoded into attribute vectors and further used to construct an attribute similarity mapping table with LLaMA.

Attribute Extraction and Categorization

To convert unstructured product descriptions into computable structured representations, we employ OpenAI's GPT-4o Large Language Model (LLM) to perform semantic parsing and attribute extraction over the JSON metadata. We design a structured prompt template—comprising a task description, definitions of eight attribute categories, output formatting rules, and illustrative examples (see full prompt in Fig. 5)—to guide the model toward consistent, schema-aligned outputs.

Leveraging GPT-4o's deep language understanding, we extract key fashion attributes and normalize them into (category, value) pairs under eight high-level categories: Color, Design, Pattern, Sleeve Length, Length, Silhouette, Sleeve, and Neck. The resulting vocabulary sizes for each category are as follows: Color (257), Design (214), Pattern (254), Sleeve Length (315), Length (69), Silhouette (58), Sleeve (216), and Neck (34), totaling 1417 fine-grained attribute values.

The attribute vocabulary and the LLaMA-based attribute similarity table are constructed from the full processed IQON3000 item catalog, including items appearing in the training, validation, and test splits. Therefore, the evaluation follows a transductive item-catalog setting rather than an inductive cold-start setting. Test-set interaction labels and outfit-pair compatibility labels are not used for model training, early stopping, or modality-weight selection; they are used only for final evaluation.

```
You are a fashion product attribute analyzer and classifier.
Based on the following product information, complete the following two tasks:

1. Determine whether the product is "top" (upper-body garment),
   "bottom" (lower-body garment), or "other".

2. Extract high-level fashion attributes and their corresponding values,
   and present them in JSON format.

For example:
{
  "color": "black",
  "sleeve_length": "short sleeve",
  "Pattern": "Solid Color",
  "Design": "V-neck"
}

Please return a JSON object in the following format:

{
  "type": "top",
  "attributes": {
    "color": "...",
    "Pattern": "...",
    ...
  }
}

If an attribute cannot be determined, omit it.
If the item cannot be classified, set "type" to "other".

Return only a valid JSON object.
Do not include explanations, comments, or markdown.

Product information:

* itemName: [itemName]

* categorys: [categorys]

* breadcrumb: [breadcrumb]

* colors: [colors]

* options: [options]

* expressions: [expressions]
```

Figure 5: English-translated prompt template used for fashion item classification and high-level attribute extraction. In the actual implementation, Chinese task instructions were used together with the original Japanese IQON3000 product metadata. The model was instructed to return only a valid JavaScript Object Notation (JSON) object for downstream processing.

To improve extraction consistency, GPT-4o is used with a fixed prompt, JSON-format output constraints, and deterministic decoding with temperature set to 0. The extracted results are cached so that

each item is processed consistently across experiments, and invalid or missing attributes are mapped to the reserved missing-value index in the attribute module.

These controls improve consistency but do not replace an independent validation of extraction quality. Since IQON3000 does not provide ground-truth labels for the eight normalized attribute categories, this study does not report per-category precision, recall, or F1-score for GPT-4o-based attribute extraction. Manual or semi-automatic validation of schema violations, unsupported attributes, missing rates, and consistency with the original Japanese metadata remains future work.

The distribution of the extracted attribute values and an example of the structured JSON output are shown in Fig. 6.

Categories	Number
Color	257
Design	214
Pattern	254
Sleeve Length	315
Length	69
Silhouette	58
Sleeve	216
Neck	34
Total	**1417**

```
"37396155" : {
  "Color":    "Beige",
  "Pattern":  "Plain",
  "Design":   "Turtleneck",
  "Sleeve Length":    "Long sleeve"
}
```

Figure 6: Attribute distribution statistics and an example of product attribute extraction. The left panel reports the value counts of the eight fashion attribute categories, and the right panel presents an example structured JSON output generated by the attribute extraction process.

Attribute Encoding and Vector Representation

To obtain a numeric representation of product attributes, we assign a unique positive integer ID to each attribute value, yielding a compact encoding scheme. For example, the encoding map for the Color category is shown in Table 2.

Table 2: Illustrative subset of the color attribute encoding map.

Attribute Value	Encoded ID
Brown	1
Gray	2
White	3
Beige	4
Black	5
Green	6
Yellow	7
Blue	8
Purple	9
Orange	10

Based on this scheme, each item is represented by an 8-dimensional attribute vector $\mathbf{v} = [v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8]$, where v_i denotes the encoded ID for the i-th attribute category. If an item lacks information for a given attribute, that dimension is set to 0, indicating a missing value.

As an illustrative example, Table 3 shows that item ID `37396155` is represented by the following 8-dimensional attribute vector:

$$\mathbf{v} = [4,\ 2,\ 254,\ 315,\ 0,\ 0,\ 0,\ 0],$$

where the elements correspond to *Color*, *Design*, *Pattern*, *Sleeve Length*, *Length*, *Silhouette*, *Sleeve*, and *Neck*, respectively; a value of 0 marks the absence of that attribute.

Table 3: Dimensions of the 8D attribute vector.

Dim	Attribute Category	Encoded ID	Attribute Value
v_1	Color	4	Beige
v_2	Design	2	Turtleneck
v_3	Pattern	254	Plain
v_4	Sleeve Length	315	Long sleeve
v_5	Length	0	(missing)
v_6	Silhouette	0	(missing)
v_7	Sleeve	0	(missing)
v_8	Neck	0	(missing)

Through this process, IQON3000 items are transformed from unstructured fashion descriptions into standardized numerical representations, providing a consistent data foundation for model training, fashion recommendation algorithms, and style similarity computation.

Attribute Similarity

Naively computing similarity via inner products is fundamentally flawed in our setting. The attribute IDs are assigned by enumerating the GPT-4o–derived attribute lists within each category on IQON3000; hence, the numeric IDs themselves carry *no* semantic meaning. For example, within the *Color* category, "Pink" might be encoded as 1 and "Baby Pink" as 11. Although these two colors are close in colorimetry, the numeric gap between their IDs does not reflect semantic proximity, causing inner-product–based similarity to deviate markedly from true attribute similarity.

To address this issue, we construct a per-category similarity lookup table over all pairs of attribute values using Meta's *Llama-3.3-70B-Instruct*. We design a dedicated prompt (see Fig. 7) that instructs the model to assess pairwise similarity grounded in fashion knowledge and to output a score in $[0, 1]$, where 1 denotes identical (or effectively equivalent) and 0 denotes entirely unrelated. An example for the *Color* category is shown in Fig. 8.

LLM Inference Settings

For GPT-4o-based attribute extraction, the model identifier is `gpt-4o`. The temperature is set to 0.0 because the task requires stable JSON-formatted outputs rather than diverse generation. The prompt explicitly asks the model to return only a JSON object, and the extracted results are cached for reuse.

The top_p value and maximum output length are not explicitly specified in the original API call; therefore, the API default settings are used.

```
You are a fashion styling consultant. Please refer to all values under the
following attribute category: [attribute_type]: [values].


Then, evaluate whether the two values, [v1] and [v2], are
semantically similar or interchangeable within this attribute
category.


If the two values are highly similar or consistent in terms
of outfit coordination logic, assign a score of 1.


If they are completely different, assign a score of 0.
For all other cases, assign a score between 0.1 and 0.9.


Return only a single numeric value (e.g., 0.7).
```

Figure 7: English-translated prompt template used for attribute similarity scoring. In the actual implementation, Chinese task instructions were used to ask Llama-3.3-70B-Instruct to evaluate whether two normalized attribute values are semantically similar or interchangeable within the same fashion attribute category. The model was instructed to return only a single numerical score in the range of $[0,1]$.

```
"Color": {
    "Pink|Brown": 0.2,
    "Pink|Gray": 0.2,
    "Pink|White": 0.2,
    "Pink|Beige": 0.2,
    "Pink|Black": 0.1,
    "Pink|Green": 0.2,
    "Pink|Yellow": 0.2,
    "Pink|Blue": 0.2,
    "Pink|Purple": 0.4,
    "Pink|Orange": 0.2,
    "Pink|Baby Pink": 0.8
}
```

Figure 8: Example of an attribute similarity mapping table for the Color category generated by Llama-3.3-70B-Instruct. Each key represents a pair of attribute values, and each score indicates their semantic similarity for outfit coordination.

For the LLaMA-based attribute similarity table, the model identifier is `meta-llama/Llama-3.3-70B-Instruct`. The generation settings are `temperature`=0.3, `top_p`=0.8, and `max_tokens`=10. This low-temperature nucleus-sampling setting is used to obtain relatively stable numerical similarity scores while keeping the output short. The prompt asks the model to return only a numerical score in the range $[0,1]$. The similarity table is constructed once offline and then fixed before recommendation

model training and inference. Since attribute similarity represents semantic proximity between two values within the same attribute category, rather than a directional compatibility relation, the table is treated as symmetric:

$$\mathrm{sim}(a, b) = \mathrm{sim}(b, a).$$

During preprocessing, only one score is generated for each unordered pair of attribute values. Therefore, reverse pairs are not queried separately, and the same stored score is used for both directions during lookup. The self-similarity of an attribute value is defined as 1.

3.2 Model Architecture and Design

Before presenting the mathematical formulation of the proposed model, Table 4 summarizes the major symbols used throughout this section for ease of reference.

Table 4: Summary of mathematical notation used in the proposed model.

Symbol	Description
m	User index
i	Top item index
j	Positive bottom item index
k	Negative bottom item index
q	Generic item index, where $q \in \{i, j, k\}$ when needed
p_{ij}^{m}	Preference score of user m for pairing top i with bottom j
s_{ij}	General compatibility score between top i and bottom j
c_{mj}	Personal preference score of user m for bottom item j
μ	Balance coefficient between general compatibility and personal preference
$\lambda_v, \lambda_t, \lambda_a$	Fusion weights for visual, textual, and attribute modalities
$\mathbf{x}_q^v$	Raw 2,048-dimensional visual feature vector of item q
$\mathbf{z}_q^v$	Projected 512-dimensional visual representation of item q
$\mathbf{x}_q^t$	Raw 768-dimensional BERT textual representation of item q
$\mathbf{z}_q^t$	Projected 512-dimensional textual representation of item q
$\mathbf{z}_q^a$	512-dimensional attribute representation of item q
b_m, b_j	User-specific and item-specific bias terms
$\mathbf{u}_m, \mathbf{v}_j$	Latent factor vectors of user m and bottom item j
$\boldsymbol{\xi}_m^{\mathrm{vis}}$	User preference vector for the visual modality
$\boldsymbol{\xi}_m^{\mathrm{text}}$	User preference vector for the textual modality
$\boldsymbol{\xi}_m^{\mathrm{attr}}$	User preference vector for the attribute modality
$\mathbf{f}_j^{\mathrm{vis}}$	Visual feature vector of bottom item j used in personal preference modeling
$\mathbf{f}_j^{\mathrm{text}}$	Textual feature vector of bottom item j used in personal preference modeling
$\mathbf{f}_j^{\mathrm{attr}}$	Attribute feature vector of bottom item j used in personal preference modeling
η_v, η_t, η_a	Modality weights in personal preference modeling
Θ	Set of trainable model parameters
λ_{reg}	ℓ_2 regularization coefficient in the BPR loss

3.2.1 Overall Framework

We adopt an LLM-enhanced multimodal fashion recommendation system that models both general compatibility and personal preference in a dual-objective manner [13]. The core formulation is:

$$p_{ij}^{m} = \mu \cdot s_{ij} + (1 - \mu) \cdot c_{mj}$$

where:

- p_{ij}^{m}: the preference score of user m for pairing top i with bottom j
- s_{ij}: the general compatibility score between top i and bottom j
- c_{mj}: the personal preference score of user m for bottom j
- μ: a weighting parameter that balances the two modeling components

3.2.2 Tri-Modal Feature Fusion Architecture

We propose a tri-modal fusion mechanism:

$$s_{ij} = \lambda_v \cdot \text{visual\_sim}(i, j) + \lambda_t \cdot \text{textual\_sim}(i, j) + \lambda_a \cdot \text{attribute\_sim}(i, j),$$

where λ_v, λ_t, and λ_a denote the modality fusion weights for the visual, textual, and attribute modalities, respectively, and satisfy:

$$\lambda_v + \lambda_t + \lambda_a = 1.$$

For the visual modality, we reuse the pre-extracted features from PAI-BPR [13]. These features are obtained by a modified AlexNet trained for multi-task attribute classification and combined with k-means color quantization, yielding a raw 2048-dimensional visual feature vector. However, this raw visual vector is not directly used by the fusion module. Since the multimodal fusion module operates in a shared 512-dimensional latent space, the 2048-dimensional visual feature is first passed through a trainable visual encoder consisting of a linear projection layer followed by a sigmoid activation:

$$\mathbf{z}_q^v = \sigma(\mathbf{W}_v \mathbf{x}_q^v + \mathbf{b}_v), \quad q \in \{i, j\},$$

where $\mathbf{x}_q^v \in \mathbb{R}^{2048}$ denotes the raw visual feature of item q, $\mathbf{W}_v \in \mathbb{R}^{512\times 2048}$ and $\mathbf{b}_v \in \mathbb{R}^{512}$ are trainable parameters, $\sigma(\cdot)$ denotes the sigmoid activation function, and $\mathbf{z}_q^v \in \mathbb{R}^{512}$ is the projected visual representation used for fusion.

The textual and attribute modalities are also transformed into 512-dimensional representations, as described in the following subsections. Therefore, for each top–bottom pair $(i,\ j)$, the fusion module receives aligned visual, textual, and attribute representations for both items:

$$\mathbf{z}_q^v, \mathbf{z}_q^t, \mathbf{z}_q^a \in \mathbb{R}^{512}, \quad q \in \{i, j\}.$$

The similarity scores in the visual, textual, and attribute modalities are computed using these aligned 512-dimensional representations, ensuring dimensional consistency before the weighted fusion step.

3.2.3 Textual Feature Enhancement Module

For the textual modality, encoding product descriptions with `tohoku-nlp/bert-base-japanese` yields a 768-dimensional semantic vector. Since the multimodal fusion module operates in a shared 512-dimensional latent space, the BERT-based representation must be transformed before fusion.

To align dimensions while preserving important semantic information, we adopt a bottlenecked *TextEncoder* module:

$$\mathbb{R}^{768} \to \mathbb{R}^{384} \to \mathbb{R}^{512}.$$

The first linear transformation compresses the 768-dimensional BERT representation into a 384-dimensional bottleneck representation, followed by a GELU (Gaussian Error Linear Unit) activation and Dropout regularization. The second transformation maps the compressed representation to 512 dimensions, followed by Layer Normalization and Dropout. This bottleneck design encourages the model to select and retain salient semantic cues from the BERT representation while producing a textual feature vector compatible with the visual and attribute modalities.

Formally, given the BERT textual representation $\mathbf{x}_q^t \in \mathbb{R}^{768}$, the textual encoder produces:

$$\mathbf{z}_q^t = \text{TextEncoder}(\mathbf{x}_q^t), \quad q \in \{i, j\},$$

where $\mathbf{z}_q^t \in \mathbb{R}^{512}$ is the final textual representation used in the tri-modal fusion architecture.

3.2.4 Attribute Feature Module

The attribute feature module is tailored to encode explicit semantic properties of products (e.g., color, pattern, design, sleeve type, neckline), complementing visual and textual signals where they may fail to capture fine-grained categorical differences. It further leverages an attribute similarity matrix to increase the recommender's sensitivity to semantic compatibility.

For each attribute category, we instantiate a trainable embedding layer that maps attribute values to 64-dimensional dense vectors. Index 0 in every embedding table is reserved to denote missingness (e.g., an item without a recorded neckline) and is mapped to the all-zero vector so that the model can explicitly recognize and handle missing attributes—reflecting realistic data conditions.

Because raw attribute labels are discrete and their ordinal IDs do not encode semantics, we introduce a similarity-based smoothing mechanism. Building on an LLM-derived similarity lookup table, we model pairwise semantic proximity among values within each attribute category. After the initial lookup embedding, we compute a similarity-weighted average of embeddings according to the lookup scores, so that semantically close but differently coded values (e.g., "Pink" and "Light Pink") are pulled nearer in the vector space, improving semantic generalization and compatibility reasoning.

After embedding and smoothing, the eight attribute embeddings are concatenated to form a 512-dimensional preliminary attribute feature vector. This vector is then fed into a three-layer Multi-Layer Perceptron (MLP) with two ReLU nonlinearities and Dropout regularization, finally producing a 512-dimensional attribute semantic representation. This representation serves as the attribute modality in the tri-modal fusion architecture, participating—together with textual and visual semantics—in feature fusion and compatibility estimation for outfit recommendation.

3.2.5 Personal Preference Modeling

This component aims to capture each user's individualized style tendencies in clothing selection. Concretely, for every user the system learns preferences over different styles or attributes—for example, some users favor light colors, some prefer formal styles, while others particularly like specific patterns. Such preferences may reside in the item's visual appearance, textual description, or structured attributes (e.g., color,

material, pattern). Accordingly, we decompose user preference learning into three modalities: visual, textual, and attribute.

The overall personal preference score c_{mj} denotes user m's affinity for a bottom item j and is computed as:

$$c_{mj} = b_m + b_j + \mathbf{u}_m^\top \mathbf{v}_j + \eta_v (\boldsymbol{\xi}_m^{\text{vis}})^\top \mathbf{f}_j^{\text{vis}} + \eta_t (\boldsymbol{\xi}_m^{\text{text}})^\top \mathbf{f}_j^{\text{text}} + \eta_a (\boldsymbol{\xi}_m^{\text{attr}})^\top \mathbf{f}_j^{\text{attr}}.$$

The formulation consists of three parts:

- **Bias terms:** b_m and b_j capture user- and item-specific base preferences.
- **Latent factor interaction:** $\mathbf{u}_m$ and $\mathbf{v}_j$ are latent vectors representing user and item style tendencies; their inner product measures latent style compatibility learned from data.
- **Multimodal semantic interaction:** For each user, we learn modality-specific preference vectors $(\boldsymbol{\xi}_m^{\text{vis}}, \boldsymbol{\xi}_m^{\text{text}}, \boldsymbol{\xi}_m^{\text{attr}})$ and compute inner products with the corresponding item feature vectors $(\mathbf{f}_j^{\text{vis}}, \mathbf{f}_j^{\text{text}}, \mathbf{f}_j^{\text{attr}})$. The coefficients η_v, η_t, and η_a control the visual, textual, and attribute contributions in the personal preference component.

This modeling approach uncovers users' implicit style inclinations while leveraging interpretable semantic cues (e.g., attributes like "striped" or "chiffon"), thereby providing a more comprehensive profile of fashion preferences. The resulting scores are matched against each item's multimodal features and used for ranking.

3.3 Training Phase and Loss Function

We train the model using the Bayesian Personalized Ranking (BPR) objective [29], whose core goal is to rank items that a user prefers ahead of those the user dislikes. In practice, we follow the proposed PAI-BPR framework [13].

At each training step, the model samples a quadruple (m, i, j, k), where m denotes a user ID, i a top item, and j and k two different bottom items, with j being an observed positive pair (matched with i) and k an unobserved negative sample. The objective is to maximize the probability that the preference score for the matched pair (i, j) exceeds that for the non-matched pair (i, k), thereby encouraging the model to learn user-specific outfit preferences so that more plausible combinations are ranked higher at inference time.

The loss function is defined as:

$$\mathcal{L}_{\text{BPR}} = -\sum_{(m,i,j,k)\in\mathcal{D}} \ln \sigma\left(p_{ij}^m - p_{ik}^m\right) + \lambda_{\text{reg}} \|\Theta\|^2,$$

where σ is the sigmoid function, Θ denotes all trainable parameters of the model, and λ_{reg} is the ℓ_2 regularization coefficient used to mitigate overfitting.

3.4 Evaluation Metrics

Because our task focuses on relative ranking in outfit recommendation—prioritizing better top–bottom matches rather than merely deciding suitability—we adopt two primary metrics: the Area Under the Receiver Operating Characteristic Curve (AUC) and Mean Reciprocal Rank (MRR). AUC assesses whether preferred matches are ranked ahead of non-preferred ones, while MRR emphasizes the system's ability to surface the single best match in a retrieval setting. The two are complementary and together capture overall ranking quality and personalization accuracy.

3.4.1 Area under the ROC Curve (AUC)

We select AUC as a principal metric because the core objective in recommender systems is **ranking**, not binary classification. In our experiments, AUC is computed under the BPR pairwise-comparison scheme, measuring whether a user-preferred outfit (i, j) is ranked above a non-preferred pairing (i, k). Scores approaching 1 indicate that the model accurately captures user preferences.

For fairness and comparability, we follow the original PAI-BPR data split protocol. We select the best checkpoint on the validation set and report the final AUC on the held-out test set, mitigating overfitting and ensuring reliable evaluation.

3.4.2 Mean Reciprocal Rank (MRR)

As a complementary metric, we report MRR to quantify ranking performance in a personalized retrieval scenario. Unlike AUC's pairwise focus, MRR is suited to cases where the system must select the **single most appropriate** item from a candidate set. Concretely, given a query composed of user u_m and a top t_i, the model ranks T candidate bottoms, among which exactly one is the ground-truth match from the user's history. Let rank_q denote the position of the correct bottom for query q under scores p_{ij}^m. MRR is then

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q}.$$

If the correct item is ranked first, the reciprocal rank is 1; if third, it is 1/3, and so on. Higher MRR indicates stronger ability to place the true match near the top.

3.4.3 Top-K Mean Reciprocal Rank (MRR@K)

In practical recommender interfaces, users typically only inspect the top few results. Beyond overall MRR, we therefore report MRR@K to evaluate performance within the first K positions. Whereas MRR credits any rank position (e.g., 1/20 for 20th), MRR@K only rewards the correct answer if it appears in the top K; otherwise the contribution is zero:

$$\text{MRR@}K = \frac{1}{|Q|} \sum_{q \in Q} \left[\mathbb{I}(\text{rank}_q \le K) \cdot \frac{1}{\text{rank}_q} \right],$$

where $\mathbb{I}(\cdot)$ is the indicator function that equals 1 when the correct item ranks within the top K, and 0 otherwise.

In summary, our methodology strengthens the baseline outfit recommendation pipeline through attribute vector construction, semantic extraction, similarity lookup, and fusion strategies. By combining LLM-based processing of unstructured text with attribute-level compatibility modeling, the system more finely assesses item–item compatibility and user preferences, thereby improving overall recommendation effectiveness. The next section presents experimental results and analysis demonstrating the method's practical performance.

4 Results and Discussion

This section systematically presents and analyzes the experimental results of our multimodal personalized fashion recommendation model. We organize the evaluation into four parts. First, we quantify the contribution of each modality (vision, text, attributes) via single-modality ablations and weighted-fusion studies to identify optimal fusion settings. Second, we compare against multiple baselines to verify the accuracy gains of our approach. Third, we assess ranking performance using Mean Reciprocal Rank (MRR), simulating practical Top-k recommendation scenarios. Finally, we provide visualized ranking cases

to illustrate how the model captures user-specific outfit preferences and to discuss avenues for further improvement. These findings confirm the effectiveness of the proposed framework and offer guidance for the design and deployment of future multimodal recommender systems.

4.1 Experimental Environment and Settings

This section details the experimental setup and dataset specifications. We first describe the hardware and software environments used for training and evaluation. We then present the datasets, including their sources, scale, and organization.

4.1.1 Hardware Environment

The hardware setup used in this study is summarized in Table 5.

Table 5: Hardware environment specifications.

Component	Specification
CPU	13th Gen Intel(R) Core(TM) i5-13600K
CPU Specs	3.50 GHz; 14 cores (6P + 8E); 20 logical processors
RAM	16.0 GB
GPU	NVIDIA GeForce RTX 4070 Ti
Operating System	Windows 11

4.1.2 Software Environment

The software environment is summarized in Table 6, and the key library versions are listed in Table 7.

Table 6: Software environment specifications.

Component	Version/Details
Python	3.11.5
PyTorch CUDA runtime	12.4
GPU Driver	560.94

Table 7: Key libraries used in the experiments.

Category	Library	Version
Core Models	transformers	4.53.0
	torch	2.6.0+cu124
	torchvision	0.21.0+cu124
	torchaudio	2.6.0+cu124
	tokenizers	0.21.2
Supporting Packages	pandas	2.3.1
	Pillow	11.0.0
	numpy	2.1.2
	opencv-python	4.11.0.86
Others	openai	1.88.0
	seaborn	0.13.2

4.2 Weighting Experiments

In multimodal recommendation systems, the contribution of each modality may be uneven. Simply concatenating multimodal features may therefore fail to fully exploit their respective strengths. To further improve performance, we conduct a systematic weight search to identify the optimal combination of Visual (Vis), Text (Text), and Attribute (Attr) feature weights.

All weighting experiments in this section are conducted on the validation set. The training set is used for model parameter learning, the validation set is used for early stopping and modality-weight selection, and the held-out test set is used only for final performance evaluation after the best weight configuration has been fixed.

4.2.1 Unimodal Ablation Analysis

To design a reasonable weighting scheme, we first conduct unimodal ablation experiments in which each modality is fed to the model independently (Table 8). The results show that textual features perform best (AUC = 0.8430), while visual and attribute features are comparable (0.8286 and 0.8271, respectively). This suggests that text serves as the primary discriminative signal, with vision and attributes providing complementary information.

Table 8: Ablation results with single-modality inputs.

Model	AUC
Vis only	0.8286
Text only	0.8430
Attr only	0.8271

To further examine modality interactions, we also evaluate pairwise fusion settings with equal modality weights. The visual–textual combination achieves the best pairwise result (AUC = 0.8464), outperforming both visual-only and text-only inputs. By contrast, visual–attribute obtains an AUC of 0.8280, and text–attribute reaches 0.8419, slightly below the text-only model.

These results suggest that the attribute modality is useful but weight-sensitive. Since many attributes are extracted from product fields already included in the BERT input, the attribute signal may partially overlap with the textual representation when equally weighted. Therefore, BERT should be regarded as the main contributor to the performance gain, while the GPT-4o/LLaMA-based attribute module serves as a complementary structured feature under appropriate fusion weights.

4.2.2 Weight Search Strategy

Given the performance gap across the three modalities, we conduct a systematic search over modality weights to better exploit the potential of the fusion model. We adopt a step size of 0.1 under the constraint vis + text + attr = 1. Since the unimodal ablation indicates that text performs best, the search is designed to be text-centric: we start from higher text weights and gradually adjust the visual and attribute proportions to explore the optimal fusion.

From Table 9, assigning a moderate weight to the textual modality (roughly 0.5–0.7) yields stable and comparatively higher AUC, indicating that text plays a leading role in this task. The best result occurs at *Visual*=0.3, *Text*=0.5, *Attr*=0.2 (AUC = **0.8477**). When the text weight becomes too high (e.g., 0.8), performance slightly declines relative to the mid-range settings, and increasing the attribute weight toward 0.4 tends

to reduce AUC as well. Overall, these results suggest that a text-centric yet balanced fusion—supplemented by visual cues and a moderate use of attribute signals—provides the most reliable gains.

Table 9: Validation-set AUC under different modality fusion weights. The best result is shown in bold.

Visual	Text	Attr	AUC
0.1	0.8	0.1	0.8442
0.2	0.7	0.1	0.8452
0.1	0.7	0.2	0.8441
0.3	0.6	0.1	0.8472
0.2	0.6	0.2	0.8462
0.1	0.6	0.3	0.8468
0.4	0.5	0.1	0.8470
0.3	0.5	0.2	**0.8477**
0.2	0.5	0.3	0.8465
0.1	0.5	0.4	0.8446
0.3	0.4	0.3	0.8455
0.2	0.4	0.4	0.8456

4.3 Comprehensive Performance Comparison

As shown in Table 10, our method achieves an AUC of 0.8477 on the IQON3000 dataset, outperforming all baselines by a clear margin. Relative to popularity-based approaches (POP-T: 0.6042; POP-U: 0.5951) and the random baseline (RAND: 0.5014), the gains are substantial, underscoring the potential of multimodal deep learning for personalized fashion recommendation. Even against neural baselines such as Bi-LSTM (0.6611) and BPR-DAE (0.6912), which already employ deep architectures, there remains a considerable gap when faced with complex multimodal fashion data.

Table 10: Comprehensive model comparison (AUC).

Category	Method	AUC
Baselines	POP-T	0.6042
	POP-U	0.5951
	RAND	0.5014
	Bi-LSTM [30]	0.6611
	BPR-DAE [31]	0.6912
	BPR-MF [32]	0.7867
	VBPR [9]	0.8088
	TBPR [25]	0.8102
	VTBPR [25]	0.8194
	GP-BPR [25]	0.8321
	PAI-BPR [13]	0.8368
Ours	LLM-PAI-BPR	0.8477

The importance of multimodal fusion is evident from the comparisons. VTBPR (0.8194) surpasses single-modality variants VBPR (0.8088) and TBPR (0.8102), indicating that jointly leveraging visual and textual information is crucial for capturing user preferences and item characteristics. GP-BPR (0.8321), which simultaneously models item compatibility and user-specific preference, outperforms several earlier baselines that focus on a single objective. However, PAI-BPR achieves a higher AUC of 0.8368, indicating the additional benefit of attribute-aware modeling.

Our improvements highlight the strengths of large language models in understanding and processing natural language text. By introducing BERT-based text encoding together with attribute vectors, the system captures richer, multi-dimensional characteristics of fashion items. BERT handles the semantic complexity of brand information, product descriptions, and category-related text, while the attribute vectors provide structured, discrete representations of item properties. The two components are complementary: the broad linguistic knowledge and strong semantic reasoning of LLMs, coupled with precise attribute-level signals, yield marked performance gains for fashion-text understanding and, consequently, for recommendation accuracy.

4.4 Mean Reciprocal Rank (MRR) Evaluation

To assess ranking performance in the bottom-item pairing task under practical conditions, we adopt Mean Reciprocal Rank (MRR) as a primary metric. MRR is a widely used ranking measure with values in $[0, 1]$. For each query (in our case, a pairing instance), MRR computes the reciprocal of the rank of the correct answer among all candidates and then averages this value over all queries.

A higher MRR indicates that the model tends to place the correct bottom item near the top of the list, reflecting better recommendation quality; conversely, a lower MRR suggests weaker ranking capability and a tendency to surface incorrect pairings earlier. This metric is particularly suitable for Top-k recommendation scenarios, effectively evaluating whether the system can surface an ideal match within the first few results.

4.4.1 Experimental Setup

We evaluate on a test set comprising 23,095 real top–bottom pairing instances. To compare the model's ranking ability under varying levels of difficulty, we construct multiple candidate sets per test instance with sizes $N \in \{5, 10, 20, 50\}$ (i.e., the total number of items to be ranked). Each candidate set contains exactly one positive item and $N - 1$ negatives sampled from other users' bottom items, with no duplication and excluding the original matched pair. The model ranks each candidate set, we record the reciprocal rank of the positive item, and we average over all 23,095 instances to obtain MRR under each candidate size.

- **Evaluation size:** 23,095 instances.
- **Candidates per instance (N):** 5/10/20/50 (one positive + N−1 randomly sampled negatives).
- **Positive definition:** the ground-truth top–bottom pair observed in the original interaction data.
- **Negative sampling:** bottoms randomly drawn from other users' inventories, non-duplicated and excluding the positive.

4.4.2 MRR Results and Discussion

As shown in Table 11, our method consistently outperforms the PAI-BPR baseline across all candidate sizes ($N \in \{5, 10, 20, 50\}$), indicating a clear advantage in ranking quality for the outfit matching task.

With a smaller candidate set ($N = 5$), our model attains an MRR of 0.8737, exceeding PAI-BPR's 0.8598 and demonstrating stronger ability to surface the correct pairing at the top positions. In the more challenging

scenario with $N = 50$, our method still achieves 0.5705, compared with 0.5363 for PAI-BPR, showing robust ranking capability even when the candidate pool is large.

Table 11: Model MRR under different candidate set sizes.

Model	Candidates = 5	Candidates = 10	Candidates = 20	Candidates = 50
PAI-BPR	0.8598	0.7594	0.6570	0.5363
Ours	0.8737	0.8294	0.7037	0.5705

For both models, MRR naturally decreases as the candidate size increases, since positioning the positive item near the top becomes more difficult with more distractors. Nevertheless, our method remains consistently superior and stable across settings, suggesting good Top-k behavior in practice and reliable performance in diverse candidate pools—thus supporting stronger recommendation credibility and user experience.

4.4.3 Top-K Ranking Analysis: MRR@K Evaluation

To make the evaluation more comprehensive and comparable, we compute MRR@K for all candidate set sizes ($N \in \{5, 10, 20, 50\}$). For each instance, we only consider whether the positive item appears within the top K positions; if its rank exceeds K, it is treated as a miss and its reciprocal rank (RR) is set to 0.

When selecting K, we follow the principle $K \leq N$ (where N is the candidate set size) and uniformly evaluate the following Top-K levels:

- Candidates = 5: MRR@1, MRR@3, MRR@5
- Candidates = 10: MRR@1, MRR@3, MRR@5, MRR@10
- Candidates = 20: MRR@1, MRR@3, MRR@5, MRR@10, MRR@20
- Candidates = 50: MRR@1, MRR@3, MRR@5, MRR@10, MRR@20, MRR@50

This evaluation protocol allows us to observe ranking performance at different Top-K depths and to assess the model's applicability across various recommendation scenarios.

Table 12 reports MRR@K for our method and the PAI-BPR baseline across candidate sizes $N \in \{5, 10, 20, 50\}$. Overall, our approach consistently outperforms the baseline under all settings, indicating stronger ranking capability and a greater tendency to place the correct recommendation near the top.

Table 12: MRR@K under different candidate set sizes.

Candidates N	Model	MRR@1	MRR@3	MRR@5	MRR@10	MRR@20	MRR@50
5	PAI-BPR	0.7879	0.8325	0.8598	—	—	—
	Ours	0.8136	0.8530	0.8737	—	—	—
10	PAI-BPR	0.6974	0.7275	0.7420	0.7594	—	—
	Ours	0.7323	0.7570	0.7706	0.8294	—	—
20	PAI-BPR	0.6037	0.6218	0.6312	0.6446	0.6570	—
	Ours	0.6445	0.6592	0.6676	0.6896	0.7037	—
50	PAI-BPR	0.5001	0.5089	0.5137	0.5203	0.5272	0.5363
	Ours	0.5372	0.5452	0.5495	0.5557	0.5620	0.5705

As N increases, the task naturally becomes more challenging, which is reflected in decreasing MRR values for both models; nevertheless, our method remains stable and superior across K. For instance, with $N = 50$, our model attains MRR@50 of 0.5705 vs. 0.5363 for PAI-BPR, demonstrating robust performance even with large candidate pools.

Performance gains are particularly notable at the top ranks that users most often inspect (e.g., MRR@1 and MRR@3). With $N = 10$, our MRR@1 is 0.7323 compared to 0.6974 for PAI-BPR, implying a higher likelihood that the correct item appears in the very first position—an outcome important for user experience and click-through rate.

In sum, the proposed method delivers consistent and strong results across candidate sizes and Top-K settings. The fusion of LLM-based textual understanding with attribute-level compatibility modeling effectively enhances overall recommendation accuracy and practical utility.

4.5 Qualitative Results from Real-World Tests

4.5.1 Visualization of Ranking Cases

Fig. 9 illustrates the ranking outcome for a test instance. The query consists of user `429607` with a top item `38031971` and a candidate set of five bottoms. Among the candidates, there is one ground-truth positive bottom `38385533` and four randomly sampled negatives. The model correctly ranks the positive item at the first position, indicating strong discriminative ability at ranking time. The reciprocal rank (RR) for this instance is 1.0, which contributes positively to the overall MRR; we therefore regard it as a successful case.

Figure 9: Successful ranking example for a top–bottom recommendation query. The ground-truth bottom item is ranked first among five candidate bottoms.

As shown in Fig. 10, this query corresponds to user `2462803` with a top item `11053621` and a candidate set of five bottoms. The objective is to place the positive bottom `11606201` at the top. However, the final ranking places the positive item in the second position. While this is not entirely incorrect and still acceptable in practice (with RR = 1/2 = 0.5), it falls short under the Mean Reciprocal Rank (MRR) criterion,

where hitting Top-1 is ideal. Since users typically focus on the top-most recommendation, missing the first position can degrade practical utility and user experience. We therefore categorize this example as a ranking error, suggesting room for improvement in fine-grained ranking accuracy.

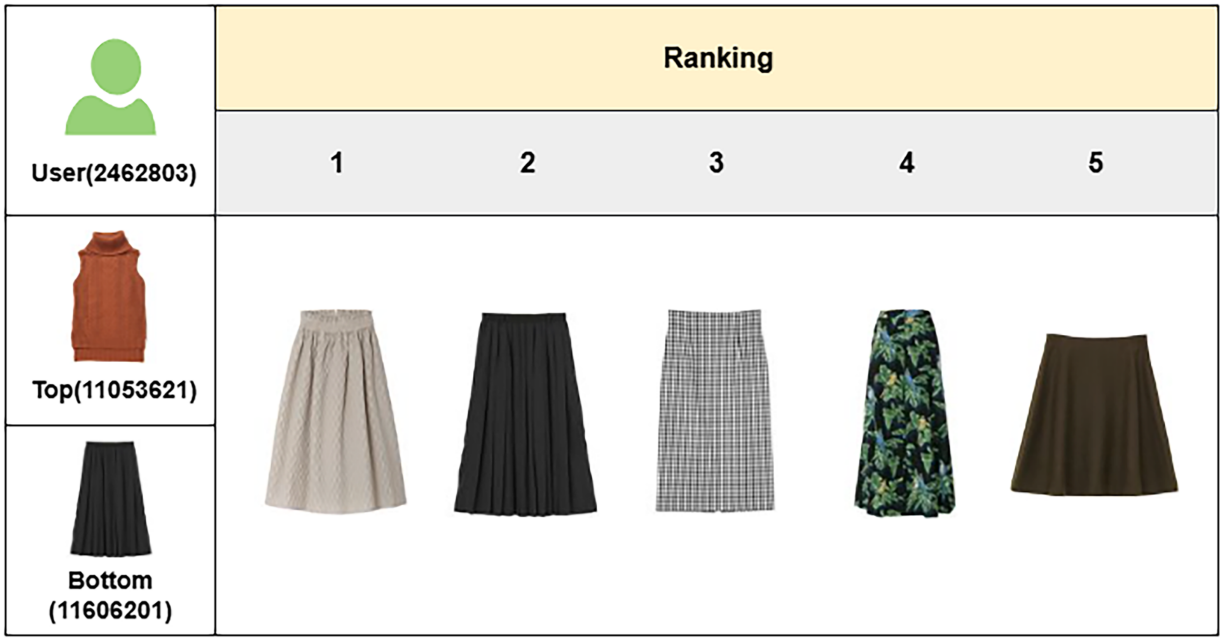

Figure 10: Failure ranking example for a top–bottom recommendation query. The ground-truth bottom item is ranked second among five candidate bottoms.

This example reflects a common failure type in top–bottom recommendation: *near-miss ranking error*. In this case, the model does not completely fail to identify a compatible item, since the positive bottom is still ranked within the top two positions. However, the model assigns a slightly higher score to another candidate, indicating that it may have difficulty distinguishing between highly similar or visually plausible alternatives. Such errors are especially important in fashion recommendation because multiple candidate bottoms may appear reasonable from a visual or stylistic perspective, but only one item is treated as the ground-truth positive pair in the evaluation protocol.

This type of error suggests that the proposed model can capture general compatibility signals, but its fine-grained ranking ability remains limited when candidate items share similar visual, textual, or attribute-level characteristics. In particular, the model may overemphasize general style compatibility while failing to sufficiently capture user-specific preference or subtle attribute differences. Therefore, this case indicates that future improvements should focus on more discriminative ranking mechanisms, hard-negative evaluation, and stronger user-preference modeling.

4.5.2 Visualization of Different User Preferences

To further verify whether the system effectively reflects different users' outfit preferences, we take the same top and generate ranked recommendations for two users with distinct style histories. The results are shown in Fig. 11.

The first user's history favors minimalist styles and denim, with a predominance of slim-fit bottoms. Accordingly, the model places denim candidates (ranks 1–3) at the top of the list, reflecting a preference for basics and practical styles.

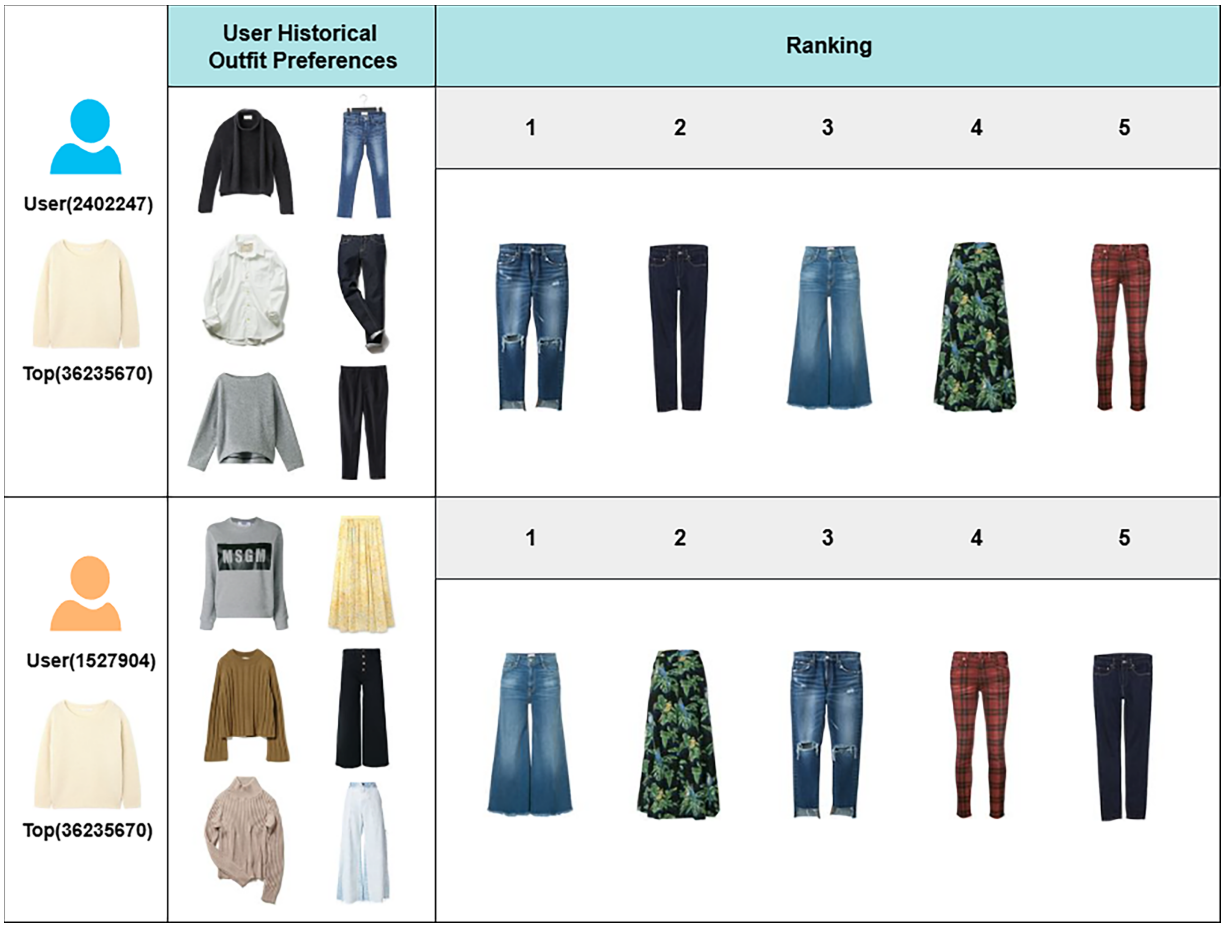

Figure 11: Examples of personalized recommendation results for two users with different historical outfit preferences. Given the same top item, the model generates different bottom-item rankings according to each user's style history.

By contrast, the second user prefers relaxed silhouettes and floral skirts, with a more casual yet design-forward style history. For the same query top (`36235670`), the model tends to recommend wide-leg pants and printed skirts (ranks 1–2), yielding a different ranking pattern from the first user's denim-oriented list.

This case demonstrates that the model adapts the ranking to users' past outfit preferences, achieving personalized recommendations. It also highlights that the model considers not only inter-item compatibility but also learns and responds to users' individual styles and tastes.

4.6 General Discussion

4.6.1 Effects of System Optimizations

This study successfully leverages the semantic understanding afforded by large language models to improve textual feature processing in the recommender. The AUC increases from 0.8368 to 0.8477, indicating that language models can materially strengthen multimodal recommendation performance. Moreover, the modality-weight analysis identifies the best combination as Text: 0.5, Visual: 0.3, Attr: 0.2, further highlighting the central role of language models in capturing outfit semantics. The MRR results likewise confirm superior ranking accuracy, suggesting tangible gains in user experience and click-through rates.

4.6.2 Computational Cost Discussion

The proposed framework introduces additional computational cost mainly during offline preprocessing. GPT-4o is used for item-level attribute extraction, LLaMA is used to construct the attribute similarity lookup table, and BERT is used to generate textual representations. These large language models are not called during online recommendation inference. In the online stage, the recommender operates on precomputed visual features, precomputed BERT textual features, and encoded attribute representations.

Based on the processed attribute files, the GPT-4o attribute extraction stage covers 128,821 top/bottom items, including 89,095 top items and 39,726 bottom items. Since the extraction process uses caching, the number of GPT-4o calls is bounded by the number of uncached items and is at most one item-level call per processed product.

For the LLaMA-based attribute similarity table, the number of calls is determined by the number of unordered attribute-value pairs within each attribute category:

$$N_{\text{sim}} = \sum_{c \in \mathcal{C}} \binom{|\mathcal{V}_c|}{2},$$

where $\mathcal{C}$ denotes the set of attribute categories and $\mathcal{V}_c$ denotes the value set of category c. Using the 1417 attribute values reported in this study, the similarity table requires:

$$\binom{257}{2} + \binom{214}{2} + \binom{254}{2} + \binom{315}{2} + \binom{69}{2} + \binom{58}{2} + \binom{216}{2} + \binom{34}{2} = 165{,}053$$

pairwise LLaMA similarity-scoring calls. This table is constructed once offline and then kept fixed during recommendation model training and inference.

The BERT textual features are also precomputed. Since each item is represented by a 768-dimensional float32 vector, storing BERT features for 128,821 items requires approximately 377 MB of storage. Therefore, the main cost-performance trade-off of the proposed framework is that it introduces additional offline preprocessing cost in exchange for improved recommendation performance. Under the adopted IQON3000 evaluation setting, the final model improves the AUC from 0.8368 in PAI-BPR to 0.8477, while avoiding online calls to GPT-4o, LLaMA, or BERT during recommendation inference.

Exact wall-clock preprocessing time, training time, inference latency, and monetary cost were not logged in the original experiments. Therefore, this study reports the preprocessing scale, call-count estimates, and memory footprint rather than reconstructed measured costs. A controlled cost-performance benchmark under the same hardware and software environment remains an important direction for future work.

4.6.3 Limitations and Future Improvements

The current evaluation is based on IQON3000, which is suitable for personalized top–bottom outfit recommendation and allows direct comparison with GP-BPR and PAI-BPR. However, since the dataset mainly contains Japanese product metadata, the results should be interpreted within this dataset setting and should not be viewed as evidence of cross-lingual or cross-cultural generalization. Future work should further evaluate the framework on English-language and culturally diverse fashion datasets with compatible user histories, item metadata, and evaluation protocols.

The experiments are conducted using the standard train–validation–test split of IQON3000 and a fixed experimental setting. Therefore, this study does not report standard deviations, confidence intervals, or statistical significance tests across multiple random seeds. The reported improvements should be interpreted as observed gains under the adopted evaluation protocol, rather than as statistically verified improvements

across repeated runs. Future evaluation should include multiple random seeds and paired statistical tests, such as bootstrap testing or paired t-tests, to better assess the robustness of AUC and MRR improvements.

The current system focuses on top–bottom pairing and does not fully support complete outfits that include shoes, bags, accessories, or other fashion items. Mathematically, the framework could be extended by representing a full outfit as a set of items and aggregating compatibility scores across multiple item pairs. For an outfit containing K items, pairwise compatibility modeling would require $O(K^2)$ comparisons among item types. A possible extension is:

$$S(O) = \sum_{p<q} \lambda_{pq} \cdot s(i_p, i_q),$$

where $O = \{i_1, i_2, \ldots, i_K\}$ denotes a full outfit, $s(i_p, i_q)$ is the compatibility score between two items, and λ_{pq} controls the importance of each item-type pair. However, generating full outfits is computationally more challenging because the candidate space grows combinatorially across item categories. A practical solution would be to first retrieve a small candidate set for each item type and then use the tri-modal compatibility model to re-rank complete outfit combinations.

From the computational perspective, the proposed framework uses pretrained language representations, which introduce additional preprocessing and memory-management costs. In our implementation, BERT textual features are precomputed and loaded only when needed, reducing GPU memory pressure during training. GPT-4o and LLaMA are also used offline for attribute extraction and similarity estimation, rather than during online recommendation. Nevertheless, this study does not provide a direct efficiency comparison with PAI-BPR in terms of training time, inference latency, or memory usage. A controlled efficiency benchmark under the same hardware and implementation settings remains future work.

Finally, as the amount of product data increases, the extracted attribute vocabulary and similarity table may also need to be updated. Although richer attributes can provide more detailed product representations, they may increase maintenance cost. Future work can explore more stable attribute schemas, vocabulary merging, and incremental update strategies.

5 Conclusion

This study proposed an LLM-enhanced multimodal framework for personalized fashion recommendation. The framework integrates visual features, contextual textual representations, and structured attribute features to improve outfit compatibility modeling and user preference prediction. Specifically, a pretrained Japanese BERT encoder was used to replace the conventional Word2Vec- and CNN-based text representation pipeline, while large language models were employed to extract fine-grained fashion attributes and estimate semantic similarities among attribute values.

Experiments on the IQON3000 dataset show that the proposed method outperforms existing baselines. The best-performing configuration achieved an AUC of 0.8477, improving upon the original PAI-BPR baseline. Ranking-based evaluations using MRR and MRR@K further indicate that the proposed model is more effective at placing compatible items near the top of the recommendation list. These results suggest that LLM-based semantic representation and attribute-level modeling provide useful complementary information for multimodal fashion recommendation.

Several limitations remain. The current framework focuses on top–bottom pairing and has not yet been extended to full outfit recommendation involving multiple item categories. In addition, LLM-based attribute extraction and similarity estimation introduce additional computational cost and may depend on the consistency of model-generated outputs. Future work will therefore investigate multi-item outfit

recommendation, more efficient attribute modeling strategies, and interpretable mechanisms for explaining recommendation results.

Acknowledgement: The authors gratefully acknowledge Chang Gung University, Taiwan, for providing the GPU computing facilities and hardware resources that made this research possible.

Funding Statement: This work was supported by the National Science and Technology Council (NSTC), Taiwan, under Grant No. 114-2221-E-182-041-MY3 and the partial financial support from Chang Gung Memorial Hospital, Linkou, under Grant No. NERPD4Q0022.

Author Contributions: Ti-Lun Miao: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization, writing—original draft. Hsien-Tsung Chang: conceptualization, methodology, supervision, project administration, funding acquisition, resources, writing—review & editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The dataset used in this study (IQON3000) is publicly available and can be accessed from the source cited in [25]. The dataset are available on GitHub at: https://github.com/hanxjing/GP-BPR.

Ethics Approval: This study did not involve human participants, animals, or identifiable personal data. Therefore, ethical approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Berti L, Giorgi F, Kasneci G. Emergent abilities in large language models: a survey. arXiv:2503.05788. 2025.
2. Elman JL. Finding structure in time. Cogn Sci. 1990;14(2):179–211. doi:10.1207/s15516709cog1402_1.
3. Kim Y. Convolutional neural networks for sentence classification. In: Moschitti A, Pang B, Daelemans W, editors. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014 Oct 25–29; Doha, Qatar. p. 1746–51.
4. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. p. 5998–6008.
5. Mikolov T, Chen K, Corrado GS, Dean J. Efficient estimation of word representations in vector space. In: Proceedings of the 2013 International Conference on Learning Representations; 2013 May 2–4; Scottsdale, AZ, USA.
6. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2019 Jun 2–7; Minneapolis, MN, USA. p. 4171–86.
7. Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. San Francisco, CA, USA; 2018. OpenAI technical report. [cited 2026 Aug 11]. Available from: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
8. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux MA, Lacroix T, et al. LLaMA: open and efficient foundation language models. arXiv:2302.13971. 2023.
9. He R, McAuley J. VBPR: visual bayesian personalized ranking from implicit feedback. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence; 2016 Feb 12–17; Phoenix, AZ, USA. p. 144–50.
10. Zadeh A, Chen M, Poria S, Cambria E, Morency LP. Tensor fusion network for multimodal sentiment analysis. In: Palmer M, Hwa R, Riedel S, editors. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing; 2017 Sep 7–11; Copenhagen, Denmark. p. 1103–14.
11. Liu Z, Shen Y, Lakshminarasimhan VB, Liang PP, Bagher Zadeh A, Morency LP. Efficient low-rank multimodal fusion with modality-specific factors. In: Gurevych I, Miyao Y, editors. Proceedings of the 56th Annual Meeting

of the Association for Computational Linguistics (Volume 1: Long Papers); 2018 Jul 15–20; Melbourne, Australia. p. 2247–56.
12. Nagrani A, Yang S, Arnab A, Jansen A, Schmid C, Sun C. Attention bottlenecks for multimodal fusion. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Vaughan JW, editors. Advances in Neural Information Processing Systems. Vol. 34, Red Hook, NY, USA: Curran Associates, Inc.; 2021. p. 14200–13.
13. Sagar D, Garg J, Kansal P, Bhalla S, Shah RR, Yu Y. PAI-BPR: personalized outfit recommendation scheme with attribute-wise interpretability. In: Proceedings of the 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM); 2020 Sep 24–26; New Delhi, India: IEEE; 2020. p. 221–30. doi:10.1109/bigmm50055.2020.00039.
14. Alshareet O, Ben Hamza A. Adaptive spectral graph wavelets for collaborative filtering. Pattern Anal Appl. 2024;27(1):10. doi:10.1007/s10044-024-01214-x.
15. Vasileva MI, Plummer BA, Dusad K, Rajpal S, Kumar R, Forsyth D. Learning type-aware embeddings for fashion compatibility. In: computer vision–ECCV 2018. Cham, Switzerland: Springer International Publishing; 2018. p. 405–21. doi:10.1007/978-3-030-01270-0_24.
16. Verma D, Gulati K, Shah RR. Addressing the cold-start problem in outfit recommendation using visual preference modelling. In: Proceedings of the 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM); 2020 Sep 24–26; New Delhi, India. p. 251–6. doi:10.1109/bigmm50055.2020.00043.
17. Salton G, McGill MJ. Introduction to modern information retrieval. Columbus, OH, USA: McGraw-Hill; 1983.
18. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. Deep contextualized word representations. In: Walker M, Ji H, Stent A, editors. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers); 2018 Jun 1–6; New Orleans, LA, USA. p. 2227–37.
19. Yu P, Xu Z, Wang J, Xu X. The application of large language models in recommendation systems. Proc SPIE. 2025;13635(4):103. doi:10.1117/12.3062544.
20. Xu W, Wu Q, Liang Z, Han J, Ning X, Shi Y, et al. SLMRec: distilling large language models into small for sequential recommendation. arXiv:2405.17890. 2024.
21. Zhang D, Yu Y, Dong J, Li C, Su D, Chu C, et al. MM-LLMs: recent advances in multimodal large language models. In: Ku LW, Martins A, Srikumar V, editors. Findings of the association for computational linguistics: ACL 2024. Bangkok, Thailand: Association for Computational Linguistics; 2024. p. 12401–30.
22. Liu Q, Zhao X, Wang Y, Wang Y, Zhang Z, Sun Y, et al. Large language model enhanced recommender systems: a survey. arXiv:2412.13432. 2024.
23. Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, et al. Measuring massive multitask language understanding. In: Proceedings of the 2021 International Conference on Learning Representations; 2021 May 4; Vienna, Austria.
24. OpenAI. GPT-4 technical report. arXiv:2303.08774. 2023.
25. Song X, Han X, Li Y, Chen J, Xu XS, Nie L. GP-BPR: personalized compatibility modeling for clothing matching. In: Proceedings of the 27th ACM International Conference on Multimedia 2019 Oct 21–25; Nice, France. p. 320–8. doi:10.1145/3343031.3350956.
26. OpenAI. GPT-4o system card. arXiv:2410.21276. 2024.
27. Meta. LLaMA 3.3 70B Instruct; 2024. Instruction-tuned 70B multilingual LLM released December 6, 2024 [cited 2026 Aug 11]. Available from: https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct.
28. Tohoku NLP Group. tohoku-nlp/bert-base-japanese model; 2023. Pretrained BERT base model for Japanese, Tohoku University NLP Group. [cited 2026 Aug 11]. Available from: https://huggingface.co/tohoku-nlp/bert-base-japanese.
29. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. BPR: Bayesian personalized ranking from implicit feedback. arXiv:1205.2618. 2012.
30. Han X, Wu Z, Jiang YG, Davis LS. Learning fashion compatibility with bidirectional LSTMs. In: Proceedings of the 25th ACM International Conference on Multimedia; 2017 Oct 23–27; Mountain View, CA, USA. p. 1078–86. doi:10.1145/3123266.3123394.

31. Song X, Feng F, Liu J, Li Z, Nie L, Ma J. NeuroStylist: neural compatibility modeling for clothing matching. In: Proceedings of the 25th ACM International Conference on Multimedia; 2017 Oct 23–27; Mountain View, CA, USA. p. 753–61. doi:10.1145/3123266.3123314.
32. Cao D, Nie L, He X, Wei X, Zhu S, Chua TS. Embedding factorization models for jointly recommending items and user generated lists. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2017 Aug 7–11; Shinjuku, Tokyo, Japan. p. 585–94. doi:10.1145/3077136.3080779.