Journal of
Theoretical and Applied
Electronic Commerce Research

MDPI

Article

Enhancing Personalized E-Commerce Recommendations Under the User–Agent–Platform Paradigm: An LLM-Driven Method

Junbiao Xu, Zhicai Zhang and Chong Zhang *

School of Computer and Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China; xujunbiao0730@163.com (J.X.); zhangzc0205@126.com (Z.Z.)
* Correspondence: zhangchong@btbu.edu.cn

Abstract

Recommendation systems are widely used in e-commerce, social media, and content distribution, yet LLM-based recommendation workflows still face recurring challenges in data completeness, sample balance, and output stability. In addition, the conventional "user-platform" structure leaves limited room for user-side mediation of exposure and preference expression. This paper presents ABP, a workflow-level extension of i^2Agent in the User–Agent–Platform setting. ABP contains three modules: Adaptive Description Enrichment (ADE), Batch-balanced Sampling Strategy (BSS), and Prompt-driven Workflow Optimization (PWO). ADE repairs missing or rigid item text with richer natural-language descriptions, BSS builds balanced comparative inputs for user profiling, and PWO strengthens multi-stage reasoning with structured output constraints. Experiments on four real-world datasets show that ABP achieves strong ranking results under the reported protocol. Across the 16 reported dataset–metric pairs, the five-run mean results of ABP show an average relative improvement of 24.62% over i^2Agent, with especially large gains on Amazon Book and Amazon Movietv. Under a fixed five-run protocol, ABP shows limited run-to-run dispersion on Amazon Book, Amazon Movietv, and Yelp, while Goodreads exhibits comparatively larger but still bounded variation. Overall, these results suggest that carefully designed workflow improvements can improve LLM-based recommendation quality in the reported setting while maintaining the agent's role as a user-side mediation layer in the User–Agent–Platform setting.

Keywords: Large Language Models; E-commerce Recommendation; User–Agent–Platform; user-side mediation

Academic Editor: Jorge Bernardino

Received: 27 May 2026
Revised: 26 June 2026
Accepted: 6 July 2026
Published: 10 July 2026

1. Introduction

Recommendation systems are central to e-commerce, social media, and online content distribution [1]. Its primary task is to filter through a vast array of candidate items to identify content that aligns most closely with users' interests and preferences. In recent years, the field has evolved from traditional recommendation algorithms [2–4] to more advanced semantic understanding. Notably, advancements in deep learning technology [5,6] laid the foundation for deep semantic capture. Building on these advances, Large Language Models (LLMs) have been introduced into recommendation systems due to their exceptional capabilities in language understanding and generation, enabling better capture of users' complex preferences and the generation of personalized recommendations [7,8]. Early exploratory works successfully demonstrated the potential of unifying recommendation tasks into language generation formats [9,10].

Despite this progress, distinct limitations persist across evolutionary stages. Traditional deep learning models, such as GRU4Rec [11], SASRec [12], and BERT4Rec [13], effectively capture sequential patterns but suffer from the "semantic gap" due to their reliance on ID-based inputs. To bridge this, researchers adopted semantic methods ranging from BM25 [14] to LLM-enhanced models like BGE-Rerank [15] and EasyRec [16]. While utilizing world knowledge, these approaches primarily function as static rankers, lacking dynamic interaction capabilities. Consequently, the field shifted towards Autonomous Agents. To understand the evolutionary lineage of our primary baseline, i^2Agent, it is essential to trace this transition chronologically: early instruction-tuned models such as P5 [9] and TALLRec [10] paved the way for conversational systems like Chat-REC [17], ultimately enabling the deployment of agentic frameworks such as ToolRec [18], and AgentCF [19], which integrate reasoning and tool usage to simulate user behavior.

Evolving from the foundational iAgent, its enhanced variant i^2Agent [20] attempts to model user preferences by introducing preliminary memory and reflection mechanisms. However, even these advanced frameworks remain vulnerable to generation instability and optimization deficiencies. Traditional ID-based sequential models capture item transitions but miss item semantics; semantic models bridge this gap but require dense text data [21]. With the advent of LLMs, various agent-based methods utilize external tools and collaborative role-playing to simulate user behavior [17–19,22]. For instance, RecMind [23] equips LLM agents with memory and tool-use capabilities, while MACRec [24] leverages multi-agent collaboration to enhance recommendation reasoning. Yet, these methods primarily focus on optimizing platform-side commercial metrics, often neglecting a user-centric perspective.

Although i^2Agent [20] introduced the "User-Agent-Platform" framing into this line of recommendation research, our experiments reveal three recurring bottlenecks in its workflow: data incompleteness, imbalanced and insufficient sampling, and workflow instability. To address them, we propose ABP (Adaptive Description Enrichment, Batch-balanced Sampling Strategy, Prompt-driven Workflow Optimization) as a workflow-level extension of the i^2Agent pipeline. ABP targets these issues through coordinated adjustments to data representation, sampling, and prompt design.

The User–Agent–Platform setting in this paper denotes a three-party interaction structure in which the user side is represented by a natural-language query field together with historical interaction and feedback signals, the agent converts these inputs into intermediate reasoning states and recommendation prompts, and the platform supplies candidate items and the surrounding recommendation environment. In this formulation, the agent serves as a user-side mediation layer within the workflow. The rerank-log-based indicators used in this study capture three observable aspects of that workflow: the User Autonomy Proxy reflects the degree of user-side influence on the final order over a shared candidate set; Long-tail Share, Exposure Gini, and related exposure statistics describe breadth and concentration in exposure allocation; and the de-identified public-data regime together with prompt-side filtering defines the privacy context examined here. These quantities are discussed as log-based descriptors in the reported setting, not as substitutes for legal, demographic, or survey-based user-rights assessments.

First, we introduce Adaptive Description Enrichment (ADE). ADE uses item titles and user reviews [25,26] to generate missing descriptions or rewrite low-quality ones, improving the textual input that downstream modules receive. Second, we propose a Batch-balanced Sampling Strategy (BSS). To address insufficient negative samples, we employ a batch-wise sampling approach where positive samples are matched with an equal number of negative samples. Inspired by contrastive learning methods [27], this enables diversified comparative analysis, thereby generating informative user profiles. Finally, we implement

Prompt-driven Workflow Optimization (PWO). Across five critical stages from candidate generation to result re-ranking, this aspect optimizes prompt design. Addressing the instability of baseline model reasoning, this module improves existing JSON output quality by introducing explicit reasoning constraints and enforcing structured protocols, thereby improving result stability and structured-output consistency.

The remainder of this paper is organized as follows. Section 2 details the ABP method and its three modules. Section 3 evaluates the proposed approach. Finally, Section 4 summarizes the key findings.

2. The ABP Method: Enhancing LLM Reasoning via Data and Workflow Optimization

This study instantiates ABP as a staged recommendation workflow that extends the i^2Agent-family pipeline to address three persistent challenges in LLM-based recommendation: data incompleteness, imbalanced and insufficient sampling, and workflow instability. Accordingly, ABP augments the baseline workflow with the ADE, BSS, and PWO modules, as shown in Figure 1. Relative to the baseline, ABP changes the information flow at three concrete points: ADE repairs sparse or rigid item text before profiling, BSS replaces isolated item-level preference elicitation with balanced group-wise contrastive input, and PWO adds explicit structured-output constraints across the multi-stage reasoning pipeline. In this sense, ABP is best understood as a concrete workflow refinement for recommendation-oriented LLM usage, not as a new standalone agent paradigm.

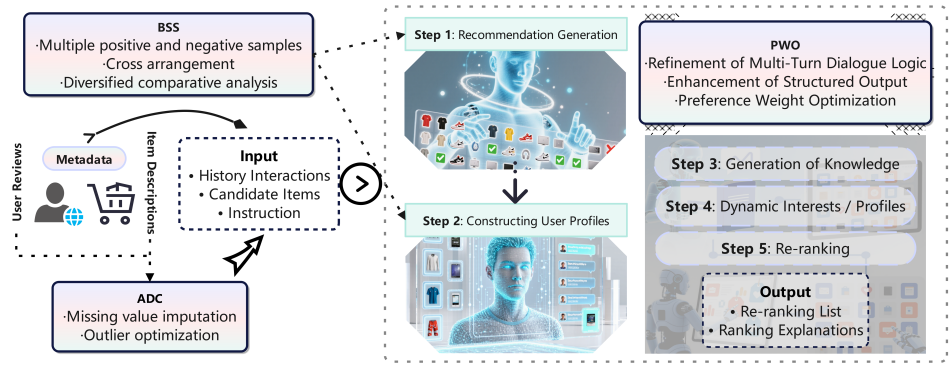

Figure 1. Module structure diagram: the overall architecture of the proposed ABP method.

Within the original workflow, three functional modules are introduced: **Adaptive Description Enrichment (ADE)**, **Batch-balanced Sampling Strategy (BSS)**, and **Prompt-driven Workflow Optimization (PWO)** [20].

These modules are designed to automatically repair and supplement missing descriptions, alleviate user profile bias caused by the scarcity of negative samples, and improve multi-turn interaction logic, respectively [20]. Working in synergy, they optimize the pipeline with respect to data quality, sampling bias, and output consistency, thereby improving recommendation performance in our experiments while staying consistent with the user-side mediation setting of i^2Agent [20].

2.1. Adaptive Description Enrichment

The Adaptive Description Enrichment (ADE) module aims to enhance the completeness and accuracy of item textual information within recommendation systems, thereby improving the quality of downstream recommendations. This module serves as a core data pre-processing component under the "User-Agent-Platform" paradigm. It employs two strategies—missing value imputation and outlier optimization—to handle missing or subpar product descriptions adaptively, as illustrated in Figure 2. Concretely, ADE

follows a detect-then-route process for each candidate item: it first checks whether the description field is absent, null, or semantically weak, and then assembles an item-centered evidence bundle from the available title, original description, and review texts. Accordingly, the upper path in Figure 2 uses "title + reviews" when the description is missing, whereas the lower path uses "title + reviews + original description" when a description exists but remains noisy, fragmented, or overly rigid in form.

For reproducibility, the routing logic is intentionally lightweight and deterministic. In our implementation, an item is sent to the missing-value branch when the description field is null-like, including cases such as `None`, `NaN`, empty strings, empty lists, or sentinel strings such as `[nan]` and `none`. An item is sent to the optimization branch when a description is formally present but is dominated by rigid metadata fragments rather than natural-language semantics, for example long concatenations of address fields, business attributes, timestamps, or sparse keyword lists. This detect-then-route rule avoids introducing an additional classifier and makes the ADE preprocessing stage directly reproducible from the raw item fields.

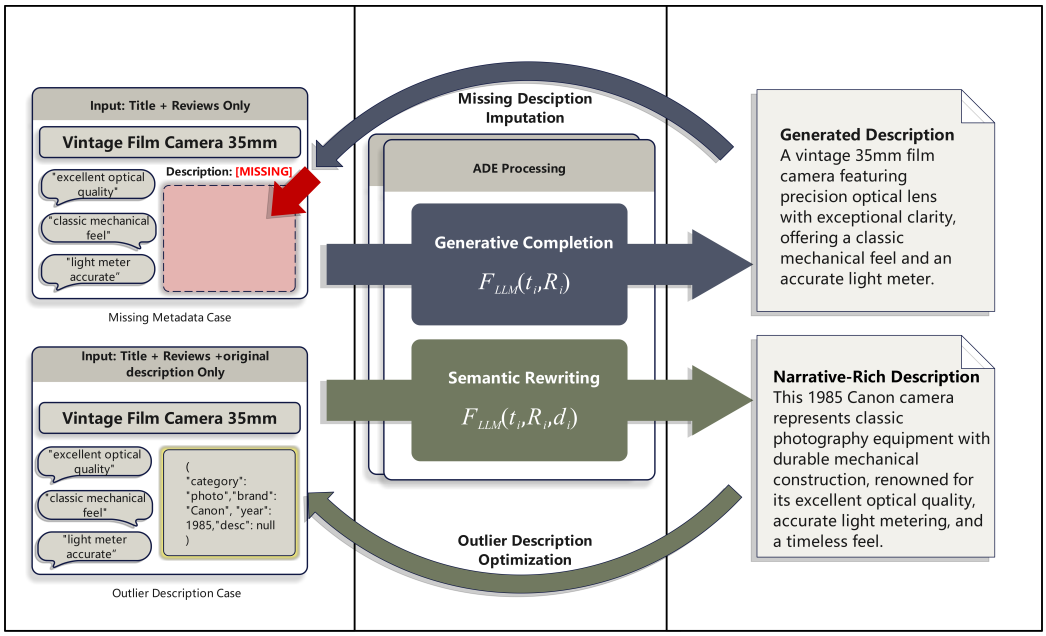

Figure 2. Principle schematic diagram: illustration of ADE Strategy 1 (Missing Description) and Strategy 2 (Outlier Description Optimization).

(1) Missing Value Imputation Strategy. For products with completely missing descriptions, we generate new high-quality descriptions using their titles and user reviews [25,26]. The implementation consists of three steps. First, the item title t_i is used to anchor the product identity and coarse semantic category. Second, the ADE module retrieves review evidence R_i associated with the item from user interaction records and organizes these review snippets into a unified prompt context. Third, the LLM synthesizes the dispersed opinion cues into a complete narrative description rather than simply copying isolated review sentences. The upper example in Figure 2 illustrates this process: when the camera item has no usable description, ADE relies on its title and review phrases such as optical quality, mechanical feel, and light-meter accuracy to generate a coherent description $\hat{d}_i$. This process can be formally represented as follows:

$$\hat{d}_i = \mathcal{F}_{LLM}(t_i, R_i) \tag{1}$$

where t_i is the item title, R_i is the set of reviews related to item i, and $\mathcal{F}_{LLM}$ represents the description generation function.

(2) Outlier Optimization Strategy. For products that have existing descriptions but are of poor quality, we propose a strategy for rewriting and supplementing information. Instead of discarding the original field, ADE treats it as a weak prior and jointly considers the product title, current description, and related reviews to generate an alternative description that is more accurate, detailed, and readable. This strategy is intended for descriptions that are formally present but practically unhelpful, such as sparse attribute lists, fragmented metadata, or boilerplate text. The lower example in Figure 2 shows this case: the camera item already contains brand/year-related metadata, yet the original description remains rigid and incomplete, so ADE rewrites it into a narrative-rich description while preserving source facts and integrating review-supported aspects. This process improves the completeness and readability of item text for downstream recommendation. The optimization process can be abstracted as follows:

$$d_i^* = \mathcal{F}_{LLM}(t_i, d_i, R_i) \tag{2}$$

where d_i is the original description and d_i^* is the optimized high-quality description.

Figure 3 corresponds to ADE Strategy 1 and visualizes the missing-description completion route: when the original description field is absent, ADE uses the item title and review evidence to generate a natural-language description for downstream recommendation. Figure 4 corresponds to ADE Strategy 2 and illustrates the description-optimization route. In the Yelp-style business data, some raw description fields are essentially structured records, such as address, categories, opening hours, and attribute dictionaries, with little natural-language content for downstream LLM reasoning. For instance, the raw field of "Sambalatte Torrefazione" is mainly a concatenation of structured metadata, whereas ADE rewrites it into a more complete natural-language business description grounded in title and review evidence. Similar behavior appears for items such as "Lotus of Siam", where rigid attribute-heavy text is reorganized into a more coherent and readable description while retaining source-supported information. These examples illustrate that ADE does not invent a new item identity; instead, it reorganizes sparse or rigid evidence into textual input that is more usable for downstream profiling and ranking. Figure 4 therefore highlights recurring rewriting patterns in business-style ADE outputs: converting structured-attribute records into coherent natural-language descriptions, reorganizing fragmented product or venue attributes into readable textual descriptions, and integrating service-related records into more complete item descriptions.

The ADE module therefore uses a simple adaptive rule: if the description is missing, it performs generative completion; if the description exists but is low-quality, it performs semantic rewriting. By integrating titles, original descriptions, and multi-source reviews, ADE generates more complete and readable descriptions while remaining grounded in the available item evidence. In the released experiment files, the enriched outputs correspond to the dataset variants with the `_description` suffix, while the original no-suffix files are retained as the non-ADE inputs for ablation. The enriched text is then passed to the subsequent BSS-based profiling and downstream ranking stages as normalized textual input, which connects the two illustrative cases in Figure 2 with the overall ABP workflow shown in Figure 1 and Algorithm 1.

Operationally, the quality-control logic of ADE is source-grounded rather than annotation-heavy. The generation prompt is restricted to the item title, the original description when available, and review evidence linked to the same item; no external catalog completion source is introduced in this stage. The detect-then-route rule is deterministic, the before/after cases in Figures 2–4 expose representative transformations and rewriting patterns, and the manuscript keeps the original non-ADE inputs as explicit ablation references. ADE is therefore presented as a reproducible rewriting/completion procedure

grounded in the available item evidence, while residual generation errors remain possible in the same way they do in other LLM-based text-synthesis settings.

Figure 3. Illustration of ADE Strategy 1: missing-description completion. When the original description field is missing, ADE uses the item title and review evidence to generate a natural-language description for downstream recommendation.

Figure 4. Illustration of ADE Strategy 2: description optimization. The top panel uses "Sambalatte Torrefazione" to show how ADE rewrites attribute-dictionary-style metadata into a coherent natural-language description; the lower panels illustrate how ADE reorganizes fragmented product or venue attributes and service records into more complete textual descriptions for downstream LLM-based recommendation.

Algorithm 1 Execution Process of ABP Method.

Require: User interaction history I_u, Candidate items C, Review set R
Ensure: Optimized Recommendation List L_{rec}
// Phase 1: Adaptive Description Enrichment (ADE)
for each item i in C **do**
  if description is missing **then**
    $d_i \leftarrow LLM_{gen}(Title_i, Reviews_i)$
  else
    $d_i^* \leftarrow LLM_{rewrite}(Title_i, d_i, Reviews_i)$
  end if
end for
// Phase 2: Batch-balanced Sampling Strategy (BSS)
$S_{pos} \leftarrow Sample(I_u, K)$
$S_{neg} \leftarrow Sample(C \setminus I_u, K)$
$Batch \leftarrow Shuffle(S_{pos} \cup S_{neg})$
$UserProfile \leftarrow LLM_{profile}(Batch)$
// Phase 3: Prompt-driven Workflow Optimization (PWO)
$Context \leftarrow OptimizePrompt(UserProfile, History)$
$L_{rec} \leftarrow LLM_{rank}(Context, C)$ with JSON Constraints
return L_{rec}

2.2. Batch-Balanced Sampling Strategy: Robust Profiling via Group-Wise Contrastive Analysis

To mitigate the bias in user profiling caused by the scarcity of negative samples or single-point sampling, we propose the Batch-balanced Sampling Strategy (BSS), as shown in Figure 5. Within the five-stage workflow, BSS spans the profile-construction and downstream knowledge-generation stages: the first dialogue stage constructs an initial contrastive selection from user interaction history, and the second dialogue stage refines the static profile that is passed to later prompts. This strategy ensures equal matching of positive and negative samples within a unified input group. Assume the historical interaction set of user u is I_u^+ and the set of candidate items is C. The negative sample set is defined as:

$$I_u^- = C \setminus I_u^+ \tag{3}$$

When sampling, first select no more than K positive samples from I_u^+:

$$S_u^+ = Top_k(I_u^+), \quad |S_u^+| \leq K \tag{4}$$

Then, we randomly sample an equal amount of negative samples from I_u^-:

$$S_u^- \leftarrow RandomSample(I_u^-, |S_u^+|) \tag{5}$$

By feeding this balanced group $G = (S_u^+ \cup S_u^-)$ into the LLM, the agent performs a group-wise contrastive analysis, identifying common preference patterns among S_u^+ while explicitly filtering out features present in S_u^-. This aligns with the principles of contrastive learning in sequential recommendation [27], adapting them for the LLM context.

To operationalize this contrastive analysis within the LLM framework, BSS implements a sequential two-stage dialogue mechanism that systematically transforms the balanced batch G into a robust static user profile P_s. In the first stage, upon feeding the balanced group G into the LLM agent, the model performs a group-wise contrastive selection that simultaneously identifies preferred items within S_u^+ and explicitly records the rejected features present in S_u^-. BSS then retains the selection outcomes, including both the selected positive items and the specific negative samples that were filtered out, in a Dialogue Memory buffer. This retention mechanism preserves the implicit decision boundaries

formed during the contrastive process, ensuring that the knowledge of why certain features were rejected remains accessible for subsequent refinement.

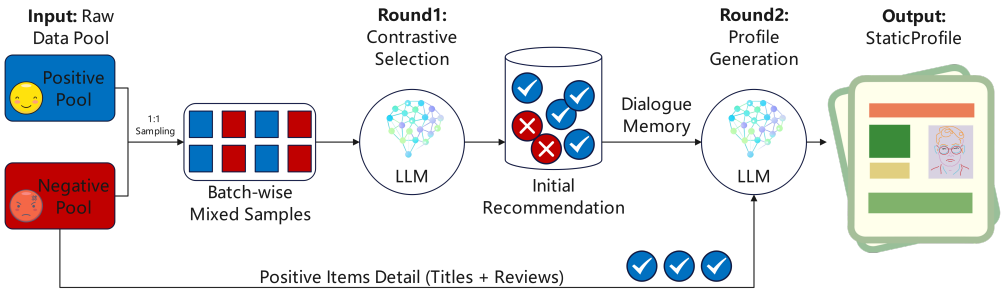

Figure 5. Principle schematic diagram: the Batch-balanced Sampling Strategy (BSS). Instead of isolated pairs, BSS constructs a balanced input batch for group-wise user profiling.

Subsequently, the second stage leverages this retained context to synthesize the static profile through a dual-input architecture. The LLM agent receives two complementary information streams: first, the Selection Context flowing from the Dialogue Memory, which encodes the explicit filtering criteria and decision boundaries derived from the initial contrastive analysis; second, the Deep Content of positive items, comprising detailed item titles and review texts R_i drawn from the original positive pool I_u^+. By integrating the contrastive boundaries indicating what the user explicitly dislikes with the semantic richness of the positive interactions, this two-stage process generates a comprehensive Static Profile P_s that captures both the core preference patterns and the clear decision boundaries necessary for accurate recommendation.

For reproducibility, the BSS prompt structure follows a fixed stage-wise template. In the first stage, the prompt presents a balanced batch containing paired positive and negative evidence and asks the agent to select preferred items while exposing rejection cues from the negative set. In the second stage, the prompt combines the retained selection context with the review-supported details of the positive items and requests a textual static profile. In our implementation, this profile is stored as natural-language bullet-style content so that it can be passed directly into subsequent ranking prompts together with the dynamic profile and candidate list. This implementation-level textual representation is also consistent with logged pipeline snippets, in which profile-like entries appear as natural-language list items describing preferred genres, favorite authors, and reading-style cues. We use these logged snippets as auditable support for the textual form of the profile, not as an independent evaluation of profile quality.The detailed two-stage profiling procedure is summarized in Algorithm 2.

Algorithm 2 Two-Stage Profiling Procedure of BSS.

Require: Positive history I_u^+, candidate pool C, review evidence R, batch budget K
Ensure: Static profile P_s
$S_u^+ \leftarrow Top_k(I_u^+, K)$
$S_u^- \leftarrow RandomSample(C \setminus I_u^+, |S_u^+|)$
$G \leftarrow Shuffle(S_u^+ \cup S_u^-)$
Construct stage-1 prompt Q_{select} from G
$(S_{\text{keep}}, B_{\text{reject}}) \leftarrow LLM_{\text{select}}(Q_{\text{select}})$
$M \leftarrow \text{DialogueMemory}(S_{\text{keep}}, B_{\text{reject}})$
Collect review-supported content R_{keep} for items in S_{keep}
Construct stage-2 prompt Q_{profile} from (M, R_{keep})
$P_s \leftarrow LLM_{\text{profile}}(Q_{\text{profile}})$
Store P_s as bullet-style textual profile entries
return P_s

2.3. Prompt-Driven Workflow Optimization: Mitigating Hallucination via Structured Constraints

The stochastic nature of LLMs often leads to unstructured outputs and hallucinations. Prompt-driven Workflow Optimization (PWO) systematically integrates structured-output constraints, staged prompting, and selective context retention into the recommendation pipeline. Table 1 illustrates the prompt-structure difference with the baseline workflow using the reranking stage as an example. Operationally, PWO can be viewed as a constrained stage-wise decoding scheme: for stage t with prompt context x_t, the model is required to emit an answer inside the stage-specific structured-output space $\mathcal{Y}_t^{json}$. The selected stage output is therefore written as

$$y_t^* = \arg \max_{y \in \mathcal{Y}_t^{json}} p_\theta(y \mid x_t), \tag{6}$$

where $\mathcal{Y}_t^{json}$ denotes the subset of candidate outputs that satisfy the required key-value schema and can be consumed by the downstream parser. This constrained view makes PWO an algorithmic workflow restriction layer for stage-wise recommendation reasoning. It leverages Chain-of-Thought (CoT) principles [28], as shown in Figure 6.

(1) Candidate Generation Optimization: PWO introduces a complete candidate set $C = \{i_1, i_2, \ldots, i_n\}$ containing positive and negative samples in an alternating arrangement. The selection process is formulated as:

$$i^* = \arg \max_{i \in C} f_\theta(i, \chi) \tag{7}$$

This enables the model to make the best global selection while maximizing relevance. Crucially, the emitted candidate-selection response must additionally satisfy a parser-validity constraint,

$$\text{Parse}_1(y_1^*) = 1, \tag{8}$$

which means that the stage-1 output can be deterministically decoded into the required schema. In practice, PWO enforces a single-key JSON format output via prompt constraints, which helps alleviate parsing errors.

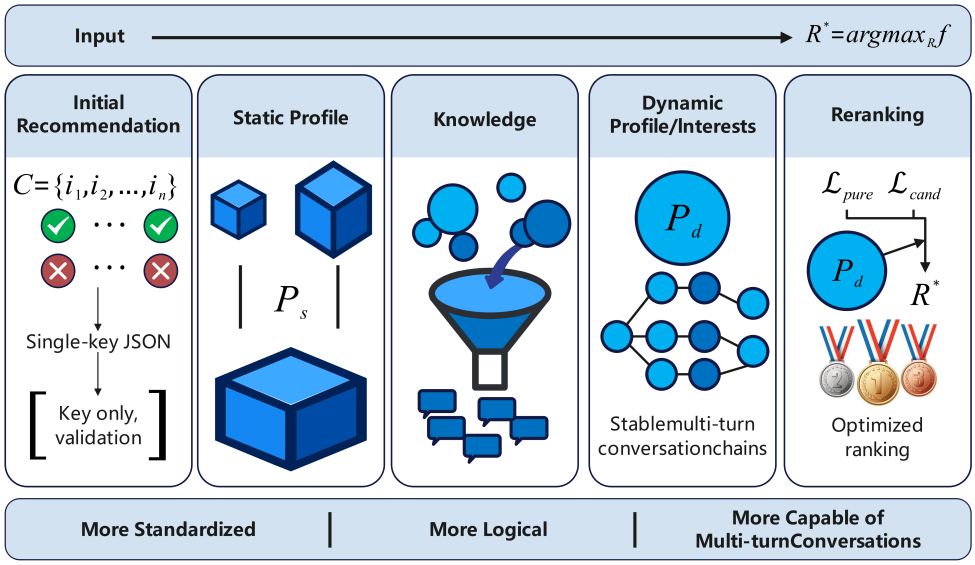

Figure 6. Principle schematic diagram: comparison of the workflow before and after PWO. The prompt-driven constraints ensure structured JSON output.

Table 1. Prompt structure comparison in the reranking stage.

Baseline Workflow Output	ABP with PWO Constraints
"Based on the information, give recommendations for the user based on the constraints."	"Based on the following information, reorganize the recommendation list. You must return one JSON object with exactly two keys: `reason` and `recommendations`."

(2) Result Re-ranking Optimization: In the final stage, PWO integrates the pure ranking list, candidate list, and dynamic interest profile P_d to obtain the optimized re-ranking list R^*:

$$R^* = \arg \max_{R \in \Pi_K(C)} f(R \mid \mathcal{L}_{pure}, \mathcal{L}_{cand}, K, P_s, P_d, I_d) \quad (9)$$

where $\Pi_K(C)$ denotes the set of length-K permutations drawn from the candidate pool. We draw upon recent findings that LLMs can act as effective zero-shot rankers [29]. Here, we implement a Selective Context Retention mechanism. Instead of retaining the full raw dialogue history (which causes "out-of-candidate hallucinations"), we restrict the model to return only the essential preference updates. Furthermore, self-consistency checks [30] are implicitly applied through the structured constraints to ensure stability and more consistent structured outputs. In the implementation, these structured-output constraints are paired with exception logging and retry control; the reported parser audit therefore measures JSON decoding stability under the current schema. In this paper, a "JSON parsing failure" specifically refers to the case where a stage output cannot be decoded by the stage-specific parser into the required key-value schema, for example because of unterminated strings, broken delimiters, or malformed field boundaries.

Across the 20 formal ABP runs retained under the fixed five-run-per-dataset protocol, the logs record 371,155 successful explicit `json_parse` events and 227 failures, corresponding to a parser-level JSON success rate of 99.94% among those explicit parse events. Most failures are concentrated in the profile-construction stage and are dominated by low-level schema or delimiter corruptions such as `missing_colon_delimiter`. This pattern is consistent with the later cross-dataset audit and indicates that the remaining instability is primarily associated with bounded structured-output issues under load.

3. Results

3.1. Datasets and Evaluation Metrics

This study utilizes datasets derived from the INSTRUCTREC benchmark [20,31], which are processed based on multiple publicly available recommendation sources. The original data includes user interaction records and rich textual information—such as titles, descriptions, and reviews—sourced from platforms like Amazon (subsets of Books and Movietv), Goodreads, and Yelp [25,26].

During the construction process, rigorous processing steps were performed on the raw data. First, records with fewer than five interactions per user or item were removed to ensure sufficient data density. Subsequently, user instructions were generated using Large Language Models (LLMs) based on user reviews and preference settings to enhance the expressive capability of interactions. Finally, instructions that could potentially reveal real item information were filtered out through LLM verification, retaining only those that do not allow for the direct inference of target items. From a privacy standpoint, this study remains within a de-identified public-data regime: the user-side records do not contain direct personal identifiers such as real names, phone numbers, ages, or home addresses. Accordingly, the privacy discussion focuses on the risks and safeguards relevant

to this setting and does not extend to formal guarantees such as differential privacy or encrypted retrieval.

Detailed statistics are provided in Table 2. By density, Amazon Book is the sparsest dataset in our study ('0.023%'), while Goodreads is the densest ('0.092%'), with Amazon Movietv and Yelp lying in between. For evaluation, we employ a leave-one-out strategy, using the last interaction for testing and the second-to-last for validation. Performance is assessed using Hit Rate (HR@K), Normalized Discounted Cumulative Gain (NDCG@K) [32], and Mean Reciprocal Rank (MRR) [33].

Hit Rate (HR@K): HR measures whether the ground-truth target item appears in the top-K recommendation list. It is defined as:

$$\text{HR@}K = \frac{1}{|U|} \sum_{u \in U} I(rank_u \leq K) \tag{10}$$

where $|U|$ is the total number of users, $rank_u$ represents the rank position of the ground-truth item for user u, and $I(\cdot)$ is the indicator function that outputs 1 if the condition is met and 0 otherwise.

Normalized Discounted Cumulative Gain (NDCG@K): NDCG evaluates the ranking quality by assigning higher scores to hits at top positions. Under the leave-one-out protocol, the ideal DCG is 1, simplifying the calculation to:

$$\text{NDCG@}K = \frac{1}{|U|} \sum_{u \in U} \frac{\ln(2)}{\ln(rank_u + 1)} \cdot I(rank_u \leq K) \tag{11}$$

Mean Reciprocal Rank (MRR): MRR calculates the average of the reciprocal ranks of the desired items across all users:

$$\text{MRR} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{rank_u} \tag{12}$$

Table 2. Statistics of the INSTRUCTREC datasets.

Dataset	Users	Items	Interactions	Density
Amazon Book	7377	120,925	207,759	0.023%
Amazon Movietv	5649	28,987	79,737	0.049%
Goodreads	11,734	57,364	618,330	0.092%
Yelp	2950	31,636	63,142	0.068%

3.2. Experiment Setup

This experiment utilized Python 3.9 as the programming language and the OpenAI API (OpenAI, San Francisco, CA, USA) to facilitate calls to large language models. GPT-4o-mini served as the core recommendation agent model for the reported recommendation experiments. The comparative results in Table 3 are presented under this unified backbone setting, and the repeated ABP runs reported throughout the paper follow the same GPT-4o-mini gateway configuration across the four datasets. This design keeps the backbone consistent across the reported comparisons and avoids conflating model differences with workflow differences.

In terms of data processing, the maximum length for user history records was constrained to 15, and the processing scale for individual data blocks was set to 1000 records. To achieve a balance between positive and negative samples, the number of positive samples was capped at 10, while the number of negative samples remained equal. Furthermore, the title information of candidate items was truncated to 50 characters, and description information was limited to 150 characters to effectively control input length.

Table 3. Comparative results on four InstructRec datasets.

Model	Amazon Book				Amazon Movietv				Goodreads				Yelp			
	H@1	**H@3**	**N@3**	**M**	**H@1**	**H@3**	**N@3**	**M**	**H@1**	**H@3**	**N@3**	**M**	**H@1**	**H@3**	**N@3**	**M**
BM25	9.92	24.48	18.21	27.00	11.29	30.27	22.09	30.04	14.25	40.34	29.01	35.40	12.85	33.08	24.34	31.85
GRU4Rec	11.00	31.41	22.53	30.10	15.80	36.85	27.63	34.36	15.36	39.52	29.08	35.41	10.94	30.67	21.88	29.70
SASRec	11.08	31.34	22.42	30.15	34.52	49.71	43.18	48.06	18.52	41.24	31.47	37.60	12.59	31.09	22.65	30.15
BERT4Rec	11.48	30.90	22.32	30.31	14.74	35.13	26.36	33.43	12.70	34.69	25.02	32.32	10.99	31.02	22.32	30.05
BGE-Rerank	25.36	45.90	37.11	42.84	25.44	47.48	38.02	43.28	17.26	40.82	30.60	36.97	33.05	55.29	45.70	49.90
ToolRec	10.56	30.60	21.88	29.77	13.84	35.67	26.20	33.21	19.06	42.79	32.61	38.44	12.07	30.92	22.83	30.21
AgentCF	14.24	34.16	25.55	32.77	25.90	49.82	39.64	44.23	21.61	46.09	35.60	40.96	13.36	34.83	25.66	32.61
EasyRec	30.70	48.87	41.09	46.14	34.96	61.30	50.15	52.98	13.94	35.38	26.11	33.27	32.41	56.31	46.04	49.86
TAIRA	41.53	55.43	49.45	54.36	45.42	64.08	56.22	59.45	25.72	48.49	38.75	43.97	46.07	**68.17**	**58.92**	60.87
iAgent	31.89	48.99	41.69	47.23	38.19	56.87	48.93	53.04	23.56	47.01	36.98	42.19	37.40	56.33	48.28	52.42
i^2Agent	35.11	53.51	45.64	50.28	46.43	65.77	57.67	60.43	30.97	56.69	45.76	49.14	39.22	57.92	49.96	53.78
ABP	**61.33**	**77.42**	**70.69**	**72.09**	**57.68**	**75.81**	**68.20**	**69.73**	**35.93**	**59.64**	**49.55**	**52.75**	**48.05**	65.72	58.27	**61.12**
Δ	74.68%	44.68%	54.89%	43.38%	24.24%	15.26%	18.26%	15.39%	16.03%	5.20%	8.28%	7.35%	22.52%	13.46%	16.63%	13.64%

Note. Bold values indicate the best result in each dataset-metric column among all compared methods.

A multi-threaded parallel processing architecture was implemented for the offline batch experiments to reduce wall-clock time per batch. This design improves throughput under concurrent API calls, but it does not by itself imply lower total computational cost, token consumption, or online serving latency. In addition, the three-retry mechanism with a 5-second interval improves completion reliability but can further increase runtime in failure cases. The reported results therefore establish recommendation-quality gains over i^2Agent, whereas efficiency-related statements are made only when the corresponding logs are available. Quantitative token usage, latency, API overhead, and monetary-cost statements are reported only when they can be traced to logs or explicit rerun summaries; when some usage fields are missing, the corresponding cost estimate is treated as a lower bound rather than as an exact full-run bill. Representative prompt structures for ADE, BSS, and PWO are summarized in the method section, while the archived implementation preserves the exact stage-wise prompts used in the reproduced runs.

3.3. Baseline Models

To evaluate the effectiveness of ABP, we compare it against representative traditional sequential recommendation methods, instruction-aware approaches, and recommendation-agent techniques.

Traditional Sequential Recommendation Methods: We selected three representative models. GRU4Rec [11] is a session-based recommendation model utilizing Gated Recurrent Units (GRUs) to capture short-term interests by modeling user interaction sequences to predict the next item. BERT4Rec [13] employs BERT's bidirectional encoding structure for masked modeling of user historical interaction sequences, effectively capturing both long- and short-term interests to enhance sequential recommendation accuracy. Additionally, SASRec [12] introduces a Self-Attention mechanism to weight user historical behaviors, thereby effectively capturing correlations at different moments within the sequence.

Instruction-aware Approaches: We included three distinct methods to assess semantic understanding. BM25 [14] serves as a classic sparse retrieval method based on term frequency and inverse document frequency, ranking items by calculating the relevance between user instructions and candidate text descriptions. BGE-Rerank [15] leverages BGE text embedding models to refine candidates through semantic similarity calculations, improving alignment with user instructions. EasyRec [16] is a lightweight instruction-aware recommendation model that combines large-scale pre-training with instruction fine-tuning to balance performance with computational efficiency.

Recommendation Agent Techniques: Finally, we compared our approach against state-of-the-art recommendation agent techniques. ToolRec [18] operates on a large language model framework, retrieving external knowledge via tool invocation while integrating it with users' historical behavior for re-ranking. AgentCF [19] integrates collaborative filtering with large language models within an agent framework, simulating user behavior through multi-agent collaboration to generate personalized recommendations. We also evaluated iAgent [20], an instruction-aware agent capable of parsing natural language commands and incorporating a self-reflection mechanism to mitigate hallucinations. Its enhanced version, i^2Agent [20], further introduces a dynamic memory mechanism that continuously optimizes user profiles and interest modeling using personalized feedback, achieving higher levels of personalization and diversity. In addition, we include TAIRA [34], a thought-augmented planning framework for LLM-powered interactive recommender agents. Among the newly screened 2025 baselines during revision, TAIRA is the only method that remains both sufficiently close to the present agentic reranking setting and stably auditable under the same shared-candidate HR@1/HR@3/NDCG@3/MRR comparison protocol used in Table 3.

The baseline set is anchored to methods that can be aligned with the same shared candidate pool used in this study. Several newer 2025–2026 LLM-centered or agentic recommenders were also screened during manuscript preparation. In the final main manuscript, we retain only those newer baselines that can be reported stably under that same shared candidate pool and the same front-rank HR@1/HR@3/NDCG@3/MRR columns used throughout this paper. Under that condition, TAIRA is the only newly screened 2025 baseline kept in Table 3. Other screened methods are discussed separately in the reviewer response, where their retained external evidence and comparability boundaries can be explained without mixing heterogeneous reporting formats or adaptation-specific evidence inside the main comparison table.

3.4. Performance Comparison

Table 3 reports the main comparison results of our proposed ABP method and eleven representative baseline models across four real-world datasets. Note that the relative improvement (Δ IMPROVE) is calculated as $\frac{M_{ABP} - M_{i^2 Agent}}{M_{i^2 Agent}} \times 100\%$, where M denotes the score of a specific evaluation metric. In the revised manuscript, the ABP row reports the mean of five formal repeated runs on each dataset, while the baseline rows remain the single-run reference results originally reported under the same shared-candidate protocol. Under this stricter reporting rule, ABP attains the highest mean scores on most reported metrics, although on Yelp the newly added TAIRA baseline still retains a small advantage on HR@3 and NDCG@3. Specifically, relative to i^2Agent, Amazon Book shows gains of 74.68% in HR@1, 54.89% in NDCG@3, and 43.38% in MRR, while Amazon Movietv shows gains of 24.24% in HR@1, 18.26% in NDCG@3, and 15.39% in MRR. The 24.62% figure is the unweighted arithmetic mean of the 16 per-cell relative improvements in the Δ row of Table 3, and it is pulled upward by the especially large gains on the sparsest dataset, Amazon Book. These results support the view that ABP improves front-rank recommendation quality in the reported setting, while the repeated-run analysis below assesses run-to-run stability across all four datasets.

Performance on Amazon Datasets. On the sparse Amazon Book dataset, ABP achieves a substantial improvement, with HR@1 reaching 61.33% compared to 41.53% for the strongest non-ABP baseline, TAIRA. This margin is consistent with the role of ADE in alleviating the semantic gap caused by sparse data. More specifically, Amazon Book is the sparsest dataset in Table 2 (0.023% density), so it is also the setting in which rigid or missing item text creates the largest semantic gap for downstream LLM reasoning. This

pattern aligns with the role of ADE and BSS: both modules are designed to compensate for weak item text, and the observed gains generally decrease as dataset density increases, although dataset-specific characteristics such as catalog heterogeneity and community writing style also influence the magnitude of improvement. Similarly, on the Amazon Movietv dataset, ABP maintains a clear lead with an HR@1 of 57.68%, remaining above both TAIRA (45.42%) and i^2Agent (46.43%).

Performance on Community-Driven Datasets. Results on the Goodreads and Yelp datasets, shown in Table 3, further support the effectiveness of our method in these settings. On Goodreads, which features dense interactions, ABP achieves an HR@1 of 35.93%, surpassing i^2Agent by 4.96 percentage points. On Yelp, ABP remains competitive rather than uniformly dominant: it leads the retained baselines on HR@1 and MRR, while TAIRA still keeps a modest advantage on HR@3 and NDCG@3. This pattern suggests that dataset-specific candidate characteristics and baseline architecture can lead to metric-dependent variation, and that ABP's uniform gains are most consistent on the sparsest catalogs where semantic repair and contrastive profiling have the clearest role.

Overall Ranking Effectiveness. Figure 7 visualizes HR@1 and MRR jointly, with marker size representing NDCG@3. In Figure 7a–c, ABP occupies the top-right frontier, indicating simultaneous gains in top-1 retrieval, ranking precision, and front-rank utility on Amazon Book, Amazon Movietv, and Goodreads. In Figure 7d, ABP remains among the strongest points and attains the highest MRR, while TAIRA stays close as the main retained competitor on Yelp. The figure therefore complements Table 3 by showing that ABP improves not only single metrics in isolation, but also the joint balance between HR@1, MRR, and NDCG@3.

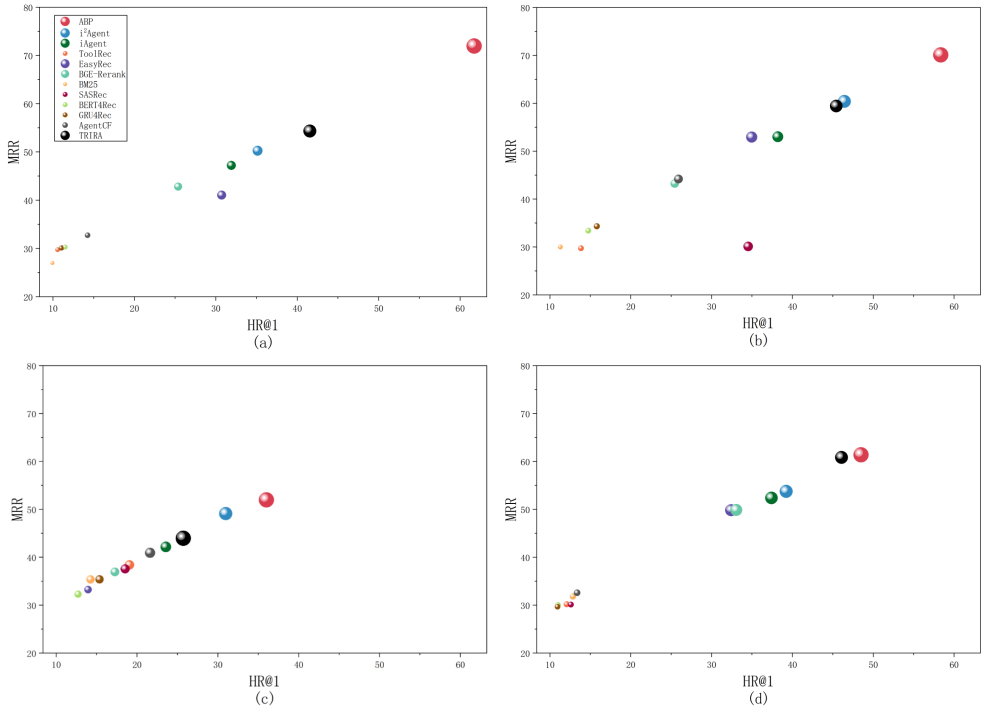

Figure 7. Performance comparison on overall ranking effectiveness. Scatter plots of MRR versus HR@1 on (**a**) Amazon Book, (**b**) Amazon Movietv, (**c**) Goodreads, and (**d**) Yelp. Marker size represents the value of NDCG@3.

Figure 8 shows a performance radar chart comparing ABP and i^2Agent. ABP covers a larger area on the averaged HR@1, HR@3, NDCG@3, and MRR axes.

Repeated-run Stability Across Four Datasets. Under the revised protocol, ABP is repeated five times on each dataset. The resulting means and standard deviations are

61.33 ± 0.18/77.42 ± 0.21/70.69 ± 0.18/72.09 ± 0.14 on Amazon Book, 57.68 ± 0.22/75.81 ± 0.15/68.20 ± 0.09/69.73 ± 0.12 on Amazon Movietv, 35.93 ± 1.01/59.64 ± 1.77/49.55 ± 1.39/52.75 ± 1.41 on Goodreads, and 48.05 ± 0.30/65.72 ± 0.28/58.27 ± 0.30/61.12 ± 0.27 on Yelp for HR@1/HR@3/NDCG@3/MRR, respectively. The repeated scores are therefore not numerically identical, but the deviations remain limited relative to the mean performance levels. The larger Goodreads variance is consistent with its heavier retry and timeout pressure in the retained logs, whereas Amazon Book, Amazon Movietv, and Yelp remain tightly clustered across reruns.

Log-backed Error Analysis Across Four Datasets. To understand residual error patterns, we aggregate the retained log evidence from all 20 formal ABP runs. After deduplication, the rerank-log audit covers 136,890 canonical rerank events. The structured logs record 371,155 successful JSON parses and 227 explicit JSON failures, corresponding to a parser success rate of 99.94% over 371,382 parse outcomes. Retry and timeout pressure is highly concentrated in Goodreads, which alone contributes 39,748 retries and 19,757 timeouts, whereas the other three datasets exhibit far lower retry and timeout pressure under the same parser-safe workflow. This cross-dataset pattern suggests that the remaining instability is driven more by dataset-specific interaction and API pressure than by systematic semantic failure of the ABP reranker.

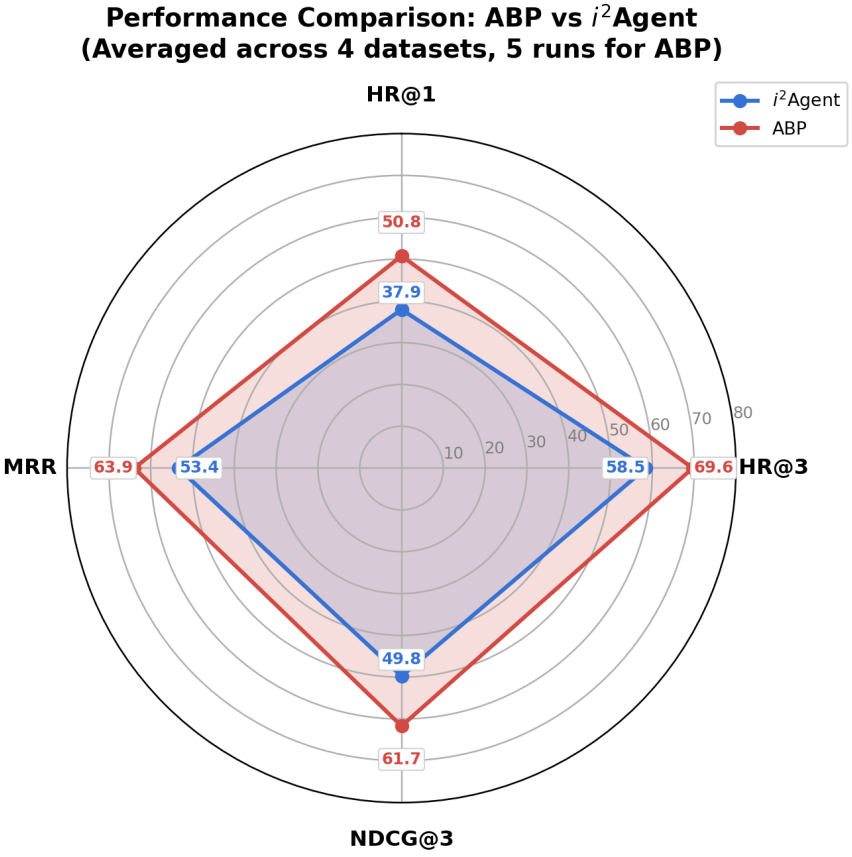

Figure 8. Performance radar chart comparing ABP and i^2Agent on HR@1, HR@3, NDCG@3, and MRR. The ABP values use the five-run mean on each dataset before averaging across the four datasets.

3.5. Repeated-Run Stability and Auxiliary Audit

Tables 4 and 5 report the five-run stability and runtime summary, respectively. In both tables, each row corresponds to one dataset and the ABP values come from the fixed five-run protocol used in this revision. This design keeps the repeated-run evidence aligned with the main comparison table instead of relying on a single-dataset audit.

The statistical formulas are as follows. For a metric value x_i from rerun i, we compute the repeated-run mean and standard deviation as $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$. When needed, the 95% confidence interval is reported as $\bar{x} \pm t_{0.975,n-1}\frac{s}{\sqrt{n}}$. In this revision, the main emphasis is therefore on the empirical clustering of the repeated-run means and standard deviations across all four datasets.

Table 4. Five-run repeated statistics of ABP across four datasets (Mean ± SD).

Dataset	HR@1	HR@3	NDCG@3	MRR
Amazon Book	61.33 ± 0.18	77.42 ± 0.21	70.69 ± 0.18	72.09 ± 0.14
Amazon Movietv	57.68 ± 0.22	75.81 ± 0.15	68.20 ± 0.09	69.73 ± 0.12
Goodreads	35.93 ± 1.01	59.64 ± 1.77	49.55 ± 1.39	52.75 ± 1.41
Yelp	48.05 ± 0.30	65.72 ± 0.28	58.27 ± 0.30	61.12 ± 0.27

Note. Values are reported as mean ± SD over five repeated runs. Minor variation remains because the end-to-end workflow still contains controlled stochasticity from multi-stage LLM generation, asynchronous API scheduling, bounded retry handling, and occasional low-level parser recovery.

We therefore interpret narrow standard deviations and stable confidence intervals, rather than bitwise identity, as the relevant evidence of stability. Goodreads exhibits the largest spread, which is consistent with its heavier timeout and retry load, whereas the other three datasets show much tighter clustering.

Table 5. Runtime summary of the five-run ABP protocol across four datasets.

Dataset	Runs	Samples/Run	Mean Runtime (s)	Avg s/Sample	Runtime Range (s)
Amazon Book	5	7377	4047.70	0.549	[3740.00, 4497.00]
Amazon Movietv	5	5649	3602.68	0.638	[3155.10, 4115.88]
Goodreads	5	11,734	7070.56	0.603	[5994.77, 8389.85]
Yelp	5	2950	1823.57	0.618	[1419.76, 2084.38]

The repeated-run stability and runtime tables are generated directly from the retained result, runtime, and step logs under the fixed five-run configuration for each dataset. This same log chain also supports the cross-dataset error analysis and the auxiliary beyond-accuracy audit reported below.

Table 6 summarizes the ABP-side repeated-run beyond-accuracy descriptors across the four datasets. Here, R_u denotes the final reranked list for user u, P_u denotes the original platform order on the same shared candidate set, and M denotes the catalog size. User Autonomy Proxy captures user-side control over reranking, Long-tail Share plus Exposure Gini characterize exposure breadth and concentration, ILD Proxy and catalog coverage describe diversity and exploration breadth, and Novelty characterizes movement away from highly frequent items. These quantities are log-based descriptors under a shared candidate set, not legal or survey-based user-rights measures. Amazon Book shows the highest long-tail share and novelty, while Goodreads and Amazon Movietv expose a broader fraction of their catalogs. The strict category-coverage ratio is not reported because the released metadata for all four datasets collapses to a single coarse category label after item-level joining, making that descriptor uninformative in the present audit.

Table 6. ABP repeated-run beyond-accuracy descriptors across four datasets.

Dataset	User Autonomy	Long-Tail Share (%)	ILD Proxy	Exposure Gini	Novelty (Bits)	Catalog Coverage (%)	Shannon Entropy
Amazon Book	0.3634	78.63	0.9349	0.7270	17.35	31.94	0.9942
Amazon Movietv	0.3709	61.45	0.9091	0.6659	15.31	48.39	0.9825
Goodreads	0.3502	64.71	0.8959	0.6037	17.28	57.98	0.9824
Yelp	0.3531	63.50	0.8556	0.7315	15.31	35.35	0.9886

Figure 9 is included as an explanatory plot. It provides context for why the sparse-data regime also exhibits the strongest ABP gain over i^2Agent, while remaining separate from the rerank-log descriptors reported in Table 6.

Among the 17 retained ABP reruns with explicit token-usage logs, the observed token cost averages USD 16.82/run (USD 10.63 input plus USD 6.19 output; range USD 7.02–30.37) under the GPT-4o-mini pricing schedule. These quantities are reconstructed directly from the available logs.

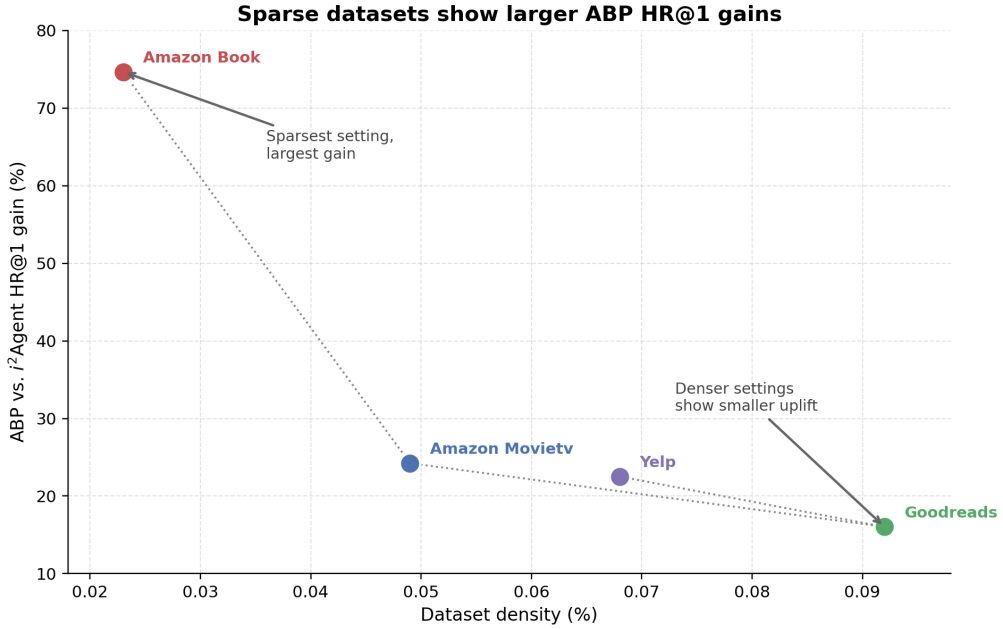

Figure 9. Dataset density versus ABP HR@1 gain over i^2Agent. The plotted HR@1 gains are computed from the five-run ABP means in Table 3. The sparsest dataset, Amazon Book, also shows the largest HR@1 gain, suggesting that semantic repair and contrastive profiling are especially helpful when the original text signal is sparse.

3.6. Ablation Study

To verify the effectiveness of each component in the proposed ABP method, we first conduct single-module ablation studies by removing ADE, BSS, and PWO individually. We also report representative dual-module ablations on Amazon Book by jointly removing ADE and BSS, and by jointly removing ADE and PWO. In Table 7, the ABP rows now use the same five-run means reported in Table 3, while the ablated rows remain representative controlled variants under the same pipeline. The remaining pairwise removal, w/o BSS + w/o PWO, is not included, so the dual-module evidence represents sparse-setting coverage rather than a complete pairwise interaction map.

Impact of ADE. The variant w/o ADE removes the Adaptive Description Enrichment module. As illustrated in Figure 10, removing ADE causes the most significant performance drop on Amazon Book and Yelp: relative to the full ABP configuration, HR@1 decreases by 10.85 and 14.08 percentage points, respectively, and the losses are similarly large on HR@3, NDCG@3, and MRR. This confirms that ADE is especially valuable when the original item text is sparse, noisy, or semantically incomplete. Furthermore, on Yelp the HR@1 metric for w/o ADE falls below that of i^2Agent (33.97% vs. 39.22%), indicating that without ADE to repair defective text, the downstream contrastive reasoning may amplify metadata noise rather than correct it. Goodreads is the main exception: the w/o ADE results, including the supplementary reinforced runs, remain higher than the full ABP mean, suggesting that in a denser community-written domain the enrichment step can occasionally over-regularize

already informative text. We therefore interpret ADE as a strongly positive module overall, but with dataset-dependent marginal utility.

Table 7 also has a supplementary statistical-reinforcement audit for the single-module w/o ADE setting, using only runs that achieved full sample coverage and no zero-tail chunks. After the supplementary reruns completed, all four datasets reached the minimum bar for formal aggregation, with two statistically reinforced runs each. The resulting means further confirm the dataset dependence of ADE: Amazon Book reaches 49.81% HR@1/64.47% HR@3/58.26% NDCG@3/61.68% MRR, Amazon Movietv reaches 56.39%/74.04%/66.63%/68.51%, Goodreads reaches 39.25%/63.85%/53.36%/56.01%, and Yelp reaches 31.37%/52.66%/43.53%/48.14%. Earlier partial and zero-output attempts are retained only as reproducibility records and are excluded from the supplementary aggregate.

Table 7. Ablation study results including representative dual-module ablations on Amazon Book. The ABP rows report the five-run mean used in this revision; ablated variants remain representative controlled runs under the same settings.

Dataset	Variant	HR@1	HR@3	NDCG@3	MRR
Amazon Book	**ABP**	**61.33**	**77.42**	**70.69**	**72.09**
	w/o ADE	50.48	62.98	57.71	61.82
	w/o BSS	58.73	75.11	68.20	69.95
	w/o PWO	56.25	76.00	67.75	69.06
	w/o ADE + w/o BSS	47.36	61.90	55.71	59.53
	w/o ADE + w/o PWO	46.34	66.29	57.88	60.42
Amazon Movietv	**ABP**	**57.68**	**75.81**	**68.20**	**69.73**
	w/o ADE	54.49	72.76	65.05	67.11
	w/o BSS	55.57	74.35	66.49	68.23
	w/o PWO	54.67	75.78	66.95	68.10
Goodreads	**ABP**	**35.93**	**59.64**	**49.55**	**52.75**
	w/o ADE	38.30	61.70	52.01	55.47
	w/o BSS	33.28	56.79	46.68	50.62
	w/o PWO	34.11	59.12	48.42	51.70
Yelp	**ABP**	**48.05**	**65.72**	**58.27**	**61.12**
	w/o ADE	33.97	53.85	44.92	49.65
	w/o BSS	48.30	65.85	58.46	61.30
	w/o PWO	47.82	67.47	59.24	61.74

Note. Bold values denote the full ABP reference configuration for each dataset block, reported as the five-run mean used in this revision; the remaining rows are representative ablated variants under the same settings.

Impact of BSS. The variant w/o BSS removes the Batch-balanced Sampling Strategy. Relative to the full ABP configuration, removing BSS reduces all four metrics on Amazon Book, Amazon Movietv, and Goodreads. On Amazon Movietv, for example, HR@1 decreases from 57.68% to 55.57% and NDCG@3 decreases from 68.20% to 66.49%, showing that BSS helps not only top-1 retrieval but also overall rank quality. The only ambiguous case is Yelp, where w/o BSS is ahead by at most 0.25 percentage points on any metric. Because these Yelp differences are smaller than the repeated-run standard deviations of ABP itself, we treat them as a statistical tie rather than as a meaningful reversal. Overall, BSS still improves cross-dataset consistency by sharpening the contrastive boundary when the candidate pool is semantically uneven.

Impact of PWO. The variant w/o PWO removes the Prompt-driven Workflow Optimization. The revised results indicate that PWO mainly stabilizes high-rank precision rather than uniformly maximizing every top-K metric. On Amazon Book and Goodreads, removing PWO lowers all four metrics. On Amazon Movietv, HR@3 is nearly tied (75.78% vs. 75.81%), but HR@1 still drops from 57.68% to 54.67% and MRR drops

from 69.73% to 68.10%. On Yelp, w/o PWO slightly increases HR@3, NDCG@3, and MRR, yet HR@1 still decreases from 48.05% to 47.82%. This pattern is consistent with the idea that unconstrained outputs can occasionally widen the effective top-K search behavior, while PWO is more useful for preserving structured-output discipline and reliable top-tier ranking precision.

Representative Dual-module Ablation on Amazon Book. The two representative dual-module rows in Table 7 directly test module interaction in the sparsest setting. Jointly removing ADE and BSS yields 47.36% HR@1, 61.90% HR@3, 55.71% NDCG@3, and 59.53% MRR, while jointly removing ADE and PWO yields 46.34% HR@1, 66.29% HR@3, 57.88% NDCG@3, and 60.42% MRR. Relative to the full ABP mean on Amazon Book, these correspond to drops of 13.97/15.52/14.98/12.56 points and 14.99/11.13/12.81/11.67 points, respectively. Both variants therefore remain clearly below the full configuration and below the related single-module variants, which is consistent with complementary gains from ADE together with the downstream BSS or PWO modules in the sparse-data regime.

The remaining pairwise removal, w/o BSS + w/o PWO, is not included; the dual-module evidence represents sparse-setting coverage rather than a complete pairwise interaction map. Figures 10 and 11 visualize only the single-module ablation patterns on the Amazon datasets and on Goodreads/Yelp, respectively; the representative dual-module rows are reported in Table 7 but are not plotted in the figures. These single-module plots make the cross-dataset differences easier to compare at a glance: ADE removal causes the sharpest decline on the sparse Amazon Book dataset and on Yelp, whereas the BSS and PWO variants are much closer to the full ABP configuration on Goodreads and Yelp, consistent with the metric-level observations reported above.

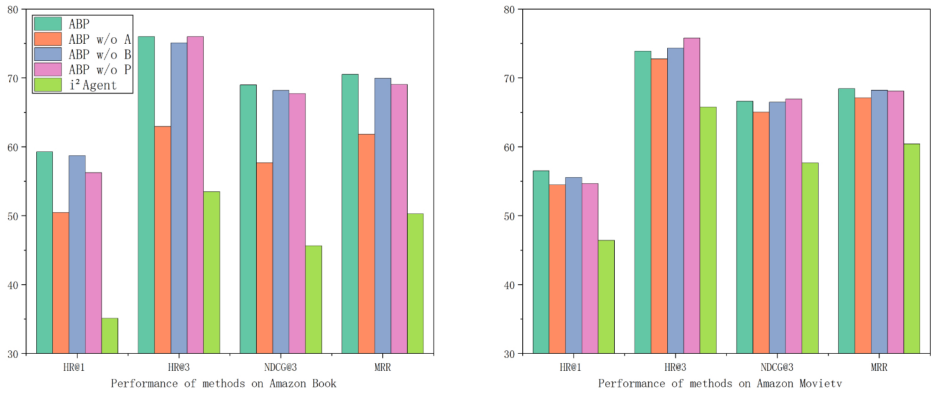

Figure 10. Metric result statistical chart—ablation on Amazon datasets.

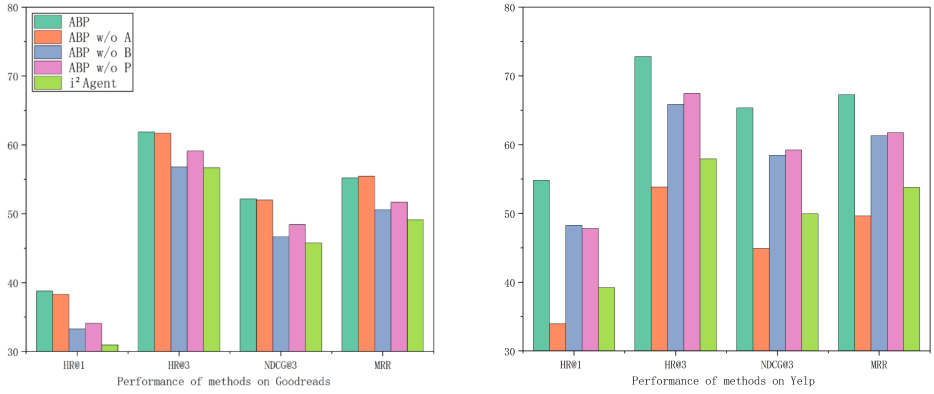

Figure 11. Metric result statistical chart—ablation on Goodreads and Yelp datasets.

4. Conclusions

This paper proposes ABP, a workflow enhancement to i^2Agent comprising the ADE, BSS, and PWO modules. Experiments on four datasets show strong ranking accuracy relative to the evaluated protocol-aligned baselines, and across the 16 reported dataset–metric pairs the five-run mean ABP results yield an average relative improvement of 24.62% over i^2Agent. The largest gains appear on the sparsest catalog (Amazon Book). Across five repeated runs on each dataset, ABP shows limited run-to-run dispersion on Amazon Book, Amazon Movietv, and Yelp, with comparatively larger but still bounded variation on Goodreads; this pattern coincides with heavier retry and timeout pressure in the retained Goodreads logs. The rerank-log audit further shows stable structured-output parsing and observable user-side reranking behavior under the reported protocol; these log-based descriptors should be interpreted as workflow-level observations rather than as formal user-rights measures or causal exposure claims.

Future work will extend the evaluation to demographic fairness with protected-attribute labels, complete the remaining pairwise ablation (w/o BSS + w/o PWO), multimodal inputs, and broader protocol-aligned baseline screening.

Author Contributions: J.X.: Conceptualization, Methodology, Software, Formal analysis, Investigation, Validation, Visualization, Writing—original draft. Z.Z.: Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Writing—review & editing. C.Z.: Conceptualization, Methodology, Resources, Supervision, Project administration, Funding acquisition, Writing—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 62201019) and the Research Foundation for Youth Scholars of Beijing Technology and Business University (Grant No. RFYS2025).

Institutional Review Board Statement: Not applicable. This study used publicly available datasets and did not involve human or animal subjects.

Informed Consent Statement: Not applicable. This study used publicly available datasets and did not involve human subjects.

Data Availability Statement: The datasets used in this study are publicly available as part of the INSTRUCTREC benchmark. Further detailed data are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this work, the authors used ChatGPT 5.4, Gemini 3.1, and Kimi 2.6 to improve English readability and to aesthetically enhance schematic figures via generative AI. For Figure 1, all core logic and framework were designed by the authors; AI only refined decorative background graphics per the authors' instructions. For Figure 3, the authors designed the complete structural layout and scientific content, while AI handled visual polish and formatting. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the content of this publication. The authors are grateful to the anonymous reviewers for their valuable comments and suggestions.

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

References

1. Zhao, W.X.; Fan, W.; Li, J.; Liu, Y.; Mei, X.; Wang, Y.; Wen, Z.; Wang, F.; Zhao, X.; Tang, J.; et al. Recommender systems in the era of large language models (LLMs). *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 6889–6907. [CrossRef]
2. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
3. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295.

4. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
6. Devlin, J.; Chang, M.-W.; 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, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
7. Wu, L.; Zheng, Z.; Qiu, Z.; Wang, H.; Gu, H.; Shen, T.; Qin, C.; Zhu, C.; Zhu, H.; Liu, Q.; et al. A survey on large language models for recommendation. *World Wide Web* **2024**, *27*, 60. [CrossRef]
8. Li, J.; Zhang, W.; Wang, T.; Xiong, G.; Lu, A.; Medioni, G. GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation. *arXiv* **2023**, arXiv:2304.03879.
9. Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; Zhang, Y. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In Proceedings of the 15th ACM International Conference on Web Search and Data Mining, Virtual, 21–25 February 2022; pp. 13–24.
10. Bao, K.; Zhang, J.; Zhang, Y.; Wang, W.; Feng, F.; He, X. TALLRec: An effective and efficient tuning framework to align large language model with recommendation. In Proceedings of the 29th ACM SIGKDD Conference, Long Beach, CA, USA, 6–10 August 2023; pp. 1007–1014.
11. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
12. Kang, W.-C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206.
13. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1441–1450.
14. Robertson, S.; Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.* **2009**, *3*, 333–389.
15. Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N.; Lian, D.; Nie, J.-Y. C-Pack: Packed resources for general Chinese embeddings. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington, DC, USA, 14–18 July 2024; pp. 641–649.
16. Ren, X.; Huang, C. EasyRec: Simple yet Effective Language Models for Recommendation. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, Suzhou, China, 4–9 November 2025.
17. Gao, Y.; Sheng, T.; Xiang, Y.; Xiong, Y.; Wang, H.; Zhang, J. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. *arXiv* **2023**, arXiv:2303.14524.
18. Zhao, Y.; Wu, J.; Wang, X.; Tang, W.; Wang, D.; de Rijke, M. Let me do it for you: Towards LLM empowered recommendation via tool learning. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington, DC, USA, 14–18 July 2024; pp. 1796–1806.
19. Zhang, J.; Hou, Y.; Xie, R.; Sun, W.; McAuley, J.; Zhao, W.X.; Lin, L.; Wen, J.-R. AgentCF: Collaborative learning with autonomous language agents for recommender systems. In Proceedings of the ACM Web Conference, Singapore, 13–17 May 2024; pp. 3679–3689.
20. Xu, W.; Shi, Y.; Liang, Z.; Ning, X.; Mei, K.; Wang, K.; Zhu, X.; Xu, M.; Zhang, Y. iAgent: LLM agent as a shield between user and recommender systems. *arXiv* **2025**, arXiv:2502.14662.
21. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.-S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
22. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, 1–5 May 2023.
23. Wang, Y.; Jiang, Z.; Chen, Z.; Yang, F.; Zhou, Y.; Cho, E.; Fan, X.; Huang, X.; Lu, Y.; Yang, Y. RecMind: Large Language Model Powered Agent For Recommendation. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, 16–21 June 2024; pp. 1–15.
24. Wang, Z.; Yu, Y.; Zheng, W.-X.; Ma, W.; Zhang, M. MACRec: A Multi-Agent Collaboration Framework for Recommendation. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington, DC, USA, 14–18 July 2024; pp. 120–130.
25. Ni, J.; Li, J.; McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 188–197.

26. Wan, M.; Misra, R.; Nakashole, N.; McAuley, J. Fine-grained spoiler detection from large-scale review corpora. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics 2019, Florence, Italy, 28 July–2 August 2019; pp. 2605–2610.
27. Xie, X.; Sun, F.; Liu, Z.; Wu, S.; Gao, J.; Zhang, J.; Ding, B.; Cui, B. Contrastive learning for sequential recommendation. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 1259–1273.
28. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In Proceedings of the Thirty-Sixth Annual Conference on Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; pp. 24824–24837.
29. Hou, Y.; Zhang, J.; Lin, Z.; Lu, H.; Xie, R.; McAuley, J.; Zhao, W.X. Large language models are zero-shot rankers for recommender systems. In Proceedings of the 46th European Conference on Information Retrieval (ECIR 2024), Glasgow, UK, 24–28 March 2024; pp. 364–378.
30. Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.V.; Chi, E.H.; Narang, S.; Chowdhery, A.; Zhou, D. Self-consistency improves chain of thought reasoning in language models. In Proceedings of the 2023 International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
31. Xu, W.; Shi, Y.; Liang, Z.; Ning, X.; Mei, K.; Wang, K.; Zhu, X.; Xu, M.; Zhang, Y. INSTRUCTREC: Instruction-Based Recommendation Datasets for User-Agent-Platform Research. Available online: https://drive.google.com/drive/folders/1-3kHU9D4IH210kSYL-m2cCgWbcY5ilBI (accessed on 27 May 2026).
32. Järvelin, K.; Kekäläinen, J. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* **2002**, *20*, 422–446. [CrossRef]
33. Craswell, N. Mean Reciprocal Rank. In *Encyclopedia of Database Systems*; Liu, L., Özsu, M.T., Eds.; Springer: Boston, MA, USA, 2009; p. 1703.
34. Yu, H.; Wu, Y.; Wang, H.; Guo, W.; Liu, Y.; Li, Y.; Ye, Y.; Du, J.; Chen, E. Thought-Augmented Planning for LLM-Powered Interactive Recommender Agent. *arXiv* **2025**, arXiv:2506.23485.