

Article

Decoupled Decision-Stage Awareness for Conversational Recommendation with Large Language Model Agents in Information Analysis

Chaoyang Li ¹, Yiwei Lu ^{2,*}, Bo Huang ², Ruopeng Yang ², Yongqi Shi ¹, Zhaoyang Gu ¹, Tianjin Ni ¹ and Yongqi Wen ¹

¹ Graduate School, National University of Defense Technology, Changsha 410073, China; lichao yang26@nudt.edu.cn (C.L.); shiyongqi17@nudt.edu.cn (Y.S.); guzhaoyang21@nudt.edu.cn (Z.G.); nitianjin@nudt.edu.cn (T.N.); wenyongqi20@nudt.edu.cn (Y.W.)

² Department of Information and Communication Command, Information Support Force Engineering University, Wuhan 430035, China; huangbo19@nudt.edu.cn (B.H.); yangruopeng17@nudt.edu.cn (R.Y.)

* Correspondence: yiweilu89@nudt.edu.cn

Abstract

Information analysis recommendation differs from conversational recommender systems (CRS) because relevance changes with the decision phase. The same event may support observation, interpretation, option selection, or action feedback, yet most large language model (LLM)-agent CRS represent dialogue state as intent and preference. This study examines whether explicit decision-stage awareness improves recommendation and whether it can be added independently of the LLM backbone. We propose Stage-Aware Conversational Recommender System (SA-CRS), a plug-in layer guided by the Observe–Orient–Decide–Act cycle. It decouples stage detection from LLM reasoning and uses detected stages to guide dialogue strategy and candidate re-ranking. We evaluate SA-CRS on an information analysis recommendation dataset from event-structured reports, using multi-turn simulated dialogues and four LLM backbones. Oracle stage injection improves Hit@5 by 3.0 percentage points (pp), showing that decision stage provides a signal beyond topic matching. With a prompt-based detector, SA-CRS improves Hit@5 by 9.0 pp on a strong backbone; with an independent Bidirectional Encoder Representations from Transformers (BERT) detector and probabilistic re-ranking, gains range from 6.5 to 15.5 pp. Negative controls with uniform or random stage signals fail to reproduce the improvements and may reduce efficiency. The results suggest that, within the evaluated single-domain information-analysis setting, decoupled decision-stage awareness is practical for decision-intensive CRS.

Keywords: conversational recommender systems; large language models; decision-stage awareness; information analysis recommendation; candidate re-ranking



Academic Editor: George Angelos Papadopoulos

Received: 2 July 2026

Revised: 18 August 2026

Accepted: 19 August 2026

Published: 21 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Conversational recommender systems (CRS) aim to infer user needs and deliver personalized recommendations through multi-turn natural language interactions, and have become an important direction in recommender systems research [1–3]. With the emergence of large language models (LLMs), tool-augmented CRS have further extended this paradigm by combining the reasoning and planning ability of LLM agents with external retrieval and ranking tools [4–6]. These systems make interactive recommendations feasible even when domain-specific conversational data are scarce. However, most existing CRS still

organize dialogue management around user intent and preference estimation. They rarely model where the user is located in an ongoing decision process, although this position can fundamentally change what information should be recommended.

This omission becomes critical in decision-intensive domains such as intelligence and information analysis. In such settings, intelligence items are not consumed for stable personal preference; their utility depends on the decision stage in which they are requested. An analyst may first need factual information for evidence collection, then analytical assessment for orientation, then response options for decision-making, and finally execution feedback for action. These stages are often implicit in the user's query rather than explicitly stated. For example, queries about the same equipment or region may ask "What entities are present in region X?", "What is the threat level?", or "What response options are available?" Although they share topical overlap, they correspond to different decision-stage requirements and therefore call for different recommendation strategies. A CRS that only tracks topic, intent, or preference can easily conflate these needs.

The CRS literature has evolved from slot-filling and end-to-end generation to LLM-agent-based tool orchestration, but its treatment of dialogue state remains largely preference- and intent-centered. Some recent studies have introduced the term "stage" [7–9]; however, these stages usually describe system-internal processes, such as reasoning workflows [8], training pipelines [9], or thinking-mode augmentation [7]. They do not explicitly represent the user's stage within a domain decision cycle. This distinction motivates the central question of this paper: Can decision-stage information serve as an independent recommendation signal, and can it be incorporated into LLM-based CRS without tying the mechanism to a specific LLM backbone?

The straightforward solution is to ask the underlying LLM to infer the stage directly. In practice, this approach is fragile for three reasons. First, stage classification quality varies across LLM backbones, especially when domain semantics are specialized. Second, when the same LLM is responsible for both stage detection and recommendation, classification errors can propagate into prompt planning, tool invocation, and candidate ranking. Third, prompt-based classifications are sensitive to prompt wording and model updates, limiting reproducibility. These issues motivate a stage-awareness mechanism that is explicit, controllable, and separable from the LLM agent's own reasoning process.

We ground this mechanism in the OODA loop (Observe–Orient–Decide–Act) [10,11], a decision-theoretic model widely used in decision-support systems [12] and situational awareness research [13]. The OODA loop provides a natural stage structure for information-analysis CRS because its four stages correspond to distinct information needs: fact acquisition, analytical interpretation, option comparison, and execution feedback. The OODA loop has previously been used to characterize the role of AI recommenders in single-turn, human-autonomy decision support [14], but to our knowledge, it has not been adopted as an explicit dialogue-state model for multi-turn conversational recommendation. To fill this gap, we propose SA-CRS, an OODA-guided stage-aware layer built on the InteRecAgent framework. SA-CRS follows a plug-and-play design—interface-level decoupling from the LLM backbone rather than domain-free transfer: it detects the user's decision stage, injects stage-specific dialogue strategies, and re-ranks candidate items by stage relevance, while leaving the underlying LLM agent and tool pipeline unchanged.

The main contributions of this work are as follows:

1. We develop SA-CRS, a plug-and-play framework that introduces OODA-guided stage awareness into LLM-agent conversational recommendation. The framework augments an existing tool-augmented CRS through three non-invasive modules: a stage detector, a dialogue strategy engine, and a candidate re-ranker.

2. We formulate decision-stage awareness as a distinct dimension of CRS dialogue state. By mapping the OODA cycle to stage-specific information needs, the framework represents the user's position in the decision process as a recommendation signal beyond topic, intent, and preference; oracle-stage experiments show a Hit@5 improvement of 3.0 pp.
3. We provide empirical evidence for both the usefulness of stage awareness and the importance of decoupled stage detection. A training-free LLM prompt detector improves Hit@5 by 9.0 pp on a strong backbone, while an independent BERT detector improves Hit@5 by 6.5–15.5 pp across four LLM backbones. Detector comparison and negative controls further show that inaccurate stage signals can reduce performance.

The remainder of this paper is organized as follows. Section 2 reviews related work, with emphasis on distinguishing the existing notion of “system-level stage design” in CRS from the “user decision-process stage awareness” proposed herein. Section 3 formulates the problem of stage-aware CRS. Section 4 presents the methodological design of SA-CRS. Section 5 reports experiments and results, including stage detector evaluation, ablation analysis, negative controls, cross-backbone validation, parameter sensitivity analysis, and detector architecture comparison. Section 5.7 discusses validity, reproducibility, and deployment considerations. Section 6 concludes the paper.

2. Related Work

2.1. Conversational Recommendation and LLM-Agent Frameworks

Early CRS framed recommendation as a process of preference elicitation through dialogue state tracking. Slot-filling methods [15] progressively narrow the candidate space by asking about item attributes, which yields interpretable policies but restricts user expression to predefined slots. Data-driven neural CRS, such as ReDial [2], KBRD [16], and UniCRS [17], further combine dialogue modeling, knowledge grounding, and recommendation learning. Their central objective, however, remains estimating user preference from conversational context, and their training usually depends on large-scale paired dialogue data.

LLMs relax this data requirement by enabling zero-shot or few-shot recommendation. Chat-REC [5] represents user preferences through structured prompts, Dai et al. [18] evaluate the zero-shot recommendation ability of ChatGPT (GPT-3.5-turbo), and MACRS [19] introduces multi-agent collaboration. Tool-augmented systems further connect LLMs with external recommendation tools: LangChain [20] and AutoGPT [21] popularized tool invocation, InteRecAgent [4] introduced a Plan-First mode and shared candidate buffer for recommendation-oriented tool coordination, RecMind [6] optimized tool selection through self-inspiring planning, and MACRec [22] used multi-agent division of labor to support recommendation decisions. Lin et al. [3] provide a broader survey of LLM-based recommender systems. These studies supply the architectural foundation for LLM-based CRS, but they primarily improve intent understanding, preference modeling, tool planning, and candidate retrieval. They do not treat the user's decision stage as a separate recommendation signal, which motivates the first contribution of this work.

2.2. Dialogue Policy and the Notion of Stage in CRS

Dialogue policy controls when a CRS should ask, recommend, refine constraints, or terminate an interaction. CP-Rec [23] performs topic prediction with learnable continuous prompts and organizes recommendations through sub-task planning. SCPR [24] conducts interactive path reasoning on a knowledge graph and learns to ask and recommend actions with reinforcement learning. CR-Walker [25] performs structured navigation over a knowledge graph with a pre-trained language model. DAHCR [26] learns hierarchical dialogue

policies under weak supervision, while UNICORN [27] jointly learns asking and recommendation policies through graph reinforcement learning. These approaches allow system behavior to change as a conversation unfolds, but the change is usually driven by dialogue history, candidate-set state, or graph structure rather than by an explicit representation of the user's decision-process stage.

Several recent CRS studies use the term "stage", but their stages describe the system's internal operation rather than the user's position in a domain decision cycle. EAR [8] employs an Estimation–Action–Reflection loop as a recurrent reasoning process. TPCRS [9] organizes model training into a tri-phase pipeline. TAIRA [7] enhances LLM-agent planning with distilled thought patterns, which characterize how the system reasons rather than what decision stage the user is in. These works are valuable for improving system-side reasoning and policy learning, yet they leave open how a CRS should adapt its dialogue strategy and candidate ranking when the same topic appears in different user decision stages. SA-CRS addresses this gap by formulating the user's decision-stage awareness as a distinct dimension of CRS dialogue state, which then drives stage-conditioned dialogue strategy and candidate ranking, corresponding to the second contribution of this work.

2.3. Decision-Stage Models and Decoupled Stage Detection

Stage-aware modeling has been studied in several decision-intensive settings. Endsley's situation awareness theory [13] distinguishes perception, comprehension, and projection, providing a cognitive account of how users transform raw information into anticipatory understanding. STAN [28] adapts recommendation strategies according to user lifecycle stages learned from behavioral data. Stage-aware learning has also been used in clinical treatment, where attention mechanisms assign different weights to treatment stages [29], and in neurodegenerative disease modeling, where mixture-of-experts models adjust pathological mechanisms across disease stages [30]. These studies show that stage information can modulate model behavior, but their stage definitions are either domain-specific, data-driven, or not designed for multi-turn conversational recommendation.

For decision-intensive information-analysis CRS, a suitable stage model should satisfy three requirements. First, stage boundaries should be grounded in domain knowledge rather than inferred only from large-scale behavioral statistics, because vertical domains often lack sufficient conversational data. Second, the model should cover the full decision cycle, including execution feedback, because recommendation may occur before, during, or after an action. Third, each stage should correspond to semantically distinct information needs so that stage labels can carry recommendation value. Endsley's model satisfies the cognitive-grounding requirement but lacks an execution-feedback stage. STAN captures behavioral evolution but derives stages from user lifecycle data rather than decision-theoretic domain knowledge. The OODA loop [10] satisfies all three requirements by mapping Observe, Orient, Decide, and Act to fact acquisition, analytical assessment, option comparison, and execution feedback. This makes it an appropriate theoretical basis for defining decision-stage awareness in information-analysis CRS.

A remaining methodological issue is how the stage should be detected. In LLM-agent CRS, the most direct approach is to let the same LLM infer the stage through prompting. This design is simple, but it couples stage quality to the capability, prompt sensitivity, and version stability of the underlying LLM. In contrast, independent neural classifiers such as BERT [31] and its Chinese pre-trained variants [32] provide deterministic and backbone-agnostic classification once trained. Existing CRS studies have not systematically examined whether decision-stage detection should be delegated to the LLM agent or decoupled from it. This work therefore evaluates both prompt-based and indepen-

dent BERT-based detectors, and empirically supports decoupled stage detection as the deployment configuration, corresponding to the third contribution of this work.

3. Problem Formulation

We consider an information-analysis scenario in which an analyst interacts with an intelligence support system through natural language to obtain decision-relevant information during an unfolding situation. The system maintains a corpus of intelligence items, each comprising a title, semantic tags (e.g., event type, involved equipment, and geographic region), a threat-level estimate, a textual summary, and structured metadata. Unlike e-commerce or media recommendation, where items are consumable products whose utility depends mainly on user preference, intelligence items are information units whose relevance is contingent on the decision context in which they are requested.

A dialogue session unfolds over multiple turns as the analyst's information needs co-evolve with the decision process. Two properties of this scenario distinguish it from standard CRS settings and directly motivate the stage-aware formulation. First, item relevance is stage-dependent: the same intelligence report may be highly valuable at one point in the decision process and irrelevant at another, so a candidate ranked highly by intent matching alone may be mismatched with the analyst's current focus. Second, the decision process is largely forward-directed: once the analyst has advanced to a later phase, queries rarely revert to earlier phases, except when a new decision cycle begins. These properties imply that the analyst's position within the decision process carries a recommendation signal beyond user intent, and that this positional signal is governed by a constrained, exploitable transition structure.

We formalize this scenario as the tuple $(\mathcal{I}, \mathcal{S}, \mathcal{T})$, where $\mathcal{I}$ denotes the item set, $\mathcal{S} = \{s_1, s_2, \dots, s_K\}$ is the set of decision stages, and $\mathcal{T} : \mathcal{S} \times \mathcal{S} \rightarrow \{0, 1\}$ encodes the stage transition constraints. Given a user's natural language input u_t at turn t and the dialogue history $\mathcal{H}_{<t} = \{(u_1, r_1), \dots, (u_{t-1}, r_{t-1})\}$, the system must accomplish three sub-tasks: (1) detect the current stage $\hat{s}_t$; (2) generate a stage-conditioned dialogue strategy π_t ; and (3) return a stage-optimized recommendation list $\mathcal{R}_t$.

We instantiate the above framework using the OODA loop: $\mathcal{S} = \{s_{obs}, s_{ori}, s_{dec}, s_{act}\}$, corresponding to four decision phases—Observe (s_{obs} , fact acquisition), Orient (s_{ori} , situation interpretation), Decide (s_{dec} , option selection), and Act (s_{act} , execution feedback).

$\mathcal{T}$ is instantiated as a forward-progression constraint: stages are permitted only to remain unchanged or advance by one step downstream within the OODA loop, e.g., Observe $\rightarrow$ Orient. Wrap-around from the Act stage back to the Observe stage is allowed upon completion of an OODA cycle, corresponding to the restart of a new decision round. All other backward jumps are prohibited.

The system's objective is to leverage stage information to improve recommendation quality—given the known or detected decision stage of the user, optimize the recommendation list to minimize the rank position of the target item while reducing the number of interaction turns required to achieve a recommendation hit.

4. Method

4.1. System Architecture Overview

SA-CRS is designed as a non-invasive stage-awareness layer for tool-augmented CRS. Its central design principle is to make the decision stage an explicit control signal while keeping stage detection separate from the LLM agent's own reasoning. The underlying InteRecAgent framework is therefore treated as a black-box recommendation agent: SA-CRS does not modify its tool definitions, planning procedure, or internal reasoning structure. Instead, it interacts with the base CRS through two external interfaces, system-prompt

injection and candidate re-ranking. This design allows the same stage-aware layer to be attached to different LLM backbones, while the independent BERT detector provides the deployment configuration for backbone-agnostic stage detection. The overall architecture comprises five layers, as shown in Figure 1:

User Interface Layer: Accepts natural language input from analysts and returns recommended intelligence along with conversational responses.

Stage-Aware Layer, the core contribution of this work: contains two stage detectors—a BERT neural detector and an LLM prompt-based detector—together with a Dialogue Strategy Engine and two stage-aware re-rankers. These sub-modules implement the three tasks defined in Section 3: sensing the stage, generating stage-conditioned dialogue strategies, and optimizing candidate ranking by stage relevance.

LLM Reasoning Layer: An LLM Agent executing in Plan-First mode that formulates tool-calling plans based on user intent.

Memory Layer: Comprises short-term dialogue memory, long-term user profiles, and a shared candidate buffer.

Tool Layer: Contains a suite of collaborative recommendation tools that form a pipeline-style recommendation process via the shared candidate buffer.

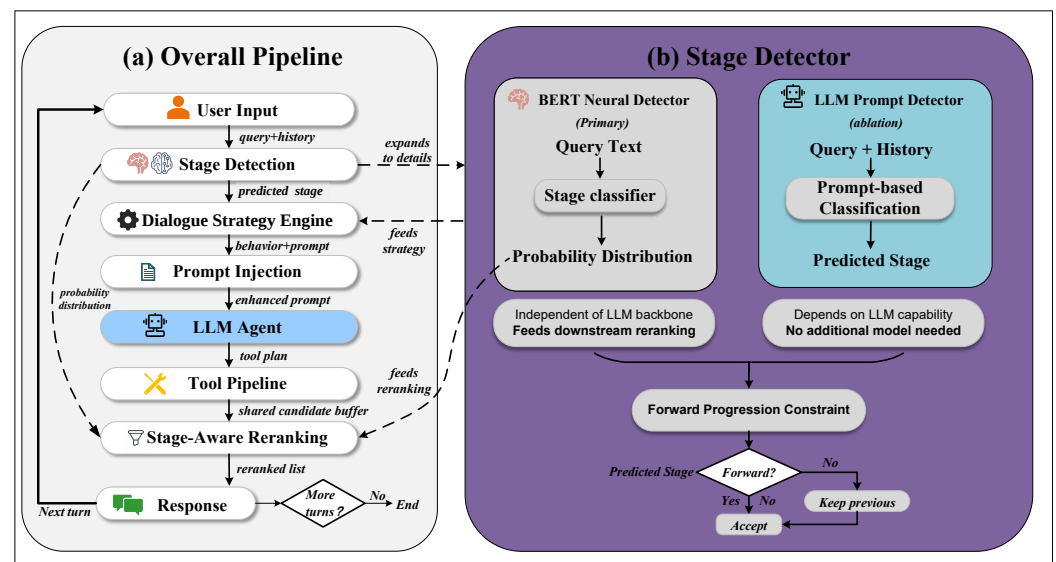


Figure 1. SA-CRS system architecture. (a) The overall pipeline processes user input through stage detection, a dialogue strategy engine, prompt injection, the LLM agent, a tool pipeline, and stage-aware re-ranking. (b) The stage detector comprises a BERT neural detector (the deployment solution) and an LLM prompt detector (an ablation counterpart), both governed by a forward-progression constraint. The stage-aware modules form a layer that augments the InteRecAgent framework through prompt injection and candidate re-ranking without modifying its internal reasoning.

Execution flow. Given the user input u_t , SA-CRS first estimates the current decision stage, then converts the detected stage into a dialogue strategy, and finally re-ranks the candidate list returned by the tool pipeline. InteRecAgent [4] is selected as the base framework for three reasons. First, its Plan-First mode generates a complete tool-calling plan before execution, allowing stage-conditioned strategy text to be injected before planning begins. Second, its modular tool pipeline provides a natural point for stage-aware re-ranking. Third, its shared candidate buffer allows candidates produced by different tools to be re-ranked uniformly after tool execution. The original pipeline consists of CandidateBuffer, LookUp, HardFilter, SoftFilter, Ranking, and Map. SA-CRS preserves this pipeline unchanged and adds stage-aware processing before planning and after candidate generation.

In this paper, “plug-and-play” refers to interface-level decoupling rather than domain-free transfer. The stage-aware layer can be attached to another tool-augmented CRS through prompt injection and candidate re-ranking interfaces, but domain-specific components—including stage definitions, strategy rules, and dimension tables—must be redesigned using target-domain knowledge. Event-structured information analysis is used here as the empirical scenario for validating the architecture.

4.2. Stage Detector

The stage detector estimates the user’s current OODA stage from the dialogue context and supplies the stage signal used by both the strategy engine and the re-ranker. SA-CRS implements two detector variants. The BERT neural detector is the deployment solution: it provides deterministic, backbone-agnostic classification through an independent lightweight classifier, so detection quality is not tied to the reasoning capability of the underlying LLM. The LLM prompt-based detector is retained as a training-free counterpart for concept validation and detector comparison. It can classify stages without additional model training, but its reliability depends on the selected LLM backbone.

The NeuralStageDetector is fine-tuned for four-way classification on `chinese-roberta-wwm-ext` [32], a Chinese pre-trained model based on BERT [31]. It takes the user’s current query text as input with a maximum sequence length of 128 and outputs a four-dimensional probability distribution $P(s_{obs}), P(s_{ori}), P(s_{dec}), P(s_{act})$. The detector is trained on a base set of 2917 OODA-annotated samples balanced across stages, generated by sampling real items from the 3200-item evaluation catalog (Section 5.1) and prompting DeepSeek-V4-Flash to produce a new natural-language query for each item–stage pair; for deployment, it is further fine-tuned on an extended pool of 9316 queries that augments this base set (Section 5.5). Deterministic inference improves reproducibility, while the probability distribution provides a continuous uncertainty signal for probabilistic candidate re-ranking. Training settings and classification performance are reported in Sections 5.1.5 and 5.5.

The LLM prompt-based detector performs stage classification directly through prompted reasoning. A naive prompt using only OODA stage names is insufficient, because the LLM may rely on surface keywords rather than the underlying information need. For example, “analyze current asset deployment” can be incorrectly assigned to Orientation because it contains the word “analyze,” although the query seeks factual deployment information and therefore belongs to Observation. To reduce such errors, we use a structured prompt containing stage semantic definitions, easily confusable boundary rules, and a disambiguation priority principle. The complete prompt is provided in Appendix A.

In the stage semantic definitions, each stage is described in terms of “what question it answers,” specifying the core task and typical query phrasing (e.g., Observation is defined as “obtaining/aggregating objective facts without interpreting their significance,” while Orientation is defined as “inferring actor intentions, threat levels, and vulnerabilities based on established facts”). To address semantic overlap across stages, the prompt includes 15 explicit boundary rules (e.g., “analyze threat level” is classified as Orientation, while “analyze current asset deployment” is classified as Observation). When a query exhibits characteristics of multiple stages, the disambiguation priority principle directs the LLM to select the most downstream stage, as downstream information needs are more specific. The LLM detector requires no additional model and can achieve high-accuracy classification on strong backbones, but classification quality is bounded by the underlying LLM capability. Weaker models may exhibit accuracy degradation or even negative gains.

Because single-turn stage predictions can be noisy, both detector variants are combined with the forward-progression constraint defined in Section 3. A predicted stage is accepted if it remains unchanged, advances by one step downstream in the OODA loop, or cycles from Action back to Observation at the start of a new decision round. Otherwise, the previous stage assignment is retained. This constraint suppresses abrupt backward jumps such as Orientation to Observation, thereby stabilizing multi-turn strategy selection. It smooths predictions over turns but cannot correct sustained misclassifications, which still propagate downstream.

4.3. Stage-Guided Dialogue Strategy Engine

The dialogue strategy engine converts the detected stage into the stage-conditioned dialogue strategy π_t defined in Section 3. Rather than replace the agent's planning, it constrains the planning context with decision-stage guidance. This is implemented through structured prompt injection, which allows SA-CRS to influence dialogue behavior without changing the agent's tool-calling logic.

4.3.1. Dialogue Behavior Space

The strategy engine defines three dialogue behaviors covering the primary interaction modes in decision-intensive information-analysis scenarios. As shown in Table 1, each behavior is activated by specific trigger semantics and produces a corresponding conversational effect.

Table 1. Three dialogue behaviors defined in SA-CRS, along with their trigger semantics and conversational effects.

Behavior	Trigger Semantics	Conversational Effect
Ask	User intent or information is insufficient	Proactively inquire about information relevant to the current stage's focus dimensions
Recommend	Intent is clear and matching candidates exist	Directly present recommendation results with accompanying explanations
Supplement	Current stage has uncovered information dimensions	Recommend while prompting about unaddressed information dimensions

4.3.2. Stage-Intent-Behavior Rule Mapping

The core of the strategy engine is a rule mapping over stage, intent, and candidate-buffer status. Given the current stage s , user intent type $i \in \{\text{clear}, \text{unclear}, \text{chitchat}\}$, candidate buffer status b , and uncovered stage dimensions, the system evaluates rules in priority order and selects the first applicable behavior. R0 defines a dedicated path for the Observation stage: when candidates already exist, the system recommends early because preliminary intelligence can help the analyst refine still-forming requirements. R3 relies on dimension tracking; when the user's input mentions a keyword associated with a stage dimension, that dimension is marked as touched, and Supplement is triggered if relevant dimensions remain uncovered. The complete rule-mapping logic is shown in Algorithm 1.

Algorithm 1 Dialogue Strategy Engine.

```

1: Input: Current stage  $s$ , user intent  $i$ , candidate buffer  $b$ , dialogue history  $\mathcal{H}$ 
2: Output: Dialogue behavior  $a \in \{\text{Ask, Recommend, Supplement}\}$ 
3: Retrieve the information dimension set  $\mathcal{D}_s$  associated with stage  $s$ 
4: Compute the uncovered dimension set  $\mathcal{D} = \mathcal{D}_s \setminus \mathcal{D}_{\text{touched}}$  based on dimensions touched in  $\mathcal{H}$ 
5: if  $s = s_{\text{obs}}$  and  $b \neq \emptyset$  then
6:   return RECOMMEND  $\triangleright$  R0: Recommend immediately when candidates exist in Observation
7: end if
8: if  $i = \text{unclear}$  then
9:   return ASK  $\triangleright$  R1: Ask when intent is unclear
10: end if
11: if  $i = \text{clear}$  and  $b \neq \emptyset$  then
12:   return RECOMMEND  $\triangleright$  R2: Recommend when intent is clear and candidates exist
13: end if
14: if  $|\mathcal{D}| > 0$  then
15:   return SUPPLEMENT  $\triangleright$  R3: Supplement when stage  $s$  has uncovered dimensions
16: end if
17: return ASK  $\triangleright$  Default: Ask by default

```

The dimension set $\mathcal{D}_s$ and recommendation preference direction in the above rules are defined by domain knowledge. As shown in Table 2, each OODA stage is associated with five information dimensions and a recommendation preference description, which the strategy engine uses to generate follow-up dimensions and injection text.

Table 2. Information dimensions and recommendation preferences for the four OODA stages.

	Observation	Orientation	Decision	Action
Dimensions	Entity positions and composition	Threat level assessment	Course of action comparison	Execution status monitoring
	Equipment types and quantities	Actor intent inference	Resource requirement estimation	Outcome assessment feedback
	Terrain and weather conditions	Vulnerability identification	Resource availability assessment	Contingency handling
	Deployment posture	Actor behavioral pattern analysis	Risk trade-off analysis	Dynamic situational changes
	Target identification and characteristics	Situational analysis and conclusions	Feasibility assessment	Follow-up action recommendations
Preference	High-precision raw observation data	Processed evaluative assessments	Decision-support intelligence	Real-time situation intelligence

The rule mapping is hand-crafted from information-analysis domain knowledge rather than learned from interaction logs. We hand-craft the rules for three reasons. First, the target domain lacks large-scale dialogue-recommendation pairs, making supervised policy learning difficult. Second, decision-support systems require traceable behavior, and rule mapping provides explicit explanations such as Decision stage detected + intent clear + candidates exist, therefore Recommend. Third, rules provide a conservative behavioral constraint when the detected stage is correct; for example, R0 prevents over-probing in the Observation stage. The rules constrain behavior conditional on the stage signal, so a wrong stage prediction can still misguide the strategy engine. The downstream soft re-ranker mitigates, but does not eliminate, this error propagation. Cross-domain deployment therefore requires redesigning the rules with target-domain knowledge.

4.3.3. Dynamic Prompt Injection

After selecting a behavior, the strategy engine generates stage-conditioned guidance text and appends it to the agent's original system prompt using the XML tag

<stage_strategy>. The injected text contains three elements: the current stage identifier, the stage-specific recommendation preference, and behavior-specific guidance instructions.

Stage identifier and recommendation preference. Each stage is associated with a set of recommendation preference descriptions encoding the type of information need for that stage. For example, the Observation stage preference is “obtain factual intelligence regarding entity positions, composition, equipment, terrain, and weather—high-precision raw observation data,” while the Orientation stage preference is “obtain analytical intelligence regarding threat levels, actor intentions, vulnerabilities, and situational conclusions—processed evaluative assessments.”

Behavior-specific guidance. The three behaviors correspond to distinct injection templates. Ask lists the top-3 focus dimensions and the recommendation preference for the current stage, guiding the user to refine the request along the appropriate analytical direction. Recommend instructs the system to present results directly with concise explanations. Supplement lists information dimensions that have not yet been touched, prompting the user to consider them alongside the recommendation. Taking the Recommend behavior in the Orientation stage as an example, the injected text is shown in Listing 1:

Listing 1. Example of the stage-strategy XML text injected for the Recommend behavior in the Orientation stage.

```
<stage_strategy>
Current decision stage: Orientation
Recommendation preference: Obtain analytical intelligence
    regarding threat levels, actor intentions,
    vulnerabilities, and~situational conclusions
Present recommendation results directly with brief explanations
</stage_strategy>
```

The complete injection templates for all four stages are provided in Appendix B. The XML wrapper separates stage guidance from the agent’s original Plan-First instructions, so the guidance stays non-invasive and removable.

4.4. Stage-Aware Candidate Re-Ranking

The strategy engine influences how the agent plans and responds, whereas candidate re-ranking influences which items are ultimately prioritized. As the final module in the stage-aware pipeline, the re-ranker adjusts candidates returned by the original tool pipeline according to their relevance to the detected decision stage [33].

4.4.1. Stage Relevance Modeling

Each intelligence item is associated with a four-dimensional stage relevance vector $(r^{obs}, r^{ori}, r^{dec}, r^{act})$, whose entries indicate the item’s relevance to the four OODA stages and take values in $[0, 1]$. These scores are annotated during data preprocessing and stored as item metadata. Given a detected stage, the re-ranker retrieves the corresponding relevance score for each candidate; given a stage probability distribution, it computes the expected stage relevance across all stages.

4.4.2. Stage-Aware Re-Ranking

The base re-ranker, StageAwareFilter, consumes a hard detected stage label $\hat{s}$ and combines each candidate’s stage relevance with its original ranking position through a single fusion weight α :

$$\text{Score}(i) = \alpha \cdot r_i^{\hat{s}} + (1 - \alpha) \cdot \text{rank_score}(i) \quad (1)$$

where $r_i^{\hat{s}}$ is the relevance of candidate i to the detected stage $\hat{s}$, $\alpha \in [0, 1]$ trades stage relevance against the ranking prior, and $\text{rank_score}(i)$ is a rank-position prior derived from the tool pipeline's raw output. We use rank position rather than raw similarity because the raw similarity scores in the candidate set were insufficiently discriminative. Since this base form consumes a hard stage label, StageAwareFilter is paired with the LLM prompt detector in the concept-validation experiments. A single $\alpha = 0.5$ is used throughout the ablation and sensitivity experiments.

4.4.3. Probabilistic Re-Ranking

The deployment re-ranker, ProbabilisticStageFilter, generalizes the base form in two ways. It replaces the hard label with the full stage probability distribution output by the BERT detector, and it uses an independent fusion weight α_s for each stage:

$$\text{Score}(i) = \sum_{s \in \mathcal{S}} P(s) \cdot [\alpha_s \cdot r_i^s + (1 - \alpha_s) \cdot \text{rank_score}(i)] \quad (2)$$

where $P(s)$ is the BERT detector's predicted probability for stage s , and r_i^s is the relevance of candidate i to stage s . Marginalizing over the distribution allows the re-ranker to account for uncertainty when the detector is not fully confident. The two filters form a single family: StageAwareFilter is recovered when $P(s)$ collapses to a one-hot vector at $\hat{s}$ and all α_s values share the same α . Because the deployment form requires a four-dimensional probability distribution, ProbabilisticStageFilter is paired with the BERT neural detector.

Per-stage independent weight parameters. Each stage uses an independent α_s parameter, determined through grid search with 5-fold cross-validation on the candidate retrieval data: Observation $\alpha_{obs} = 0.7$, Orientation $\alpha_{ori} = 0.7$, Decision $\alpha_{dec} = 0.7$, and Action $\alpha_{act} = 0.5$, with a mean validation Hit@5 of 0.893 ± 0.059 . The Action stage uses a lower weight, reflecting the convergence of its information needs with default recommendation behavior.

4.4.4. Soft Fusion Strategy

Both re-rankers use soft weighted fusion and do not impose a minimum stage-relevance threshold for hard filtering. This choice is important in information-analysis recommendation, where candidate sets are often small and overly aggressive filtering may remove useful evidence. Soft fusion preserves diversity while still promoting candidates aligned with the current decision stage.

5. Results and Discussion

This section reports and discusses the empirical evaluation of SA-CRS along five questions: (i) whether stage awareness improves end-to-end recommendation over strong tool-augmented baselines; (ii) which component of the stage-aware layer contributes to the gain; (iii) whether the gain depends on meaningful stage signals rather than arbitrary re-ranking perturbations; (iv) whether the design generalizes across LLM backbones; and (v) how detector quality and fusion weights affect downstream performance. Section 5.1 introduces the dataset, baselines, metrics, and implementation details. Sections 5.2–5.6 then present the corresponding results and analysis.

5.1. Evaluation Setup

5.1.1. Dataset

Public benchmarks for decision-stage information-analysis recommendations are scarce. We therefore constructed an evaluation corpus from CMNEE [34], which contains approximately 17,000 documents and 29,223 domain-specific events collected from public

news websites and covering eight event types: conflict, deployment, display, experiment, maneuver, injury, accident, and support. We converted these events into intelligence items and desensitized the data by redacting real entity names and unit identifiers. Each item was further annotated with a four-dimensional stage-relevance vector $(r^{obs}, r^{ori}, r^{dec}, r^{act}) \in [0, 1]$ using an LLM-assisted annotation procedure. After conversion and augmentation, the catalog contained 3200 items, each with a title, tags, a threat level, a textual summary, and stage-relevance metadata. Because no real user-item interaction logs are available in this domain, the evaluation follows a zero-shot, tool-augmented CRS setting rather than a supervised recommendation setting.

For evaluation, we constructed 400 multi-turn dialogues, balanced across the four OODA stages with 100 dialogues per stage. The 400 first-turn queries serve as the stage-detector test set. A 200-dialogue subset is used for end-to-end recommendation evaluation, with per-stage counts of 52 Observe, 49 Orient, 48 Decide, and 51 Act dialogues. For each dialogue, a user simulator generated an initial query from the target intelligence attributes and refined the request over at most five turns, allowing the evaluation to measure both recommendation accuracy and interaction efficiency.

5.1.2. Baselines

We follow the baseline design of InteRecAgent [4]. Following the evaluation convention of LLM-agent conversational recommendation [19,35], the main comparison is drawn among conversational recommenders that share the multi-turn, tool-orchestrating agent paradigm of SA-CRS. The baselines below represent increasing levels of recommendation capability:

- **Random:** samples k items uniformly from the entire intelligence corpus.
- **Popularity sampler:** samples k items with item popularity (`visited_num`) as the sampling weight. As CMNEE contains no viewing logs, `visited_num` is a synthetic uniform random integer in $[10, 499]$ assigned during corpus construction, so Popularity here is a popularity-weighted sampler rather than a true popularity baseline.
- **DeepSeek-V4-Flash (LLM-only) [4]:** DeepSeek-V4-Flash is accessed directly via API as a conversational recommender with no retrieval or recommender tool. Unlike InteRecAgent's GPT-3.5/GPT-4 direct-LLM baselines, which recommend from the LLM's internalized knowledge of real-world items, our intelligence items are synthetic and unknown to the LLM; we therefore supply the full catalog as a numbered list of all 3200 item titles in a fixed system prompt shared across all runs of this baseline, and the LLM selects k items directly from it.
- **Chat-REC [5]:** a conversational recommendation agent that retrieves candidates with a text-embedding tool and then selects k items with an LLM. Following the original design, we use BAAI/bge-small-zh-v1.5 for retrieval (replacing OpenAI text-embedding-ada-002, which is unavailable for our Chinese corpus) and DeepSeek-V4-Flash as the selection LLM.

Among these baselines, the two query-agnostic samplers (Random and the Popularity sampler) provide lower-bound references. The LLM-only and Chat-REC baselines use DeepSeek-V4-Flash, which is also used by the InteRecAgent (vanilla) backbone in the main experiment. This control keeps the LLM backbone fixed and attributes performance differences primarily to retrieval, tool orchestration, and the proposed stage-aware layer.

In addition to external baselines, we include **InteRecAgent (vanilla) [4]**, a Plan-First tool-augmented CRS without the stage-aware layer. This within-framework baseline is the most direct comparison because SA-CRS is obtained by attaching the proposed layer to the same backbone. Supervised CRS methods [16,17,25] are not included because they

require user–item interaction records or paired dialogue–recommendation data, which are unavailable in this zero-shot information-analysis setting.

Beyond the CRS baselines above, we additionally report two direct retrieval references Revised. The premature citation has been removed; the retrieval-reference table is now first cited in the Results section, after the main-results table, restoring numerical order. The two case-study tables are now cited in the text as well. **BM25** [36] (lexical retrieval with jieba tokenization, $k_1 = 1.5$, $b = 0.75$) and **bge-m3-nostage** (BAAI/bge-m3 dense retrieval [37] over the top-150 candidates, with the OODA stage modules removed). We initially indexed all item fields (title, tags, region, description) for both retrievers, but recognized that this inflated their scores: the goal-oriented simulator discloses the target’s tag and region values in its queries, so indexing those fields let the retriever match the disclosed answer directly rather than perform document-content retrieval. We therefore corrected the setup to document-text indexing (title and description only)—the standard retrieval configuration, which indexes an item’s body text and excludes curated metadata columns—isolating content-matching ability from the simulator’s attribute disclosure. These retrieval references are reported separately from the CRS comparison because, unlike the conversational agents, they are stateless retrievers: each turn they perform a single query–item match, holding no dialogue state and applying no selection policy or re-ranking. A standalone retriever and a conversational agent thus differ in task definition (single-shot matching vs. multi-turn interactive recommendation), not in algorithmic quality; the retrieval references characterize single-shot retrieval accuracy under the goal-oriented protocol rather than conversational recommendation ability.

5.1.3. Evaluation Metrics

Because the main research question concerns item discovery, we evaluate recommendation effectiveness rather than open-ended response quality. For the multi-turn user-simulator protocol, we report Hit@K and AT@5. Hit@K ($K = 1, 3, 5$) is the proportion of dialogues in which the target intelligence item appears in the recommendation list within K turns (following the multi-turn success-within- K -turns convention of InteRecAgent [4], not a top- K rank cutoff). AT@5 is the average number of turns required for a successful recommendation within the five-turn budget; unsuccessful dialogues are assigned a turn count of $K + 1 = 6$. Higher Hit@K indicates better recommendation accuracy, while lower AT@5 indicates greater interaction efficiency.

5.1.4. User Simulator Protocol

The multi-turn dialogues are generated by an LLM-based user simulator (DeepSeek-V4-Flash, temperature 0.8, at most 256 tokens per turn). The simulator follows the goal-oriented paradigm standard for conversational-recommendation evaluation, inherited from InteRecAgent [4]. At the start of each dialogue, the simulator receives the target item’s attributes. It then emits queries referencing a few attribute values (tags, region, threat level) to steer the recommender toward the target, without naming its title. In later turns, it refines the request from the agent’s replies. A dialog ends when the simulator emits <END> or the five-turn budget is reached. The recommender returns $k = 5$ items per turn, following InteRecAgent, so the user sees up to 25 candidates per dialogue. All methods are evaluated under this same simulator. The complete prompt appears in Appendix C.

5.1.5. Implementation Details

Tables 3 and 4 summarize the system configuration and the BERT detector training parameters. In the main and ablation experiments, the recommendation agent uses DeepSeek-V4-Flash and the LLM prompt detector uses DeepSeek-V4-Flash. In the cross-backbone experiment (Section 5.4), the stage signal is provided by the BERT neural detector

and the recommendation backbone is varied across four LLMs from different providers. The negative-control experiment (Section 5.3) is conducted on GLM-4.7-flash using the same 200 test dialogues and the same ProbabilisticStageFilter configuration as in the cross-backbone experiment; only the source of the stage signal is changed. The HardFilter (SQL) tool uses a candidate-floor setting throughout the main experiments: when an SQL query returns fewer than $k = 5$ items, it progressively drops AND conditions until at least k candidates are obtained, modeling the realistic deployment case in which a retriever returns a small set of partly noisy candidates that the recommender must then order.

We use two detector–re-ranker pairings for different experimental purposes. In the main, ablation, and sensitivity experiments (Sections 5.2, 5.3 and 5.6), the training-free LLM prompt detector is paired with StageAwareFilter using a single fusion weight $\alpha = 0.5$ on the strong DeepSeek-V4-Flash backbone. This setting isolates the effect of adding stage awareness to a capable LLM-agent CRS. In the cross-backbone and control experiments (Sections 5.3 and 5.4), the BERT neural detector is paired with ProbabilisticStageFilter, whose per-stage weights α_s are selected by 5-fold cross-validation. This deployment-oriented setting tests whether stage awareness remains useful when the recommendation backbone changes.

Table 3. SA-CRS system configuration parameters.

Parameter	Value
Recommendation Agent	DeepSeek-V4-Flash
LLM stage detector	DeepSeek-V4-Flash
LLM detector temperature	0
Recommendation generation temperature	0.5
Dialogue history window	500 characters
Ablation re-ranking	StageAwareFilter, $\alpha = 0.5$
Cross-backbone re-ranking	ProbabilisticStageFilter, per-stage independent α

Table 4. BERT stage detector training parameters.

Parameter	Value
Pre-trained model	chinese-roberta-wwm-ext
Hidden layers	12
Hidden size	768
Attention heads	12
Number of classes	4
Max sequence length	128
Learning rate	2×10^{-5}
Batch size	16
Max training epochs	20
Early stopping	patience = 3
Validation	5-fold stratified k -fold CV

5.2. Overall Recommendation Results

5.2.1. Recommendation Performance

We first evaluate whether SA-CRS improves end-to-end recommendation quality. All methods are tested on the same 200 multi-turn dialogues with Hit@1, Hit@3, Hit@5, and AT@5. The comparison includes four external baselines and the InteRecAgent (vanilla) backbone. SA-CRS is evaluated in the concept-validation configuration, namely, the LLM prompt detector paired with StageAwareFilter.

Table 5 reports the results. SA-CRS achieves the best performance among the conversational agents on Hit@3, Hit@5, and AT@5 (Hit@1 0.365, Hit@3 0.785, Hit@5 0.885, AT@5 2.42); on Hit@1, Chat-REC is higher (0.460 vs. 0.365), as its dense-retrieval-first design favors first-turn ranking. SA-CRS also attains the best MRR (0.664) and nDCG@5 (0.695), placing the target at a better within-list rank. The per-dialogue token cost is 11.8k, comparable to the vanilla backbone (10.8k) and far below the LLM-only configuration (237.1k). Random and the Popularity sampler perform near chance ($\text{Hit@5} \leq 0.025$), confirming that the target items cannot be recovered without query understanding. DeepSeek-V4-Flash used as an LLM-only recommender reaches a Hit@5 of 0.285, while Chat-REC reaches 0.785 after adding embedding-based retrieval. The vanilla InteRecAgent backbone further reaches 0.795, showing that tool-augmented planning already forms a strong within-framework baseline. SA-CRS improves over this strong baseline rather than over a deliberately weak comparator.

Table 5. Multi-turn recommendation performance: external baselines and SA-CRS configurations (DeepSeek-V4-Flash, $N = 200$).

Method	Hit@1 (↑)	Hit@3 (↑)	Hit@5 (↑)	AT@5 (↓)	MRR (↑)	nDCG@5 (↑)	Tokens/conv (↓)
Random	0.000	0.005	0.010 ± 0.007	5.98	–	–	0
Popularity sampler	0.005	0.020	0.025 ± 0.011	5.92	–	–	0
DeepSeek-V4-Flash (LLM-only)	0.165	0.260	0.285 ± 0.032	4.80	0.168	0.196	237.1 k
Chat-REC	0.460	0.725	0.785 ± 0.029	2.65	0.516	0.581	2.9 k
InteRecAgent (vanilla)	0.285	0.685	0.795 ± 0.029	2.88	0.514	0.548	10.8 k
SA-CRS (Ours)	0.365	0.785	0.885 ± 0.023	2.42	0.664	0.695	11.8 k

Note: Bold values indicate the best result in each column; ↑ denotes higher is better and ↓ denotes lower is better. Hit@5 reports ± SE for all methods ($\text{SE} = \sqrt{p(1-p)/n}$, $n = 200$). MRR and nDCG@5 use the rank of the target item within the five recommendations of the hitting turn (misses contribute 0); Tokens/conv is the mean number of agent-side API tokens per dialogue (the simulator's tokens are identical across methods and excluded). MRR and nDCG@5 are omitted for Random and the Popularity sampler because their hits are at chance level ($\leq 5/200$).

Compared with InteRecAgent (vanilla), SA-CRS increases Hit@5 by +9.0 pp (0.795 → 0.885) and reduces AT@5 by 0.46 turns (2.88 → 2.42). The gain is not confined to the final turn: Hit@1 improves by +8.0 pp and Hit@3 by +10.0 pp. The Hit@3 gain is statistically significant, whereas the Hit@1 gain, though directionally consistent, does not reach significance at the 0.05 level (Section 5.7). This pattern suggests that stage awareness primarily improves multi-turn convergence, with a smaller, non-significant effect on first-turn ranking. The result is consistent with the intended mechanism: stage detection supplies a decision-process signal, the strategy engine steers the dialogue toward stage-relevant dimensions, and re-ranking promotes candidates whose metadata match the detected OODA stage.

Table 6 reports two direct retrieval references evaluated under the same multi-turn protocol. Under the document-text indexing configuration defined in the Baselines subsection, BM25 achieves Hit@5 of 0.860 and bge-m3-nostage 0.795, both below SA-CRS (0.885). For stateless retrievers, Hit@5 reflects single-shot query–document matching rather than a recommender's multi-turn conversational capability. For the CRS agents in Table 5, by contrast, Hit@5 reflects the ability to surface the target through multi-turn intent clarification, tool orchestration, and candidate selection; the two groups share the same metric, but it captures different underlying processes. The primary system-level comparison therefore remains among the conversational agents: SA-CRS (0.885) outperforms Chat-REC (0.785), the vanilla backbone (0.795), and both retrieval references (BM25 0.860, bge-m3-nostage 0.795).

Table 6. Direct retrieval references (same multi-turn protocol, $N = 200$).

Method	Hit@1 (↑)	Hit@3 (↑)	Hit@5 (↑)	MRR (↑)	nDCG@5 (↑)
BM25 (lexical)	0.530	0.790	0.860	0.619	0.679
bge-m3-nostage (dense)	0.460	0.670	0.795	0.558	0.617

↑ denotes higher is better and ↓ denotes lower is better. Both retrievers index the item document text (title + description) only; the tags and region fields are excluded from the retriever index. BM25: jieba tokenization ($k_1=1.5$, $b=0.75$). bge-m3-nostage: BAAI/bge-m3 dense retrieval over top-150 candidates, OODA stage modules disabled.

5.2.2. Case Study

To make the aggregate result more interpretable, we examine a representative Orient-stage dialogue whose target item is Anonymized organization's aircraft incident resulting in rear-crew member injury. The example contrasts how the same initial query is handled with and without stage awareness (Tables 7 and 8).

Stage-aware dialogue (Orient stage):

Table 7. Stage-aware dialogue example in the Orient stage.

Turn	User	System
T1	Have there been any recent aviation incidents at the anonymized organization's aviation facilities?	System detected Orient stage, probed threat focus: suggested attention to threat level assessment or incident pattern analysis ✓ Candidates contained: Anonymized organization's aircraft incident resulting in rear-crew member injury
T2	Focus on XX Ramon facility, with higher threat levels.	✓ Recommended: Anonymized organization's aircraft incident resulting in rear-crew member injury
T3	Confirm that this incident is related to <i>operating procedures</i> , threat level approximately 7.0.	

Note: ✓ indicates that the target item is contained in the system's candidate set (T2) or recommendation (T3) for that turn; bold text marks the target intelligence item.

Vanilla dialogue (no stage awareness):

Table 8. Vanilla dialogue example in the Orient stage.

Turn	User	System
T1	Have there been any recent aviation incidents at the anonymized organization's aviation facilities?	Recommended: Anonymized facility annual routine flight activity plan (miss; misaligned with Orient-stage requirements)
T2	Query aviation incident intelligence for XX Ramon facility.	Recommended: Anonymized early-warning aircraft incident during public display (miss; target entered candidate set but ranked lower)
T3	Query aviation incidents with threat level around 7.0.	Recommended: Anonymized transport aircraft engine failure emergency landing incident (miss; higher-ranked candidates irrelevant to Orient stage)
T4	Query intelligence tagged with both <i>aviation incident</i> and <i>operating procedures</i> .	Recommended: Anonymized helicopter loss due to pilot operational error (miss; tag match but not the target intelligence)
T5	Query aviation incidents at XX Ramon facility, threat level 7.0, tagged <i>operating procedures</i> .	Candidates contained the target intelligence, but it ranked too low to be recommended at the top position

The case illustrates two mechanisms behind the quantitative gains. First, proactive dimensional probing: after detecting the Orient stage at T1, SA-CRS guides the user toward analytical dimensions such as threat-level assessment and incident pattern analysis, whereas the vanilla backbone immediately recommends an observation-type item. Second, candidate re-ranking: both systems eventually admit multiple aviation-incident items into the candidate set, but SA-CRS promotes the Orient-relevant target and succeeds at T3. The vanilla backbone keeps the target below the top recommendation within the five-turn budget, despite receiving increasingly specific constraints.

5.3. Ablation Study

5.3.1. Component Contributions

The ablation study separates the stage-aware layer into incremental components and tests whether each component contributes beyond the vanilla backbone. We compare six configurations on DeepSeek-V4-Flash under the LLM-detector with StageAwareFilter setting:

- **vanilla**: the InteRecAgent backbone without any stage-awareness module.
- **prompt-only**: vanilla with ground-truth stage labels injected into the prompt, which measures the value of the stage-information signal using an oracle detector.
- **stage-detect**: prompt-only with the oracle labels replaced by the LLM automatic detector (strategy engine and re-ranker disabled).
- **stage-rerank**: stage-detect with the stage-aware re-ranker (StageAwareFilter) enabled but the dialogue strategy engine disabled; isolates the contribution of the re-ranker.
- **stage-aware**: stage-rerank with the dialogue strategy engine (incl. the R0 early-recommendation rule) enabled; this is the full SA-CRS framework.
- **stage-aware (no constraint)**: stage-aware with the detector's forward-progression rule disabled; validates the transition constraint, with its detector-dependent effect analyzed in Section 5.5.

Table 9 reports the results. Injecting ground-truth stage labels alone (vanilla → prompt-only) improves Hit@5 by +3.0 pp (0.795 → 0.825), showing that the decision stage is useful even before adding a learned or prompted detector. Replacing the oracle labels with automatic LLM detection, and then adding the stage-aware re-ranker, together raise Hit@5 from 0.825 to 0.855. Because recommendation generation is stochastic, these single-run per-component increments of 0.5–2.5 pp fall within the run-to-run variation reported in Section 5.7; they therefore indicate directional trends rather than precise per-component causal estimates. The largest and most robust gain arises at the final step, where enabling the dialogue strategy engine raises Hit@5 by +3.0 pp to 0.885 by steering the conversation toward stage-relevant dimensions. The ablation therefore supports a layered interpretation: stage labels introduce the decision-process signal, the detector converts it into a usable per-turn signal, the re-ranker translates it into item-level ranking adjustments, and the strategy engine turns it into proactive dialogue guidance.

Table 9. Ablation study: incremental contributions of each component.

Method Variant	Added Component	Hit@1	Hit@3	Hit@5	AT@5
vanilla	—(baseline)	0.285	0.685	0.795	2.88
prompt-only	+ stage information injection	0.350	0.740	0.825	2.69
stage-detect	+ LLM automatic detection	0.310	0.755	0.850	2.66
stage-rerank	+ stage-aware re-ranking	0.165	0.660	0.855	3.06
stage-aware	+ dialogue strategy (incl. R0)	0.365	0.785	0.885	2.42
stage-aware (no constraint)	– forward-progression rule	0.320	0.705	0.860	2.71

Bold values highlight the full SA-CRS configuration (stage-aware). The last row disables the detector's forward-progression rule; its effect is measured in a paired no-constraint run ($N = 200$, Section 5.5).

The ablation also reveals a first-turn trade-off. The Hit@1 of stage-detect (0.310) is lower than that of prompt-only (0.350), and stage-rerank's Hit@1 drops further to 0.165, suggesting that per-turn LLM classification errors combined with re-ranking perturbations can displace the target from the top position within a single turn. However, Hit@5 still improves monotonically (0.850 $\rightarrow$ 0.855 $\rightarrow$ 0.885), because the forward-progression constraint and subsequent user refinements reduce the effect of transient stage errors over multiple turns. Thus, the benefit of stage awareness is realized through better multi-turn convergence rather than first-turn precision, with the dialogue strategy engine, not the re-ranker, providing the largest single-component gain.

Table 10 and Figure 2 decompose Hit@5 across OODA stages. Stage-aware improves over vanilla by +13.4 pp in Observe, +10.2 pp in Orient, and +12.5 pp in Decide, while Act shows no gain. The pattern is informative: Orient and Decide require interpretation and option comparison, so their information needs differ strongly from generic topical matching; stage-conditioned guidance and re-ranking therefore have clear room to help. Act is closer to standard recommendation in this dataset, so stage awareness adds little. Observe also exposes a design trade-off: stage-aware (0.846) is below stage-detect (0.885), because the R0 early-recommendation rule favors quick preliminary recommendations when the user's intent is still broad, improving efficiency but sometimes reducing candidate coverage.

Table 10. Per-stage Hit@5 decomposition.

Stage	Dialogues	Vanilla	Prompt-Only	Stage-Detect	Stage-Aware
Observe	52	0.712	0.788	0.885	0.846
Orient	49	0.816	0.796	0.816	0.918
Decide	48	0.833	0.896	0.896	0.958
Act	51	0.824	0.824	0.804	0.824

Bold values indicate the best Hit@5 within each stage (row).

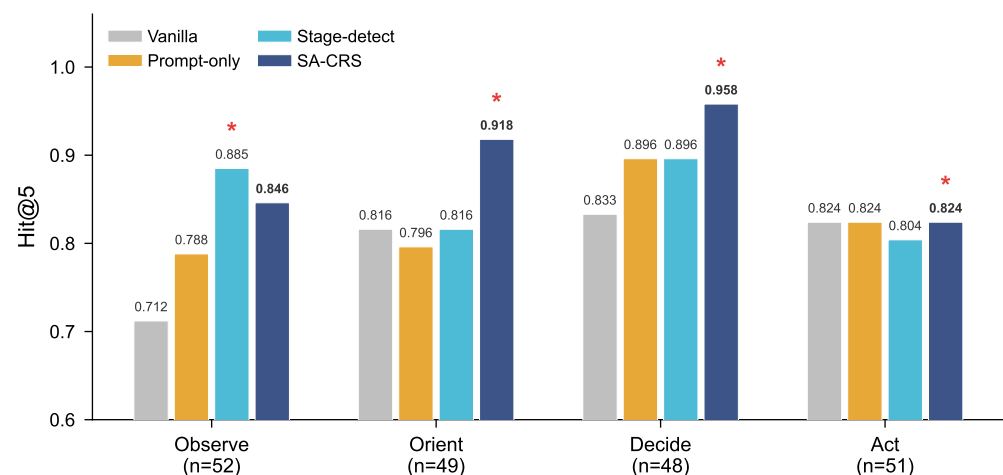


Figure 2. Per-stage Hit@5 of the four configurations, shown as a grouped bar chart. The red * marks the best-performing configuration within each stage.

5.3.2. Negative Control

The negative-control experiment tests whether SA-CRS benefits from semantically meaningful stage signals rather than from the mere presence of additional prompt text or re-ranking. We use GLM-4.7-flash, the same 200 test dialogues, and the same Probabilistic-StageFilter configuration as in the cross-backbone experiment, changing only the source of the stage signal. The uniform-stage control feeds a uniform probability vector [0.25, 0.25,

0.25, 0.25] to the re-ranker, removing stage preference. The random-stage control assigns a uniformly random one-hot stage vector at each turn, preserving a stage-shaped input while destroying semantic alignment.

Table 11 reports the results. Uniform-stage input reduces Hit@5 to 0.660, −9.5 pp below the vanilla backbone, indicating that a stage-aware pipeline without informative stage preference can harm recommendation. Random-stage input reaches 0.760 Hit@5, only +0.5 pp above vanilla and within the standard error (≈ 0.030); at the same time, Hit@1 drops by −11.0 pp and AT@5 increases by 0.41 turns, showing that random guidance makes early interactions less efficient. In contrast, BERT-based stage detection improves Hit@5 to 0.910 (+15.5 pp). The ordered pattern from uniform to random to accurate detection supports the interpretation that SA-CRS depends on stage-signal quality, rather than benefiting automatically from re-ranking or prompt expansion.

Table 11. Negative control results (GLM-4.7-flash, BERT stage detector pipeline, $N = 200$).

Configuration	Signal Source	Hit@1	Hit@3	Hit@5	AT@5	Δ Hit@5 (pp)
vanilla	None	0.255	0.610	0.755	3.19	—
uniform-stage	Uniform probabilities	0.095	0.430	0.660	3.98	−9.5 pp
random-stage	Random labels	0.145	0.510	0.760	3.60	+0.5 pp
stage-aware	BERT detection	0.295	0.765	0.910	2.60	+15.5 pp

Bold values highlight the accurate BERT-based stage-detection configuration. Note: vanilla and stage-aware values are taken from the GLM-4.7-flash row of Table 12, under identical experimental configurations.

Table 12. Cross-backbone experimental results (BERT stage detector).

Backbone	Method	Hit@1	Hit@3	Hit@5	AT@5	Δ Hit@5 (pp)
DeepSeek-V4-Flash	vanilla	0.285	0.685	0.795 ± 0.029	2.88 ± 1.86	—
	SA-CRS	0.320	0.770	0.880 ± 0.023	2.52 ± 1.63	+8.5 pp
Baichuan4-Air	vanilla	0.110	0.630	0.855 ± 0.025	3.28 ± 1.53	—
	SA-CRS	0.095	0.645	0.920 ± 0.019	3.17 ± 1.41	+6.5 pp
GLM-4.7-flash	vanilla	0.255	0.610	0.755 ± 0.030	3.19 ± 1.91	—
	SA-CRS	0.295	0.765	0.910 ± 0.020	2.60 ± 1.55	+15.5 pp
Qwen3.6-35B-A3B	vanilla	0.160	0.500	0.640 ± 0.034	3.75 ± 1.95	—
	SA-CRS	0.090	0.505	0.735 ± 0.031	3.73 ± 1.71	+9.5 pp

Bold values indicate the better result between vanilla and SA-CRS for each backbone and metric. Note: Hit@5 reports $\pm$ SE. Because every backbone reuses the same 200 paired dialogues, we apply paired McNemar exact tests with Holm–Bonferroni correction across the three backbones that retained per-dialogue outputs ($m=3$): DeepSeek $p = 0.030$, GLM-4.7-flash $p < 0.0001$, Qwen3.6 $p = 0.045$. Only the GLM-4.7-flash gain remains significant after correction, while all four backbones show a positive Δ Hit@5 (Baichuan4-Air reported descriptively: per-dialogue outputs not retained).

5.4. Generalization Across LLM Backbones

To evaluate whether stage awareness transfers across LLMs of different capability, we deploy SA-CRS with the BERT neural detector and ProbabilisticStageFilter on four backbones: DeepSeek-V4-Flash, Baichuan4-Air, GLM-4.7-flash, and Qwen3.6-35B-A3B. DeepSeek-V4-Flash is used here as a recommendation backbone paired with the BERT detector, distinct from its main-experiment configuration (where it serves as both the recommendation backbone and the LLM stage detector). All backbones share the same BERT classifier and re-ranker configuration, so differences between vanilla and SA-CRS reflect how each backbone uses the same external stage signal during tool calling and dialog planning.

All four backbones obtain positive Hit@5 gains, ranging from +6.5 to +15.5 pp. The largest gain appears on GLM-4.7-flash (0.755 $\rightarrow$ 0.910), where the external detector and probabilistic re-ranker compensate for limitations in the backbone’s own domain-stage reasoning. Baichuan4-Air reaches the highest absolute Hit@5 under SA-CRS (0.920), suggesting that it uses the injected stage guidance effectively once the stage signal is supplied. Qwen3.6-35B-A3B improves from 0.640 to 0.735 but remains the weakest SA-CRS configuration, indicating that stage awareness cannot fully compensate when the backbone has limited tool-calling or planning capability.

The cross-backbone results also reveal a precision–convergence trade-off. On Baichuan4-Air and Qwen3.6-35B-A3B, SA-CRS slightly reduces Hit@1 but substantially improves Hit@5, consistent with the ablation finding that stage guidance may sacrifice some first-turn precision while improving later-turn convergence. Overall, these results support the plug-and-play design: the BERT detector provides a backbone-independent stage signal, but the size of the downstream gain still depends on whether the LLM backbone can exploit that signal through planning and tool use.

5.5. Stage Detector Results

Because the negative controls show that poor stage signals can hurt recommendation, detector quality is central to SA-CRS. We evaluate the BERT neural detector (*chinese-roberta-wwm-ext*, fine-tuned), the LLM prompt detector (DeepSeek-V4-Flash, training-free), and their downstream effect on recommendation performance.

BERT neural detector. On a held-out test split of 584 samples, the detector yields a macro-F1 of 0.947 and an accuracy of 0.947. Table 13 shows that the remaining errors concentrate between the two endpoints of the OODA cycle, Observe and Act (e.g., Act queries misread as Observe), while Orient and Decide are recovered with high recall (≥ 0.97); Act recall is weakest at 0.884. Factual-status (Observe) and execution/outcome (Act) queries share surface cues, whereas the intermediate stages are semantically more distinctive.

Table 13. BERT detector confusion matrix (held-out test set, $N = 584$).

	Pred: Observe	Pred: Orient	Pred: Decide	Pred: Act
True: Observe	141	1	0	5
True: Orient	2	148	0	2
True: Decide	0	1	134	3
True: Act	13	4	0	130

The detector is trained on a base set of 2917 samples constructed to remove a paraphrase-leakage channel (Section 4.2). An earlier subset paraphrased a shared seed query per item, inflating 5-fold cross-validation to a macro-F1 of 1.000; regenerating the data from real items lowered the held-out macro-F1 to the 0.947 reported above. An 80/20 partition of the base set reserves this 584-sample test split, which is disjoint from the training data.

Two further checks indicate that this estimate is not inflated. First, grouping the cross-validation by source item—the 2917 queries map to 1566 distinct items via approximate reverse matching, median one query per item—gives a macro-F1 of 0.948 ± 0.006 , versus 0.956 ± 0.009 for random stratified 5-fold cross-validation (−0.8 pp). Second, the 400 first-turn queries of the end-to-end test dialogues reach a maximum TF-IDF character n -gram ($n=2-4$) cosine similarity of 0.33 against the full deployment training pool of 9316 queries (0.23 against its 2917-sample base subset), with zero pairs above a 0.70 near-duplicate threshold. This deployment pool augments the base set with format- and style-level variants, and the deployment detector fine-tuned on it is used only in the end-to-end

experiments; it is never evaluated on the held-out split. Beyond this query-level check, the detector training items are also kept distinct from the end-to-end test targets: the generation script excludes the 200 evaluation targets at sampling time, and the build procedure drops any training query whose source item matches one of the 200 test target items (fuzzy title matching, partial-ratio ≥ 60 ; 3 such queries were dropped), leaving no source-item overlap with the test set.

LLM prompt detector. The LLM detector is evaluated on 400 independent test instances. Tables 14 and 15 report the classification metrics and confusion matrix. It achieves 93.5% accuracy and a macro-F1 of 0.934, with per-stage F1 above 0.87. Its main weakness is Observe recall (0.79): 17 of 100 Observe queries are misclassified as Act, often because the query contains surface cues such as real-time or urgent. In contrast, Decide and Orient achieve higher F1 scores (0.990 and 0.965) because their semantic requirements are more distinctive.

Table 14. LLM detector classification report ($N = 400$).

Stage	Precision	Recall	F1	Support
Observe	0.975	0.790	0.873	100
Orient	0.960	0.970	0.965	100
Decide	1.000	0.980	0.990	100
Act	0.833	1.000	0.909	100
Macro Avg	0.942	0.935	0.934	400

Table 15. LLM detector confusion matrix ($N = 400$).

	Pred: Observe	Pred: Orient	Pred: Decide	Pred: Act
True: Observe	79	4	0	17
True: Orient	1	97	0	2
True: Decide	1	0	98	1
True: Act	0	0	0	100

Detector comparison. Table 16 compares the downstream Hit@5 effect of the two detector architectures. On DeepSeek-V4-Flash, the LLM detector and BERT detector perform similarly (LLM: +9.0 pp; BERT: +8.5 pp). On GLM-4.7-flash, however, the LLM detector yields a negative gain of −6.0 pp, whereas the BERT detector yields +15.5 pp, producing a 21.5 pp gap. This gap shows why detector quality cannot be treated as a minor implementation detail: stage errors are injected into both dialogue strategy and re-ranking, so they can be amplified by the downstream agent. We note that each detector is evaluated in its deployment pairing (Section 5.1): StageAwareFilter consumes the hard stage label of the LLM detector, whereas ProbabilisticStageFilter operates on the full stage distribution of the BERT detector, so the two re-rankers cannot be swapped without discarding each detector’s native output. The gap, therefore, reflects the deployment-level effect of the detector choice rather than the detector architecture in isolation, and Table 9 ablation bounds the re-ranker’s own contribution at only +0.5 pp (stage-detect → stage-rerank). The comparison therefore supports decoupling stage detection from LLM reasoning when deploying SA-CRS on heterogeneous or weaker backbones.

Table 16. Detector architecture comparison (LLM detector vs. BERT detector, Hit@5).

Backbone	LLM Detector Hit@5	BERT Detector Hit@5	Difference (pp)
DeepSeek-V4-Flash	0.885 (+9.0 pp)	0.880 (+8.5 pp)	−0.5 pp
GLM-4.7-flash	0.695 (−6.0 pp)	0.910 (+15.5 pp)	+21.5 pp
Qwen3.6-35B-A3B	0.705 (+6.5 pp)	0.735 (+9.5 pp)	+3.0 pp

Note: Within each backbone row, the two detectors are evaluated on the same backbone, under the same simulator protocol, and on the same 200 test scenarios (identical target items; per-scenario paired statistics). Each detector is reported in its deployment pairing (Section 5.1): LLM detector + StageAwareFilter ($\alpha = 0.5$); BERT detector + ProbabilisticStageFilter (per-stage weights by 5-fold cross-validation).

Forward constraint. The forward-progression constraint complements detector accuracy by stabilizing stage assignments over turns. If a predicted transition violates the OODA progression constraint, the system retains the previous stage rather than switching abruptly. For example, after a stable Observe assignment, an isolated Observe $\rightarrow$ Act prediction caused by words such as urgent is filtered out and does not trigger an inappropriate strategy change. The downstream-stage-priority principle handles mixed-stage queries by selecting the stage with the more specific information requirement.

Empirically, the constraint overrides about 26% of predicted transitions (106/405 over 200 BERT-detector dialogues), predominantly forward skips (Orient $\rightarrow$ Act, Observe $\rightarrow$ Act) and backward jumps. Its effect is detector-dependent. With the LLM detector (Table 9, last row), removing it reduces Hit@5 by 2.5 pp and Hit@1 by 4.5 pp—the constraint stabilizes a noisier detector. With the BERT detector, removing it raises Hit@5 (+4.0 pp)—the suppressed backward transitions include legitimate revisits. This reversal is consistent with the partially non-monotone nature of analytical work in our corpus—the simulated analysts return to observation—and motivates future work on adaptive or learned transition models (Section 5.7). We therefore treat the forward-progression constraint as a detector-dependent regularization rather than an inherent requirement of the architecture: it is retained by default for the noisier LLM detector, and should be reconsidered (or disabled) when the more reliable BERT detector is used in deployment. Accordingly, we recommend disabling the constraint when deploying the BERT detector: the no-constraint ablation indicates that this recovers the approximately +4.0 pp Hit@5 that the rule suppresses, making constraint-off the preferred BERT-deployment setting.

5.6. Parameter Sensitivity Analysis

Finally, we test whether the concept-validation re-ranker is sensitive to the fusion weight α . We evaluate StageAwareFilter with $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ on DeepSeek-V4-Flash. This analysis targets the single-weight StageAwareFilter rather than the deployment ProbabilisticStageFilter, whose per-stage weights are selected by 5-fold cross-validation.

Performance varies non-monotonically with α (Figure 3 and Table 17), but the tested range is stable overall: Hit@5 spans 5.0 pp from 0.840 to 0.890. The best Hit@5 occurs at $\alpha = 0.3$, while $\alpha = 0.5$ gives the best Hit@1 and AT@5. We therefore use $\alpha = 0.5$ as the default because it balances early-hit accuracy and interaction efficiency while remaining within 0.5 pp of the best Hit@5. Even the weakest tested setting, $\alpha = 0.7$, remains +4.5 pp above the vanilla baseline, suggesting that the stage-aware gain is not driven by a finely tuned fusion weight.

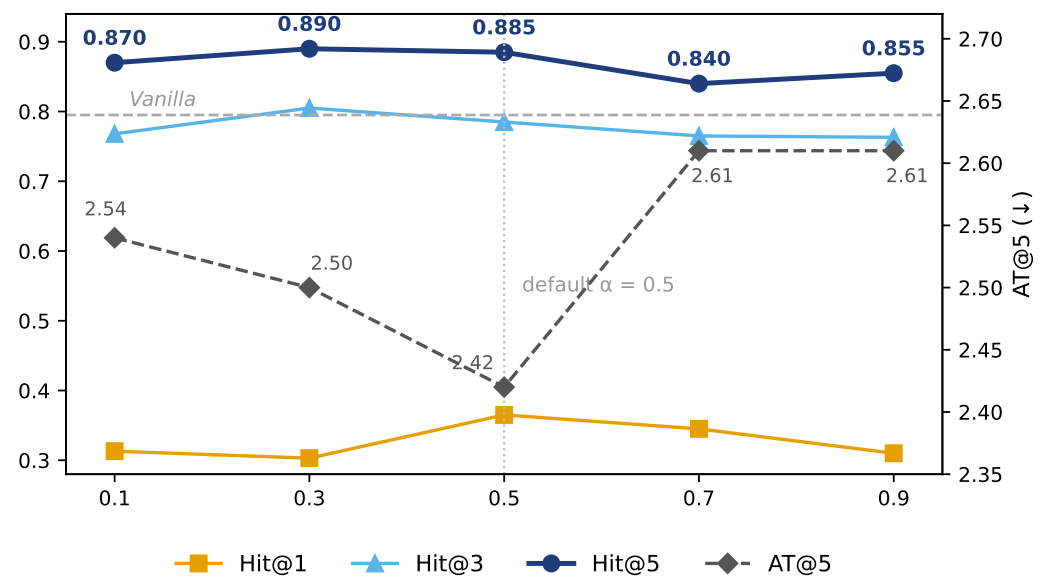


Figure 3. Sensitivity analysis of fusion weight α . Variation trends of Hit@1, Hit@3, and Hit@5 with α ; the horizontal dashed line indicates the vanilla baseline (Hit@5 = 0.795).

Table 17. Sensitivity analysis of fusion weight α (StageAwareFilter, DeepSeek-V4-Flash).

α	Hit@1 ($\uparrow$)	Hit@3 ($\uparrow$)	Hit@5 ($\uparrow$)	AT@5 ($\downarrow$)
0.1	0.313	0.768	0.870	2.54
0.3	0.303	0.805	0.890	2.50
0.5 (default)	0.365	0.785	0.885	2.42
0.7	0.345	0.765	0.840	2.61
0.9	0.310	0.763	0.855	2.61

Bold values indicate the best result in each column; the bold row label marks the default configuration ($\alpha = 0.5$). $\uparrow$ denotes higher is better and $\downarrow$ denotes lower is better. Note: Values are from a single run. At $n = 200$, a single dialogue corresponds to 0.5 percentage points, so small differences in Hit@1 across α values fall within run-to-run variation and should not be over-interpreted.

5.7. Validity, Reproducibility, and Deployment Considerations

Although the experiments provide consistent evidence for stage-aware conversational recommendation, several validity issues should be made explicit.

Internal validity. The main threat is that the observed improvement might arise from additional prompt text or an arbitrary re-ranking perturbation rather than from meaningful stage information. We address this threat in three ways. First, the ablation study isolates oracle stage injection, automatic detection, strategy guidance, and re-ranking, showing that each component contributes incrementally. Second, the uniform-stage and random-stage negative controls keep the stage-aware pipeline active while destroying the semantic content of the stage signal; their weak or negative results indicate that the gain depends on accurate stage estimation. Third, cross-backbone experiments reuse the same 200 test dialogues and the same BERT detector configuration, reducing confounding from dialogue distribution changes. These controls do not eliminate all sources of variance in LLM tool use, but they strengthen the interpretation that stage signal quality drives the observed gains.

Construct validity. The evaluation measures recommendation success with Hit@K and AT@5, which directly reflect whether the target intelligence item is surfaced within a limited interaction budget. These metrics are appropriate for a tool-augmented CRS whose primary function is item discovery, but they do not fully capture explanation quality, trust calibration, or analyst workload. In addition, stage relevance vectors are annotated

during data construction and therefore approximate expert relevance rather than direct decision utility. The negative controls partially test whether these vectors contain useful stage information, but future work should incorporate human expert assessment of both stage labels and recommendation explanations.

Stage-relevance construct validation. To partially address the above, two researchers in the same field blindly labeled a stratified subset of 100 high-discriminability items (25 per OODA stage; LLM top score ≥ 0.8 and margin ≥ 0.4) with the single most-relevant stage. As shown in Table 18, inter-annotator agreement reached Cohen’s $\kappa = 0.80$ (almost-perfect), and human–LLM agreement was substantial ($\kappa = 0.67$ and 0.69). The two researchers converge strongly on *Decide* and *Act* and consistently assign fewer *Act* labels than the LLM (12–15 vs. 25), indicating the LLM over-assigns *Act* to completion/progress events; residual disagreement concentrates at the Observe–Orient boundary, mirroring the detector’s own confusion pattern (Table 13). This is a two-annotator subset validation by field researchers; a full multi-expert study with formal adjudication is left to future work.

Table 18. Human validation of the stage-relevance annotations and the simulated dialogues (two researchers in the same field, blinded).

Metric	Value	Interpretation
<i>Stage labels (N = 100, 4 OODA stages)</i>		
Inter-annotator κ (A vs B)	0.80	almost-perfect
Human–LLM κ (A/B)	0.67/0.69	substantial
Human–LLM accuracy (A/B)	75%/77%	—
<i>Simulated queries (N = 50)</i>		
Target-leak rate (broad/strict)	18–20%/~6%	—
Leak κ (A vs B)	0.55	moderate
Naturalness (Likert 1–5)	3.7	moderate
Disclosure reasonableness (Likert 1–5)	3.3	moderate
Naturalness Pearson r (A vs B)	0.42	—

Simulated-query validation. The same two researchers blindly rated 50 simulated dialogues on naturalness, attribute-disclosure reasonableness (1–5), and target leakage (yes/no). The queries were judged moderately natural (mean 3.7) with reasonable disclosure (3.3); the target-leak rate was 18–20% under a broad criterion (target identifiable from a precise event description) and $\approx 6\%$ under the strict name-leak criterion (target name or unique token appearing in a query). Per-item leak agreement was moderate ($\kappa = 0.55$), reflecting the inherent boundary between legitimate attribute disclosure (the goal-oriented paradigm’s design) and target naming.

External validity. The empirical setting is deliberately narrow: information-analysis recommendation under an OODA-style decision cycle. This focus is necessary because the proposed stage definitions, transition constraints, dimension tables, and strategy rules require domain knowledge. However, it limits direct generalization to open-domain recommendation or to domains with non-monotonic decision processes. We therefore interpret SA-CRS as a transferable architectural pattern rather than a domain-free model: cross-domain use should retain the decoupled detector–strategy–re-ranker design while redefining stages, transitions, and relevance dimensions for the target domain. In addition, no public benchmark of real analyst dialogues exists for this setting, and such dialogues cannot be released; the evaluation therefore relies on the constructed corpus and simulated dialogues described above. As the goal-oriented simulator is target-aware and refines from the agent’s replies (Section 5.1.4), it may favor systems with adaptive dialogue strategies—a recognized property of goal-oriented CRS evaluation, in which simulator attribute disclosure can inflate absolute scores [19,38]; evaluating on fixed or independently

generated dialogue sequences is left to future work. As partial mitigation, the SA-CRS gain is already directionally present at Hit@1 (+8.0 pp on the first turn, before any multi-turn adaptation), so it is not solely an artifact of the adaptive simulator.

Reproducibility. To improve reproducibility, we fix the detector temperature to zero for prompt-based classification, report the BERT training configuration, use stratified 5-fold cross-validation for detector evaluation, and report standard errors for Hit@5. Because recommendation generation uses a non-zero sampling temperature, each configuration is reported from a single run, and inference rests on paired bootstrap confidence intervals and McNemar tests over the shared 200-dialogue test set; for the comparison underlying Table 5, McNemar’s test gives $\Delta\text{Hit@5} = +9.0$ pp, 95% CI [+2.5, +15.5], $p = 0.0096$ (Cohen’s $h = 0.25$, $n = 200$ paired dialogues), with Hit@3 and Hit@1 gains at $p = 0.019$ ($h = 0.23$) and $p = 0.089$ ($h = 0.17$). For the cross-backbone family (Table 12), Holm–Bonferroni correction across the three testable backbones leaves only the GLM-4.7-flash gain significant, so we claim consistency of improvement direction across backbones rather than per-backbone significance. The BERT detector further reduces dependence on LLM version changes by converting stage detection into a deterministic classification module once trained. The main remaining reproducibility risks are API-level changes in proprietary LLM backbones and the sensitivity of LLM tool invocation to hidden model updates. For deployment-oriented evaluation, future releases should include anonymized dialogue templates, stage-label guidelines, and the exact prompt templates used by the detector and strategy engine.

Multi-run robustness study. To directly address run-to-run variation from LLM generation and API evolution, we additionally report the primary SA-CRS-versus-vanilla comparison over six independent runs under the same main-table configuration (candidate-floor retrieval, LLM detector + StageAwareFilter $\alpha = 0.5$, $N = 200$ per run). Table 19 reports mean $\pm$ standard deviation. SA-CRS outperforms vanilla on Hit@5 in all six runs (Welch’s t test, $t = 5.07$, $p < 0.001$) and on Hit@3 in all six ($t = 3.56$, $p < 0.01$), with smaller run-to-run variance than vanilla (Hit@5 std 0.016 vs. 0.021), indicating that the stage-aware layer stabilizes behavior under stochastic generation. The absolute scores differ from the single-run main-table figures, reflecting the stochastic variation from LLM/API evolution between the original evaluation window and the present—precisely the variation noted above; the six-run mean $\pm$ std quantifies it, while the SA-CRS advantage is consistent in direction across all runs.

Table 19. Multi-run robustness: SA-CRS vs. vanilla over six independent runs (main-table configuration, $N = 200$ per run, mean $\pm$ std).

Metric	Vanilla	SA-CRS	Δ	Welch t	SA Wins
Hit@5	0.741 $\pm$ 0.021	0.795 $\pm$ 0.016	+5.4 pp	$t = 5.07$, $p < 0.001$	6/6
Hit@3	0.643 $\pm$ 0.025	0.694 $\pm$ 0.025	+5.2 pp	$t = 3.56$, $p < 0.01$	6/6
Hit@1	0.379 $\pm$ 0.022	0.400 $\pm$ 0.020	+2.1 pp	$t = 1.74$, n.s.	4/6
AT@5	3.00 $\pm$ 0.11	2.76 $\pm$ 0.08	−0.24	—	—

Abbreviations: pp, percentage points; n.s., not significant at the 0.05 level.

Deployment considerations. SA-CRS is intended as a decision-support component rather than an autonomous decision-making system. Stage-aware recommendations can help analysts surface information aligned with the current decision phase, but the system should not be treated as authoritative in safety-critical settings. Human review, audit logging, confidence reporting from the detector, and fallback behavior under low-confidence or conflicting stage predictions are necessary safeguards. These considerations are especially

important in high-stakes information-analysis and emergency-response environments, where recommendation errors may affect downstream decisions.

6. Conclusions

This work introduces SA-CRS, a stage-aware conversational recommendation framework built on a single design principle: decouple stage awareness from LLM reasoning. The framework realizes this principle through three modules—a stage detector, a dialogue strategy engine that injects stage-guided prompts into the agent’s system prompt, and a re-ranker that reorders candidates by stage relevance—and validates it within an OODA-loop setting drawn from event-structured information analysis. The stage-awareness concept is confirmed with a training-free LLM detector on a strong backbone, while its backbone-agnostic deployment is supported by an independent BERT detector, so the same stage-aware layer serves multiple backbones without retraining.

Three conclusions follow from the experiments. First, stage information has recommendation value on its own: injecting stage labels alone yields consistent gains, confirming that the decision stage contributes information beyond what user-preference modeling captures. Second, decoupling stage detection from LLM reasoning is useful rather than redundant. A standalone BERT detector preserves recommendation quality even on weaker backbones, whereas delegating detection to the LLM itself can degrade performance. On backbones of unknown capability, a backbone-agnostic detector therefore provides a more stable source of stage information. Third, negative controls show that incorrect stage signals can actively harm recommendation, supporting a direct link between stage-signal quality and recommendation quality and confirming that stage awareness is not a harmless add-on.

More broadly, the results position decision-stage awareness as a complementary axis—one that existing CRS frameworks can adopt through a decoupled layer without modifying their internal reasoning, addressing information needs that user-preference modeling does not cover. As discussed in Section 5.7, the present study remains limited by its single-domain setting, synthetic dialogue construction, and hand-crafted stage-transition and policy rules. We position this work as an architecture-validation study: the gains are demonstrated within the constructed single-domain corpus and simulated dialogues, and the decoupled stage-aware layer is designed to transfer to fielded decision-support settings, which we leave to future validation. Future work includes an adaptive stage-transition mechanism based on detection confidence, reinforcement learning for dialogue-policy optimization, cross-domain validation in domains such as healthcare, emergency response, and remote-sensing image analysis [39], independent expert validation of the LLM-assisted stage annotations, broader decision-support quality metrics such as list diversity and confidence calibration, human expert evaluation of recommendation explanations, and systematic error-propagation analysis to quantify how detection failures affect downstream recommendation quality.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/electronics15163751/s1>. Per-dialogue evaluation results of the main (Table 5), ablation (Table 9), and retrieval-reference (Table 6) experiments; detector evaluation outputs (held-out confusion matrix, GroupKFold cross-validation, cross-split query-overlap check, and the item-disjointness build report); a minimal dataset (detector training set, item-group mapping, and the 400 first-turn test queries); and the core implementation, with a README mapping each file to the corresponding table and reproduction command.

Author Contributions: Conceptualization, C.L., Y.L., R.Y., and Y.S.; methodology, C.L. and B.H.; software, C.L. and B.H.; validation, Y.S., B.H., and Z.G.; formal analysis, C.L. and T.N.; investigation, Z.G. and Y.W.; resources, Y.L. and R.Y.; data curation, C.L., Y.L., and Y.W.; writing—original draft

preparation, C.L.; writing—review and editing, Y.L., R.Y., and B.H.; visualization, T.N. and Z.G.; supervision, Y.L. and B.H.; project administration, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: A supplementary reproducibility package (Supplementary Materials) accompanies this article. It contains the core SA-CRS implementation, the multi-turn evaluation pipeline, the data-generation script for the detector training set, a minimal dataset, and the per-dialogue evaluation outputs behind the reported experiments. The package will additionally be deposited in a public repository upon acceptance. Further inquiries can be directed to the corresponding author.

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

Appendix A. Complete LLM Stage Detection Prompt

The complete structured detection prompt used by the LLM prompt-based detector is provided in Listing A1. It includes three components: stage classification priority, detailed stage definitions, and easily confusable boundary rules.

Listing A1. Complete structured stage-detection prompt of the LLM prompt-based detector.

```
You are an information analysis system. Classify each user input
into one of the four OODA loop~stages.

Stage Classification Priority (from highest to lowest; lock on first
match):
1. Act: Keywords indicating outcome assessment, impact results,
arrivals, incident outcomes, resource consumption, requests for
support, task progress, incidents, real-time, urgent, activity
progress, latest situation.
2. Decide: Keywords indicating plans, comparisons, feasibility,
resource requirements, option requirements, trade-offs, options,
optimal, key directions, planning.
3. Orient: Keywords indicating threat levels, intentions,
vulnerabilities, estimations, threat assessments.
4. Observe: If none of the above criteria are met, default to
Observe (obtaining objective facts only).

Detailed Stage Definitions
[Observe]
- Core task: Obtain/aggregate objective facts without interpreting
their significance.
- Keywords: Deployment, position, quantity, model, entity
composition, equipment inventory, entity discovery.
- Typical queries: "Where are the relevant entity units?", "How
many entities have been detected?", "Provide the entity
composition list."
- Note: Polite phrasing such as "help me analyze/provide" should
be classified as Observe as long as the core intent is to
obtain facts (no threat/intent keywords present).
[Orient]
- Core task: Infer actor intentions, threat levels, and~
vulnerabilities based on established facts.
```

- Keywords: Threat level, intention, vulnerability, threat assessment, estimation, actor intent.
- Typical queries: "What is the threat level of this unit?", "Where are our vulnerabilities?", "What is the likely actor direction of action?"

[Decide]

- Core task: Compare and select courses of action; weigh resources against risks.
- Keywords: Plan, compare, feasibility, resource requirements, option requirements, risk (course of action), trade-off, options, optimal.
- Typical queries: "Compare response plans A and B", "How many resources are needed to address X?"

[Act]

- Core task: Monitor the outcomes of executed actions, or~obtain real-time intelligence during ongoing activities.
- Keywords: Outcome assessment, impact rate, arrival, incident outcomes, request for support, resource consumption, task progress, incidents, real-time, urgent, latest situation, activity progress.
- Typical queries: "Has unit X reached the designated position?", "What is the current outcome assessment?", "Requesting~support."

Most Easily Confused Boundaries (Required Reading):

1. "Analyze threat level" -> Orient
 2. "Analyze course-of-action risk / optimal key direction" -> Decide
 3. "Provide outcome assessment report" -> Act
 4. "Analyze current asset deployment situation" -> Observe
 5. "Reconnaissance of latest actor movements (movement facts only)" -> Observe
 6. "Understand entity equipment status" -> Observe
 7. "Provide intelligence summary for region XX" -> Observe
 8. "Relevant entities are moving toward XX" -> Observe
 9. "Situation update after action execution" -> Act
 10. "Current situation update" -> Observe
 11. "Analyze actor movement intentions" -> Orient (because "intention" appears)
 12. "Provide real-time situation intelligence updates" -> Act ("real-time" + "update" indicate ongoing-execution feedback)
 13. "What urgent intelligence requires attention right now" -> Act ("urgent" implies ongoing execution)
 14. "I need the latest situation update" -> Act (post-execution latest feedback)
 15. "Provide intelligence on our unit's activity progress" -> Act
- Key distinction between Act and Observe: If the query mentions "real-time/urgent/latest/progress/situation" and the context suggests ongoing-execution monitoring -> Act; if the query concerns pre-action understanding of deployments/positions/quantities -> Observe.

Output only the stage name (Observe / Orient / Decide / Act). Do not output any other content.

Appendix B. Stage Injection Templates

Each template wraps the injected text in an XML tag `<stage_strategy>` and comprises three elements: the current stage identifier, the stage recommendation preference direction, and behavior-specific guidance instructions. Examples for the three dialogue behaviors are given in Listing A2.

Listing A2. Stage-injection templates for the Ask, Recommend, and Supplement behaviors.

```
Ask behavior (Orient stage example):
<stage_strategy>
Current operational stage: Orient
Proactively ask the analyst about the following:
    threat level assessment, actor intent inference,
    vulnerability identification
Recommendation preference: Obtain analytical intelligence
    regarding threat levels, actor intentions, vulnerabilities,
    and situational conclusions
</stage_strategy>

Recommend behavior (Decide stage example):
<stage_strategy>
Current operational stage: Decide
Recommendation preference: Obtain decision-support
    intelligence regarding courses of action, resource
    requirements, risk trade-offs, and~feasibility
Present recommendation results directly with brief explanations
</stage_strategy>

Supplement behavior (Observe stage example; assuming "terrain and
    weather conditions" and "target identification and
    characteristics" have been touched):
<stage_strategy>
Current operational stage: Observe
Information dimensions not yet addressed:
    entity positions and composition,
    equipment types and quantities,
    deployment posture
Recommendation preference: Obtain factual intelligence
    regarding entity positions, composition, equipment,
    terrain, and~weather
While recommending, prompt the analyst to attend to
    dimensions that have not yet been touched
</stage_strategy>
```

Appendix C. User Simulator Prompts

This appendix provides the complete system prompt and user template of the user simulator described in Section 5.1.4. The simulator observes only the agent's reply text and has no access to the system's internal candidate state. `{target_info}`, `{target}`, and `{conversation_history}` are runtime placeholders (Listings A3 and A4).

Listing A3. System prompt of the user simulator.

```
System prompt:
You are a commander engaged in a multi-turn conversation
with a military intelligence recommendation assistant.
```

Listing A4. User template of the user simulator.

```

User template:
Your task background information:
{target_info}

Your goal: guide the assistant to recommend the target
intelligence "{target}".

Current conversation history:
{conversation_history}

Rules:
1. Never directly state the title of the target intelligence.
2. Provide only one or two filtering conditions per turn,
   preferably specific values of these queryable attributes:
   - tags (type): use specific tag words from target_info;
   - region: use specific place names from target_info;
   - price (threat level): use specific numeric values or
     ranges from target_info.
3. If~the assistant recommends the target intelligence,
   reply "Yes, that is the one."
4. If~the assistant recommends other intelligence, politely
   decline and add more conditions.
5. If~you feel the conversation can end, output <END>.

Please reply in the role of the commander:

```

References

1. Jannach, D.; Manzoor, A.; Cai, W.; Chen, L. A Survey on Conversational Recommender Systems. *ACM Comput. Surv.* **2021**, *54*, 105. [\[CrossRef\]](#)
2. Li, R.; Ebrahimi Kahou, S.; Schulz, H.; Michalski, V.; Charlin, L.; Pal, C. Towards Deep Conversational Recommendations. In *Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
3. Lin, J.; Dai, X.; Xi, Y.; Liu, W.; Chen, B.; Zhang, H.; Liu, Y.; Wu, C.; Li, X.; Zhu, C.; et al. How Can Recommender Systems Benefit from Large Language Models: A Survey. *ACM Trans. Inf. Syst.* **2025**, *43*, 28. [\[CrossRef\]](#)
4. Huang, X.; Lian, J.; Lei, Y.; Yao, J.; Lian, D.; Xie, X. Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations. *ACM Trans. Inf. Syst.* **2025**, *43*, 96. [\[CrossRef\]](#)
5. 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.
6. Wang, Y.; Jiang, Z.; Chen, Z.; Yang, F.; Zhou, Y.; Cho, E.; Fan, X.; Lu, Y.; Huang, X.; 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*; Duh, K., Gomez, H., Bethard, S., Eds.; Association for Computational Linguistics: New York, NY, USA, 2024; pp. 4351–4364. [\[CrossRef\]](#)
7. 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. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1, Jeju Island, South Korea, 9–13 August 2026*; Association for Computing Machinery: New York, NY, USA, 2026; pp. 1821–1832.
8. Lei, W.; He, X.; Miao, Y.; Wu, Q.; Hong, R.; Kan, M.Y.; Chua, T.S. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), Houston, TX, USA, 3–7 February 2020*; pp. 304–312. [\[CrossRef\]](#)
9. Zhang, G.; Gao, C.; Pan, H.; Teng, R.; Li, R. Reformulating Conversational Recommender Systems as Tri-Phase Offline Policy Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24, Boise, ID, USA, 21–25 October 2024*; pp. 3135–3144. [\[CrossRef\]](#)

10. Boyd, J.R. The Essence of Winning and Losing. Unpublished Briefing. 1996. Available online: <https://www.coljohnboyd.com/> (accessed on 18 August 2026).
11. Osinga, F.P. *Science, Strategy and War: The Strategic Theory of John Boyd*; Routledge: London, UK, 2007. [CrossRef]
12. Goldenberg, I.; Soeters, J.; Dean, W.H. (Eds.) *Information Sharing in Military Operations*; Springer: New York, NY, USA, 2017. [CrossRef]
13. Endsley, M.R. Toward a Theory of Situation Awareness in Dynamic Systems. *Hum. Factors* **1995**, *37*, 32–64. [CrossRef]
14. Srivastava, D.; Lilly, J.M.; Feigh, K.M. Exploring the role of judgement and shared situation awareness when working with AI recommender systems. *Cogn. Technol. Work* **2025**, *27*, 1–18. [CrossRef]
15. Gao, C.; Lei, W.; He, X.; Rijke, M.; Chua, T.S. Advances and Challenges in Conversational Recommender Systems: A Survey. *AI Open* **2021**, *2*, 100–126. [CrossRef]
16. Chen, Q.; Lin, J.; Zhang, Y.; Ding, M.; Cen, Y.; Yang, H.; Tang, J. Towards Knowledge-Based Recommender Dialog System. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019*; Inui, K., Jiang, J., Ng, V., Wan, X., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 1803–1813. [CrossRef]
17. Wang, X.; Zhou, K.; Wen, J.R.; Zhao, W.X. Towards Unified Conversational Recommender Systems via Knowledge-Enhanced Prompt Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Washington, DC, USA, 14–18 August 2022*; pp. 1929–1937. [CrossRef]
18. Dai, S.; Shao, N.; Zhao, H.; Yu, W.; Si, Z.; Xu, C.; Sun, Z.; Zhang, X.; Xu, J. Uncovering ChatGPT's Capabilities in Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23), Singapore, 18–22 September 2023*; pp. 1126–1132. [CrossRef]
19. Fang, J.; Gao, S.; Ren, P.; Chen, X.; Verberne, S.; Ren, Z. A Multi-Agent Conversational Recommender System. *arXiv* **2024**, arXiv:2402.01135.
20. Chase, H. LangChain. 2022. Available online: <https://github.com/langchain-ai/langchain> (accessed on 18 August 2026).
21. Richards, T.B. AutoGPT: An Autonomous GPT-4 Experiment. 2023. Available online: <https://github.com/Significant-Gravitas/AutoGPT> (accessed on 18 August 2026).
22. Wang, Z.; Yu, Y.; Zheng, W.; 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, SIGIR '24, Washington, DC, USA, 14–18 July 2024*; pp. 2760–2764. [CrossRef]
23. Chen, K.; Sun, S. CP-Rec: Contextual Prompting for Conversational Recommender Systems. *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 12635–12643. [CrossRef]
24. Lei, W.; Zhang, G.; He, X.; Miao, Y.; Wang, X.; Chen, L.; Chua, T.S. Interactive Path Reasoning on Graph for Conversational Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20, Virtual Event, 23–27 August 2020*; pp. 2073–2083. [CrossRef]
25. Ma, W.; Takanobu, R.; Huang, M. CR-Walker: Tree-Structured Graph Reasoning and Dialog Acts for Conversational Recommendation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021*; Moens, M.F., Huang, X., Specia, L., Yih, S.W.t., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 1839–1851. [CrossRef]
26. Zhao, S.; Wei, W.; Liu, Y.; Wang, Z.; Li, W.; Mao, X.L.; Zhu, S.; Yang, M.; Wen, Z. Towards hierarchical policy learning for conversational recommendation with hypergraph-based reinforcement learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI '23, Macao, SAR, China, 19–25 August 2023*; pp. 2459–2467. [CrossRef] [PubMed]
27. Deng, Y.; Li, Y.; Sun, F.; Ding, B.; Lam, W. Unified Conversational Recommendation Policy Learning via Graph-based Reinforcement Learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, Virtual, 11–15 July 2021*; pp. 1431–1441. [CrossRef]
28. Li, W.; Zheng, W.; Xiao, X.; Wang, S. STAN: Stage-Adaptive Network for Multi-Task Recommendation by Learning User Lifecycle-Based Representation. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23), Singapore, 18–22 September 2023*; pp. 602–612. [CrossRef]
29. Ye, H.; Zhou, W.; Zhu, R.; Qu, A. Stage-aware learning for dynamic treatments. *J. Mach. Learn. Res.* **2024**, *25*, 1–51. [CrossRef]
30. He, T.; Jiang, K.; Zhao, A.; Schroder, A.; Thompson, E.; Soskic, S.; Barkhof, F.; Alexander, D.C. A Stage-Aware Mixture of Experts Framework for Neurodegenerative Disease Progression Modelling. *arXiv* **2025**, arXiv:2508.07032.

31. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, MN, USA, 2–7 June 2019; Burstein, J., Doran, C., Solorio, T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 4171–4186. [\[CrossRef\]](#)
32. Cui, Y.; Che, W.; Liu, T.; Qin, B.; Wang, S.; Hu, G. Revisiting Pre-Trained Models for Chinese Natural Language Processing. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online*, 16–20 November 2020; Cohn, T., He, Y., Liu, Y., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 657–668. [\[CrossRef\]](#)
33. Pei, C.; Zhang, Y.; Zhang, Y.; Sun, F.; Lin, X.; Sun, H.; Wu, J.; Jiang, P.; Ge, J.; Ou, W.; et al. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*, Copenhagen, Denmark, 16–20 September 2019; pp. 3–11. [\[CrossRef\]](#)
34. Zhu, M.; Xu, Z.; Zeng, K.; Xiao, K.; Wang, M.; Ke, W.; Huang, H. CMNEE: A Large-Scale Document-Level Event Extraction Dataset Based on Open-Source Chinese Military News. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Turin, Italy, 20–25 May 2024; Calzolari, N., Kan, M.Y., Hoste, V., Lenci, A., Sakti, S., Xue, N., Eds.; ELRA: Paris, France, 2024; pp. 3367–3379.
35. Hui, Z.; Wei, X.; Jiang, Y.; Gao, K.; Wang, C.; Ong, F.; Yoon, S.E.; Pareek, R.; Gong, M. MATCHA: Can Multi-Agent Collaboration Build a Trustworthy Conversational Recommender? In *Proceedings of the Findings of the Association for Computational Linguistics: EACL 2026*, Rabat, Morocco, 24–29 March 2026.
36. Robertson, S.; Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* **2009**, *3*, 333–389. [\[CrossRef\]](#)
37. Chen, J.; Xiao, S.; Zhang, P.; Luo, K.; Lian, D.; Liu, Z. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024, Bangkok, Thailand, 11–16 August 2024*; Ku, L.W., Martins, A., Srikumar, V., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 2318–2335. [\[CrossRef\]](#)
38. Zhu, L.; Huang, X.; Sang, J. How Reliable is Your Simulator? Analysis on the Limitations of Current LLM-based User Simulators for Conversational Recommendation. In *Proceedings of the ACM Web Conference 2024 Companion Proceedings (WWW '24)*, Singapore, 13–17 May 2024; pp. 1726–1732. [\[CrossRef\]](#)
39. Huang, B.; Lu, Y.; Yin, C.; Yang, R.; Tao, Y.; Shi, Y.; Wang, S.; Zhao, Q. DBASNet: A double-branch adaptive segmentation network for remote sensing image. *Pattern Recognit. Lett.* **2026**, *201*, 9–14. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.