DENIM: Ablating Catalog Access and Visual Grounding in an Agentic Fashion Shopping Assistant

Alessandro Francesco Maria Martina[1,2,*], Cataldo Musto[2], Marco de Gemmis[2], Pasquale Lops[2] and Giovanni Semeraro[2]

[1] *University of Pisa, Pisa, Italy*
[2] *University of Bari Aldo Moro, Bari, Italy*

Abstract

Large Language Models now power e-commerce shopping agents that converse with customers, call retrieval tools, and recommend from live product catalogs. Building one means deciding how the product catalog reaches the agent: what it can query or read at inference time, and in what representation. We isolate two of these decisions: what kind of catalog access the agent needs, and in what form visual item information should reach it. To answer both questions, we first implemented an agentic conversational recommender for fashion e-commerce. It is designed as a single orchestrated system with tool-augmented retrieval over a large fashion catalog. We then ran controlled single-change ablations on it, evaluated with a leakage-controlled simulated shopper in a cold-start setting. First, structured catalog access is the dominant factor: removing it costs 31 points of success rate, and most of that value (27 points) comes from a catalog *inspection* tool that lets the agent see what the catalog contains and which attribute values are valid before filtering, a capability that recent systems include in various forms but whose effect on retrieval had not been isolated. Second, what matters about visual information is the form in which it reaches the agent: injecting item images at runtime adds no measurable success over offline-extracted visual attributes, while removing those attributes' structured rendering costs a further 11 points even though the same information remains available in prose descriptions. At matched item representation, the orchestrated pipeline also outperforms ReAct and Reflexion-style baselines built on the same backbone and tools by 12–20 points, sustaining preference elicitation where they stall. All findings replicate in direction across two simulator LLM families, one of which shares the recommender's own model family. We distill the results into practical guidance for building e-commerce shopping agents.

Keywords

conversational recommendation, agentic AI, shopping agents, catalog inspection, visual grounding, user simulation

1. Introduction

E-commerce is one of the most active application areas for agentic AI: Large Language Models (LLMs) now conduct shopping dialogues, call retrieval tools, and assemble recommendations from live product catalogs [1, 2, 3]. What such an agent can do for a shopper depends on which product characteristics reach it, in what form, and how it can access them. In fashion these are concrete choices: structured product data determines what the agent can query about an item, while product imagery determines what it can see of it. How to handle each is a decision every such system takes, implicitly or explicitly, and recent systems resolve both in a variety of reasonable ways — from pure semantic search to structured query tooling on the structured side [4, 5], from offline visual captioning to runtime image injection on the visual side [6, 7]. What is missing is measured guidance on what each choice is actually worth.

GenAIECommerce'26: The Third Workshop on Agentic and Generative AI for E-Commerce, co-located with RecSys, September 28, 2026, Minneapolis, MN, USA
*Corresponding author.
alessandro.martina@phd.unipi.it (A. F. M. Martina); cataldo.musto@uniba.it (C. Musto); marco.degemmis@uniba.it (M. de Gemmis); pasquale.lops@uniba.it (P. Lops); giovanni.semeraro@uniba.it (G. Semeraro)
0009-0003-9609-1583 (A. F. M. Martina); 0000-0001-6089-928X (C. Musto); 0000-0002-2007-9559 (M. de Gemmis); 0000-0002-6866-9451 (P. Lops); 0000-0001-6883-1853 (G. Semeraro)

This paper supplies that measurement. We built DENIM (Dialogue-driven E-commerce Navigator with Inspection and Multimodality), an agentic conversational recommender for fashion: a single orchestrated system in which an LLM routes each user message to specialized components, retrieves from a large fashion catalog through tools, and composes grounded responses. DENIM is built to be ablated: every capability under study is implemented as a separable component behind a single configuration switch, so each experiment removes exactly one capability while everything else is held fixed (Section 3). The components themselves are deliberately standard: an intent-routed multi-component pipeline, a ReAct-style retrieval tool loop, a reranking stage, and offline visual enrichment, each documented across recent LLM-based recommenders and conversational shopping agents (Sections 2 and 3). The novelty lies in what we isolate from them and measure — above all, catalog inspection, which recent systems include in various forms but whose contribution to retrieval has never been measured on its own. The setting we measure in is chosen just as deliberately, and two of its properties shape everything that follows. Conversations are *cold-start*: the agent begins every session knowing nothing about the shopper — no purchase history, no stored profile — so every preference it acts on must be elicited within the dialogue itself, the situation of a new or anonymous visitor on a typical fashion storefront. The catalog is *feature-rich*: it carries structured metadata and product imagery, which is what makes the two questions we ask answerable at all — an agent cannot query attributes the catalog does not record, or see imagery it does not have.

Within that scope, we ask two questions about the interface between the agent and the catalog. **RQ1:** *What should the catalog expose to the agent, and how should the agent query it?* DENIM exposes the catalog through an *inspect* tool: for any filterable dimension — colour, fabric, silhouette, and so on — the agent can ask which values the catalog uses and how many items carry each, without fetching items. What it learns this way feeds its structured filter queries, issued alongside semantic search, so that user vocabulary ("navy", "dressy") is translated into valid filter values instead of guessed. **RQ2:** *In what form should visual information reach the agent?* DENIM extracts visual attributes from product images offline with a vision-language model (VLM); at runtime it can additionally show the images themselves to the components that reason about specific items. Both questions are answered with single-change ablations under a simulated shopper whose preference disclosure is tied, by construction, to the agent's own actions.

These two questions are not an arbitrary pair. In a cold-start session, the agent has no history and no stored profile, so everything it can know about items comes from the catalog, reaching it as the two kinds of input above. Together the questions therefore cover the whole interface between the agent and its only source of item knowledge, which is where a feature-rich domain like fashion puts the most at stake: the catalog records far more than any single query can use, and none of it helps unless the agent can reach it in a usable form. The other choices a shopping agent involves (e.g. its dialogue policy, and how it represents the user's preferences across turns) govern what the agent does with item knowledge once it has it, and we hold them fixed. A third question concerns the instrument rather than the interface. **RQ3:** *Does an orchestrated, role-specialized pipeline earn its complexity against a single tool-calling loop with the same backbone, tools, and item representation?*

Our contributions:

1. **The first isolated measurement of catalog inspection for retrieval.** Prior systems include inspection-like abilities, but never isolate what they contribute to retrieval (Section 2); we isolate the capability itself — letting the agent read the catalog's facet-value distributions so it can translate user vocabulary into valid filter values and see what the catalog can offer before querying — and measure what it is worth for retrieval (Section 5.1).
2. **Separate measurements of visual signal and visual form.** We separate how much visual signal the agent receives from the form in which it arrives, by ablating runtime image injection and, on top of it, the structured rendering of offline-extracted visual attributes (Section 5.2).
3. **A representation-matched comparison with single-loop agents.** We compare the orchestrated

pipeline against ReAct and Reflexion-style baselines with identical backbone, tools, catalog, and item representation, and use diagnostic metrics to locate where the designs diverge (Section 5.3).

4. **A target-free evaluation protocol.** A simulated shopper whose preference disclosure is tied to the agent's actions by a rule-based release mechanism, with the evaluation target never exposed to any LLM; diagnostic metrics that profile *how* a configuration elicits information; and a robustness protocol that repeats every experiment under a second simulator LLM family, including the recommender's own (Section 4).

2. Related Work

Catalog access in shopping agents. Recent conversational shopping agents increasingly include some ability to probe catalog data before acting on it, but in different roles and largely without isolation. InteRecAgent [5] exposes an ad-hoc text-to-SQL query tool that can look up item facts during planning; its ablations target orchestration mechanisms (planning, demonstrations, reflection), never individual tools. ProductAgent [8] summarises retrieved items into category statistics that ground its clarification questions, and its ablation shows that removing those statistics substantially degrades conversational product search. The statistics, however, inform only question generation; its retrieval queries are composed from the accumulated user demands, so the statistics' measured value reaches retrieval indirectly, through the user's answers. Likewise, an interview-style car recommender [9] computes attribute-value distributions over its current candidate set to select and phrase follow-up questions, and ablates that selection mechanism. CSales [10] narrows a product-category hierarchy through dialogue turns. The two ends of the tooling spectrum are also represented: in the conversational recommender of Friedman et al. [4], the agent's only catalog access is a free-text query sent to a search API, while Fashion-GPT [11] pairs the LLM with a dedicated fashion retrieval system. None of these works isolates inspection in service of *retrieval*. The novelty of our study lies there: we give the query-composing agent itself the catalog's facet-value distributions, so it can translate user vocabulary into valid filter values before committing structured queries, and we measure how much of the agent's success depends on that access by disabling it in an otherwise identical system.

Vision-language models in recommendation. How item images should be processed by a VLM-equipped recommender is an open design question, in both conversational and non-conversational settings. Benchmarking of VLM integration strategies in (non-conversational) sequential recommendation finds the *reranker* role most effective, with raw images outperforming captioned images at that stage [12]. LaViC [7] fine-tunes a VLM with distilled visual tokens for candidate selection in conversational recommendation and finds images help over title-only text. FashionM3 [13] fine-tunes a unified VLM for multi-round fashion assistance with image input and generation; Rec-GPT4V [6] prompts general-purpose VLMs with item images in single-shot recommendation tasks. The complementary strategy is *offline enrichment*: running a VLM over the catalog once and storing extracted visual information as text [12, 6]. Prior comparisons pit runtime images against thin text: titles or captions. None tests whether runtime images add anything once visual content has been extracted offline into *structured attributes*, nor whether the form of that extracted text matters. Both gaps are addressed here.

Simulation-based evaluation. Comparing designs one change at a time requires many multi-turn conversations in which a comparable shopper reacts to each variant, a scale and a degree of control that studies with human participants cannot easily provide. LLM-based user simulation is therefore standard practice for evaluating conversational recommenders [14, 15, 16], and its pitfalls are well documented: simulators leak target-item information into dialogues [17], motivating plugin-controlled disclosure [18] and target-free designs [19]; simulators incorporating uncertain

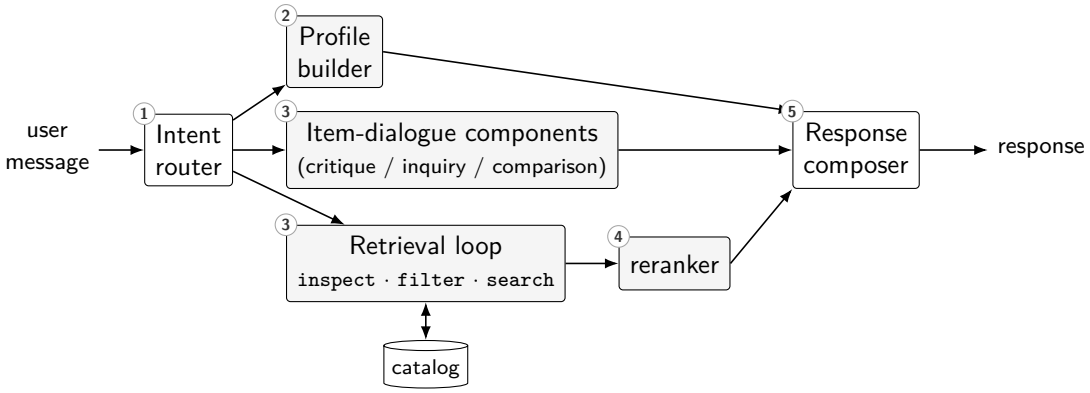

Figure 1: DENIM in outline. An intent router activates components per turn; the retrieval loop runs every turn, iterating up to 10 reason–act steps over the three catalog tools; a response composer merges component outputs. Circled numbers give the order within one turn (components numbered alike run in parallel). Shaded components can be ablated or reconfigured independently.

and passive user profiles demonstrate that the performance of current agentic recommenders degrades sharply outside ideal interaction scenarios [20, 21]. Our simulator follows a target-free design (Section 4.3), and every experiment is repeated under a second simulator LLM family.

3. The DENIM Agent

DENIM decomposes conversational recommendation into an intent router, a profile builder, a retrieval loop, a reranker, three components that handle item-directed dialogue acts, and a response composer that merges all outputs into one reply (Figure 1). All components share one backbone LLM without task-specific fine-tuning, and each is governed by a system prompt that fixes its inputs and a structured output schema. Each part follows documented practice: decompositions into role-specialised LLM components appear in recent conversational recommenders [22, 23], tool-augmented retrieval in [5], an LLM reranking stage over tool-returned candidates in [4, 23], and offline visual enrichment in the multimodal recommendation literature [12, 6].

3.1. Components

We describe the components in the order in which they act within a turn (the circled numbers of Figure 1).

Intent router. A classifier call over each incoming user message sets four independent boolean flags: whether the message states a preference, critiques an item already shown, asks a question about one, or requests a comparison. Each flag activates the corresponding component; the retrieval loop and the response composer run every turn regardless.

Profile builder. Activated when the message carries preference information, it merges that information into a structured profile, preserving previously stated values unless the shopper contradicts them and recording avoidances as explicit negative constraints. The current profile conditions both retrieval and reranking.

Critique handler. Acknowledges a critique of an item already shown and, when the critique is too vague to yield an actionable signal, emits one targeted follow-up question. It receives the item images when runtime image injection is enabled.

Inquiry and comparison handlers. Answer factual questions about shown items and compare them against each other. Both receive the item images when runtime image injection is enabled, so that questions about appearance can be answered from the image where the metadata does not settle them.

Retrieval loop. Runs every turn as a reasoning-and-acting tool loop [24]: reading the profile and the recent dialogue, it issues catalog tool calls (Section 3.2) for up to ten reason–act steps, until it either holds candidates worth showing or settles on a preference dimension worth asking about.

Reranker. Scores the candidates collected by the retrieval loop against the profile and the recent conversation and selects the items to present, each with a short explanation. It receives the item images when runtime image injection is enabled.

Response composer. Merges the labelled outputs of every component invoked in the turn into one coherent reply. It calls no tools and receives no images: its input is what the other components produced.

This decomposition is the measurement instrument: each capability under study maps to one switch, with everything else held fixed. Both capabilities enter through the components just described — the tools the retrieval loop may call (Section 3.2), and the visual information delivered to the reranker and the item-dialogue components (Section 3.3).

3.2. Retrieval tools

Three thin, learning-free tools expose the catalog:

`inspect(filters)` returns the distribution of values along each filterable dimension — counts per colour, formality level, silhouette, and so on — optionally under filter constraints, without fetching items. Constrained to a single garment type, for example, it returns (abridged and reflowed from a session log):

```
Total matching items: 220 (filters: garment_type=Pyjama bottom)
color: Blue: 64, Grey: 46, Pink: 37, [...], Mole: 5, [...]
style: classic: 65, minimalist: 57, romantic: 36, [...], bohemian: 6, [...]
silhouette: relaxed: 121, straight: 80, fitted: 17, [...]
```

The agent uses this in two ways: to map the user's words onto the values the catalog actually uses, and to reason about what the catalog holds — the counts show which options exist, which are common, and which combinations are rare or missing, before any items are fetched.

`filter(filters)` performs exact-match structured queries. Invalid dimensions or values return an informative error listing valid alternatives rather than failing silently, enabling self-correction inside the tool loop.

`search(query)` is dense vector retrieval over LLM-authored item descriptions: natural language in, ranked items out.

The error-message design matters for interpretation later: even with `inspect` removed, the agent retains a *reactive* vocabulary-correction path through `filter`'s error responses, so our measurement of `inspect`'s value is conservative.

Figure 2 shows the tools at work in one session from our study. Asked about fabric and style, the shopper answers in ordinary prose — fabric that "feels like silk, with that smooth, glossy sheen", a style that is "calm and intentional", a "truly relaxed cut". The agent inspects the catalog's facet distributions and maps the paraphrases onto schema values; "feels like silk, with that smooth, glossy sheen" becomes `fabric: satin-silk-like`, a value the shopper never utters. The resulting four-constraint query is over-constrained and matches nothing; the tool's

Shopper (opening message). "Hi, I'm looking for a night gown for loungewear and sleepwear — something comfortable to wear around the house during downtime."

Agent (excerpt). "[...] are you looking for something in a soft cotton or jersey for maximum coziness, or perhaps something more luxurious like silk or satin? [...] a particular style in mind, such as something more minimalist or perhaps a more romantic look?"

Shopper. "I'm definitely leaning towards a more luxurious feel, so I'd love something where the fabric feels like silk, with that smooth, glossy sheen. As for the look, my style is calm and intentional — I pick pieces that are sleek and functional without extra fuss. [...] could you show me options that balance that high-shine finish with a truly relaxed cut, rather than something that looks too formal or structured?"

Retrieval loop (second turn, four tool calls):

```
→ inspect({})
← Total matching items: [...] fabric: jersey-knit: 38823,
  woven-smooth: 25226, [...], satin-silk-like: 2780 [...]

→ filter({fabric: "satin-silk-like", garment_type: "Night gown",
  silhouette: "relaxed", style: "minimalist"})
← "No items found for filter: [...]. Try broadening your filters
  or removing a constraint."

→ filter({fabric: "satin-silk-like", garment_type: "Night gown"})
← Filter results: 10 returned (total matching: 27)
  1. [ID: 0636839001] Gloria n-slip
    Perceived Colour Master Name: Black
    Style Archetype: classic
    Silhouette: straight
    Formality: semi-formal
    Primary Fabric Appearance: satin-silk-like
    Design Details: V-neck | sleeveless | lace-trim [...]
  2.-10. [...]

→ search({query: "relaxed fit minimalist satin silk nightgown sleepwear [...]"})
← Found 10 items [...]
```

The reranker selects from the collected candidates; the response composer writes:

Agent (excerpt). "[ID: 0761698002] CAMP Twain n-slip (W) — This matches your request for a high-shine, silk-like fabric with a minimalist, 'without extra fuss' aesthetic and a comfortable, non-structured feel. [...]"

Shopper. "That's it! [ID: 0761698002] is the one. Thank you!" *Success.*

Figure 2: One session from the Reference configuration under the primary simulator, abridged and reflowed: [...] marks elisions; tool calls carry the tool names of Section 3.2 with their arguments verbatim. The shopper describes preferences in ordinary prose; the agent aligns them with the catalog's exact vocabulary by inspecting facet distributions ("feels like silk" → `satin-silk-like`) and repairs an over-constrained query using the tools' informative responses. The item card shows the structured attribute fields of Section 3.3 as components see them.

empty-result message suggests broadening, the agent drops the silhouette and style constraints and recovers 27 candidates, and a semantic search re-injects the dropped preferences before reranking. The shopper accepts one of the presented slips on the next turn.

3.3. Visual information: offline extraction, three surfaces

A vision-language model[1] extracts five visual attributes from each product image: style archetype, silhouette, fabric appearance, formality, and design details. These extracted attributes then surface in three places: (*i*) as *structured fields* in the item representation shown to components; (*ii*) inside each item's *prose description*, which an LLM writes from the full attribute record and which also feeds the semantic search index; and (*iii*) as *filterable dimensions* exposed by `inspect` and `filter`. The three surfaces are independent, and our ablations manipulate them separately.

At runtime, DENIM can additionally inject the product images themselves, alongside the item

[1] Qwen-3-VL-32B-Instruct; extraction runs once, offline, over all product images.

Table 1
Study conditions. The first three ablations change exactly one thing relative to the Reference configuration; *w/o Structured Visual Attributes* removes the attribute fields on top of *w/o Runtime Images*, so the two visual conditions form a ladder in which each step is a single change. Baselines replace the orchestrated pipeline with a single tool-calling loop over the same three tools and the same item representation as *w/o Runtime Images*.

Condition	Change relative to Reference
Reference	— (all capabilities on)
w/o Structured Retrieval	`filter` and `inspect` removed; `search` only
w/o Inspection	`inspect` removed; `filter` and `search` remain
w/o Runtime Images	image injection off; all text unchanged
w/o Structured Visual Attributes	attribute fields also removed from item text[2]
ReAct	single tool-calling loop [24]
Reflexion-style	ReAct + per-turn self-critique (adapted from [25])

text, into the components that reason about specific items: the reranker — where VLM image use is an established pattern [12, 7] — and the critique, inquiry, and comparison handlers.

4. Experimental Design

The previous section described DENIM's components and the two capabilities under study. This section sets out how they come together in the experiment: the configurations we compare, the catalog and personas they are evaluated on, the simulated shopper that interacts with them, the metrics, and the protocol. In total the study comprises 4,200 simulated shopping sessions: seven configurations, 100 personas, three replicates each, run under two agent–simulator pairings. The configurations defined in Section 4.1 answer the research questions in turn: two retrieval ablations for RQ1, two visual conditions for RQ2, and two single-loop baselines for RQ3.

4.1. Study conditions

Table 1 lists the seven conditions. Two deserve precision. *w/o Structured Visual Attributes* starts from *w/o Runtime Images* (image injection is off in both) and additionally removes the structured attribute fields from the item text shown to components, while the prose descriptions (which were generated *from* those attributes), the semantic index, and the filter schema all remain: the step between the two visual conditions isolates the *form* in which extracted visual content reaches the agent, not its availability. *Reflexion-style* augments the ReAct loop with a self-critique step after every turn, adapted from Reflexion [25].

Each condition acts at a specific point of Figure 1. The two retrieval ablations change the toolset within the retrieval loop: *w/o Inspection* removes `inspect` from the loop; *w/o Structured Retrieval* removes `inspect` and `filter`, leaving only semantic search. The two visual conditions change what the components receive: *w/o Runtime Images* stops image injection into the reranker and the item-dialogue components, and *w/o Structured Visual Attributes* additionally strips the attribute fields from the item text that all components read. The baselines replace the entire pipeline with a single tool-calling loop.

4.2. Catalog, personas, and ground truth

We use the H&M fashion catalog [26]: 131 product types, product photographs suitable for visual attribute extraction, and metadata rich enough to make structured filtering meaningful. One

[2]Prose descriptions, the semantic index, and the filterable dimensions are unchanged; the manipulation removes the structured attribute fields, along with three original catalog fields (detailed colour, pattern, customer segment), from the item text.

hundred simulated-shopper personas are constructed from real (customer, product) transactions, selected for diversity across product groups, occasions, styles, and formality levels. Each persona opens the conversation the way a shopper would — "I'm looking for an upper body garment, specifically a blouse, to wear to an evening-out event" — which states the garment category, the garment type, and the occasion, but says nothing about the colour, style, silhouette, fabric, formality, design details, or pattern the shopper actually has in mind. From here on we call the three preferences stated up front *volunteered*, and the seven the persona keeps back *withheld*. The withheld preferences are what the agent has to draw out: they surface only through interaction (Section 4.3).

Because real shoppers accept any item that fits, success is judged against an *ideal set* rather than a single target: all catalog items matching the persona's seed product on at least 9 of these 10 preference dimensions (median size 2; 74% of personas have ≤ 5 ideal items). A session succeeds when the agent presents any ideal-set item; it then ends, or runs to a 10-turn budget.

4.3. Simulated shopper

The simulated shopper works as follows. Each session opens with the persona's fixed opening message, which states its three volunteered preferences. From then on, the seven withheld preferences surface only in response to what the agent does. When the agent's action earns a withheld preference, a bookkeeping engine marks that preference as disclosed and the shopper may state it from that turn onward; we call this event a *release*. Two agent actions can trigger one: showing items that match a withheld preference, and asking about the dimension that preference belongs to. Every release is decided and recorded, turn by turn, by the bookkeeping engine; an LLM writes the shopper's replies but never determines the session outcome. Three properties of this design matter for interpreting the results.

Outcomes are decided by code: the ideal set never appears in any LLM prompt; the only component that reads it is the deterministic acceptance check (normalised item-ID matching against the items the agent presented). Session outcomes therefore rest on exact matching rather than LLM judgment.

Release accounting is rule-based: withheld preferences are released through two channels. When the agent *shows items*, a deterministic routine compares each item's catalog attributes to the persona's withheld values and releases up to two matching dimensions per turn (a random subset when more qualify). When the agent *asks a question*, an LLM classifier maps the question to preference dimensions; it sees the agent's utterance and the dimension names with generic descriptions and example terms (not the persona's values), and it outputs dimension names only. Every release is recorded per-turn, per-channel; all metrics are computed from that record. In the session of Figure 2, for instance, the agent's opening question about fabrics and style released the persona's withheld fabric and style preferences through the question channel; showing items on the next turn released colour and formality through the item channel.

Utterances are generated under instruction: the LLM that writes the shopper's replies is given the persona's withheld values — which keeps its reactions consistent — together with an instruction not to state them before release; all metrics read the release record rather than the utterance text. Every canonical value also comes with paraphrases generated offline, which the shopper is instructed to prefer once the value is released, so that preferences reach the agent in everyday language rather than only in the catalog's own vocabulary.

The simulated shopper always sees the images of presented items, so its feedback is visually grounded in every condition; only the *agent's* access to visual information varies. The fixed opening message guarantees identical starting conditions across conditions and replicates [16]. The simulator models a focused, cooperative shopper — a deliberate control that isolates agent-side effects; behavioural realism is discussed with the other scope choices in Section 6.

4.4. Metrics

Success Rate (SR). Fraction of sessions in which the agent presents an ideal-set item within the turn budget.

Preference Ask Ratio (PAR). Of the withheld preferences released in a session, the fraction released by the question channel. When a dimension qualifies through both channels in one turn it is credited to the item channel, so the two channels partition releases.

First Recommendation Turn (FRT). Index of the first agent reply that presents items, counting the agent's first reply as turn 0.

Wasted Turns (WT). For failed sessions, the number of agent turns after the last new preference was released — a stalling measure, 0 for successful sessions; it measures how long a failed session remains in the *unproductive stagnation* regime described in [21].

Session length. Turns until success or budget.

PAR, FRT, and WT form a diagnostic fingerprint of *how* a configuration elicits information; SR measures whether it succeeds.

4.5. Protocol

The agent backbone is Gemma-4-31B-it in all conditions, without recommendation-specific fine-tuning; the simulator LLM is Qwen-3.5-27B in the primary configuration. Using different model families for agent and simulator avoids self-preference bias [27]. We additionally repeat every experiment under a *same-family* simulator (Gemma-4-31B-it); this pairing produces no score inflation — the mean per-persona SR difference from the cross-family configuration is +0.8 points, and the Reference configuration actually scores *lower* under it (Table 2) — so we report it as a robustness replication alongside the primary configuration. Each (condition × persona) cell runs 3 replicates: $7 \times 100 \times 3 \times 2 = 4{,}200$ sessions. SR contrasts use population-averaged GEE (logistic link, persona-clustered, exchangeable working correlation); WT contrasts use Wilcoxon signed-rank on persona-level means; p-values are Benjamini–Hochberg corrected within each test family (ablations vs. Reference; the single-change visual-form contrast of Section 5.2; the baseline comparisons of Section 5.3; targeted WT contrasts). Significance markers: * $p < .05$, ** $p < .01$, *** $p < .001$ after correction.

5. Results

Table 2 reports success rates for all seven configurations under both agent–simulator pairings; Table 3 reports the diagnostic metrics that explain the mechanism behind each effect. The three subsections take RQ1–RQ3 in turn.

5.1. Catalog access dominates, and inspection carries most of it (RQ1)

RQ1. The catalog must let the agent inspect what it contains, and the agent must be able to query it with structured filters — and of the two, the inspection is worth more.

Removing structured catalog access entirely (*w/o Structured Retrieval*) is the largest effect in the study: −31.3 points of SR in the primary configuration and −23.3 under the second simulator family (Table 2), both $p < .001$. Removing only the `inspect` tool costs −26.7 and −17.3 points respectively — most of the total effect — even though the ablated agent retains reactive vocabulary correction through `filter`'s error messages (Section 3.2), which makes these numbers a floor on the value of proactive inspection.

The diagnostic metrics explain the mechanism (Table 3). Without its structured tools the agent does not elicit less: preferences keep flowing, and the ask share actually rises (PAR .21 → .39). What breaks is the conversion of elicited preferences into effective queries. First

Table 2
Main results. ΔSR: change vs. Reference (GEE, BH-corrected; for the two baselines the matched comparison against *w/o Runtime Images* is reported in Section 5.3, and for *w/o Structured Visual Attributes* the single-change contrast against *w/o Runtime Images* is reported in Section 5.2). Left block: primary configuration (Qwen simulator); right block: same experiments under the Gemma simulator.

	Qwen simulator		**Gemma simulator**	
Condition	SR	ΔSR	SR	ΔSR
Reference	.850	—	.800	—
w/o Structured Retrieval	.537	−.313***	.567	−.233***
w/o Inspection	.583	−.267***	.627	−.173***
w/o Runtime Images	.820	−.030	.870	+.070*
w/o Structured Visual Attributes	.710	−.140***	.733	−.067
ReAct	.703	−.147***	.673	−.127***
Reflexion-style	.697	−.153***	.683	−.117**

Table 3
Diagnostic metrics, primary configuration. PAR: share of preferences released via questions; FRT: first recommendation turn (0-indexed); WT: wasted turns (stalling; 0 for successful sessions). Stars mark the targeted WT contrasts of Section 5.

Condition	**PAR**	**FRT**	**WT**	**Turns**
Reference	.21	0.72	0.55	5.3
w/o Structured Retrieval	.39	1.65	1.54***	7.4
w/o Inspection	.35	1.53	1.46***	7.3
w/o Runtime Images	.22	0.77	0.66	5.3
w/o Structured Visual Attributes	.29	0.69	1.10	6.0
ReAct	.40	0.01	1.85***	5.2
Reflexion-style	.40	0.00	1.78***	5.4

recommendations come almost a full turn later (FRT $0.72 \rightarrow 1.65$), sessions run two turns longer, and the agent stalls after exhausting productive moves (WT $+0.99$, $p < .001$; *w/o Inspection* $+0.91$, $p < .001$). The bottleneck is catalog knowledge: without the ability to see what the catalog contains, elicited preferences die between the dialogue and the query. The tool logs show this directly, and the failures come in two kinds — one for each thing `inspect` provides. With `inspect` available, 0.7% of the agent's `filter` calls were rejected for naming a value the catalog does not use, and 10% returned no items; without it, 34% were rejected (the agent guesses words the catalog does not use) and 37% came back empty (it asks for combinations the catalog does not hold). Figure 2 showed the intact pathway — paraphrase to facet-grounded query — that these conditions sever.

5.2. Visual information: form over pixels (RQ2)

RQ2. Visual information should reach the agent as structured attributes extracted offline; showing it the images themselves at runtime adds nothing measurable on top.

The two visual conditions form a ladder — *w/o Runtime Images* switches off image injection; *w/o Structured Visual Attributes* additionally removes the attributes' structured rendering — so each step changes one thing, and the two steps point the same way.

From one side: removing runtime image injection does not hurt. SR moves by -3.0 points (n.s.) in the primary configuration and by $+7.0$ points (removal *helps*; $p < .05$) under the second simulator family, with diagnostics flat. The two families thus disagree on the sign of a small effect; what they agree on is that the images never improved success. On this backbone, whatever the images carry beyond the extracted attributes, the agent cannot use it — consistent with

a boundary around prior findings rather than a contradiction of them: runtime images beat *captions* in VLM reranking [12] and *title-only* text in conversational selection [7], but against *structured extracted attributes* they add nothing measurable here.

From the other side: taking the second step — removing the structured rendering of those attributes (the labelled fields in the item card of Figure 2) while the same content remains in prose, in the index, and in the filter schema — costs −11.0 points in the primary configuration ($p = .002$) and −13.7 under the second simulator family ($p < .001$): a consistent effect in both. Relative to Reference the drops read −14.0 and −6.7 (Table 2), but those contrasts fold in the image effect above, which pushes in opposite directions in the two families; the single-change contrast is the clean measurement. To interpret it as an effect of *form*, the content must really still be there — and it is: measured over the five extracted attributes, the attribute value appears in its item's prose description for 98% of items on average (minimum 94.9%), so the manipulation changed only how the information was rendered.

Together: the agent uses visual information *symbolically*. Give it the pixels and success does not improve; take away the structured fields and performance drops even with the same facts available in prose.

5.3. Viability against single-loop agents (RQ3)

RQ3. At matched item representation, the orchestrated pipeline outperforms both single-loop baselines, and it does so by sustaining elicitation rather than by recommending sooner.

Against ReAct and Reflexion-style baselines with the same backbone, tools, catalog, simulator, and item representation — the matched comparison is against *w/o Runtime Images*, since baselines consume text only — the pipeline scores +11.7 and +12.3 points of SR respectively ($p < .001$), widening to +19.7 and +18.7 under the second simulator family ($p < .001$).

The diagnostics show where the gap lives. The baselines recommend immediately (FRT ≈ 0) and keep asking questions at a high rate (PAR .40) — they interleave asking with showing rather than gathering before recommending. What fails is keeping that activity productive: their conversations stop yielding new preference information, and they stall at more than double the pipeline's rate (WT 1.85 and 1.78 vs. 0.66; both $p < .001$ against the matched configuration). The gap has a price in calls: per turn, the pipeline issues a median of nine LLM calls against two for ReAct and three for the Reflexion-style loop. We did not instrument token usage, so we report call counts rather than cost; whether the quality gap justifies the added calls depends on deployment economics we do not measure.

6. Discussion and Limitations

The three results share a shape. What moved outcomes was never how much information the agent had, but the form it arrived in and whether the agent could act on it. Runtime image injection gave the agent a genuinely new signal — pixels it had not seen before — and did not improve success. Removing the structured rendering of the visual attributes removed no information from the session, since the same content remained in each item's prose description and in the semantic index, and it cost 11.0 points of success rate. The agent, in other words, uses visual information *symbolically*: it acts on attributes it can name, match, and filter on, rather than on what it can see.

The retrieval result reads the same way. `inspect` returns no items and no new facts about them; it shows the words the catalog already uses and how many items sit behind each of them, so a preference stated in the shopper's words can be turned into a query in the catalog's, and the agent can tell what a query will find before sending it. The ablation removes both abilities at once, and each leaves its own trace in the tool logs (Section 5.1): calls rejected for using words the catalog does not know, and calls that use valid words but find nothing. This one tool was worth more success rate than any other capability we measured, and it is inexpensive: a facet-value

distribution query over a product database any store already maintains. Our estimate of its value is conservative, because the ablated agent kept reactive vocabulary correction through `filter`'s error messages — the measured effect is a floor, not a ceiling. We have not tested other catalogs, but the mechanism suggests where the value of inspection should be larger or smaller. The translation ability should matter more the wider the gap between the shopper's words and the catalog's own vocabulary. That gap is wide where the catalog labels judgements about appearance with a taxonomy of its own — here, "feels like silk" must become `satin-silk-like` — and narrow where shoppers already speak the catalog's terms, as with colours or the technical specifications of electronics. The ability to see what the catalog holds should matter more the richer the attribute schema: the more dimensions and values a catalog records, the more ways a query can name a combination no item satisfies, and the harder it is to guess blindly what exists.

The comparison with single-loop agents locates the difference in the same place. The baselines were not less active: they asked questions at a higher rate than the pipeline and recommended sooner. What they lacked was a way to keep that activity productive, and they stalled at more than double the pipeline's rate once their conversations stopped yielding new preference information. Finally, the ordering of all these effects survives a change of simulator LLM family — including a same-family pairing under which the Reference configuration scores *lower* than in the primary configuration — so what we report is not an artifact of one simulator's disposition.

These findings are bounded by scope choices, each made to keep the comparisons clean. The study covers one domain, one backbone, one kind of simulated shopper, and two design decisions about how the catalog reaches the agent; the rest of the design (e.g. dialogue policy, preference representation) is held fixed, so the hierarchy we report orders these choices rather than the full set of choices a shopping agent entails. Fashion over a feature-rich catalog is the setting where structured access and visual extraction have the most to offer, and domains with sparse metadata or weak visual signal may reward different investments. The single mid-scale backbone isolates design effects from model scale; the dominant finding operates on the agent's inputs rather than on its generation quality, which is what makes it the most likely to transfer, though transfer remains to be shown and is the object of ongoing work [28]. The simulator models a focused, cooperative shopper, and, while effects replicate across two simulator LLM families, behavioural variety — the uncertain and passive users proposed in [29, 20, 21] — and validation against human shoppers remain future work. Finally, the *w/o Structured Visual Attributes* ablation measures representation form with prose coverage retained; whether offline enrichment beats no enrichment at all is a separate question we did not test.

7. Conclusion

We ablated an agentic fashion shopping assistant one capability at a time and found a sharp hierarchy: structured, inspectable catalog access dominates — with most of its value in a simple tool that lets the agent learn the catalog's vocabulary and see what it holds, a capability no prior work had isolated for retrieval — while visual information contributes through structured offline-extracted attributes, with runtime image injection adding no measurable success on top. The orchestrated pipeline beats single-loop agents by sustaining elicitation, and all conclusions survive swapping the simulator's LLM family.

For teams building conversational shopping agents over feature-rich catalogs, these results order the investments. Catalog inspection comes first: a tool returning facet-value distributions is trivial to implement over any product database, gives the agent both the catalog's language and a view of what it holds, and was worth more success rate here than any other capability we measured. Visual attributes should be extracted offline — a one-time batch job — and surfaced as structured, filter-aligned fields rather than buried in captions or descriptions; once that is in place, runtime image injection can be skipped for mid-scale backbones, where we have not measured any quality it adds. And a single tool-calling loop, if chosen for cost reasons, should be

engineered to avoid stalling: in our study, such agents recommended early and asked questions at a high rate, but ran out of productive elicitation far sooner than the orchestrated pipeline.

The general lesson we take for e-commerce agents is about *form*: what moved every needle here was putting catalog knowledge — its vocabulary, its contents, and its visual attributes alike — into structures the agent can align with its actions, rather than adding signal or sophistication at inference time.

References

[1] Q. Peng, H. Liu, H. Huang, J. Yang, Q. Yang, M. Shao, A survey on LLM-powered agents for recommender systems, in: C. Christodoulopoulos, T. Chakraborty, C. Rose, V. Peng (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025, Association for Computational Linguistics, 2025, pp. 11574–11583. URL: https://doi.org/10.18653/v1/2025.findings-emnlp.620. doi:10.18653/V1/2025.FINDINGS-EMNLP.620.

[2] A. Wagne, T. Kolb, A. Banerjee, F. Nazary, J. Neidhardt, Y. Deldjoo, Conversational Recommender Systems Using Generative Models (Gen-CRS): A Literature Review, 2025. doi:10.13140/RG.2.2.23233.62564.

[3] R. Y. Maragheh, Y. Deldjoo, The future is agentic: Definitions, perspectives, and open challenges of multi-agent recommender systems, CoRR abs/2507.02097 (2025). URL: https://doi.org/10.48550/arXiv.2507.02097. doi:10.48550/ARXIV.2507.02097. arXiv:2507.02097.

[4] L. Friedman, S. Ahuja, D. Allen, Z. Tan, H. Sidahmed, C. Long, J. Xie, G. Schubiner, A. Patel, H. Lara, B. Chu, Z. Chen, M. Tiwari, Leveraging large language models in conversational recommender systems, CoRR abs/2305.07961 (2023). URL: https://doi.org/10.48550/arXiv.2305.07961. doi:10.48550/ARXIV.2305.07961. arXiv:2305.07961.

[5] X. Huang, J. Lian, Y. Lei, J. Yao, D. Lian, X. Xie, Recommender AI agent: Integrating large language models for interactive recommendations, ACM Trans. Inf. Syst. 43 (2025) 96:1–96:33. URL: https://doi.org/10.1145/3731446. doi:10.1145/3731446.

[6] Y. Liu, Y. Wang, L. Sun, P. S. Yu, Rec-GPT4V: Multimodal recommendation with large vision-language models, CoRR abs/2402.08670 (2024). URL: https://doi.org/10.48550/arXiv.2402.08670. doi:10.48550/ARXIV.2402.08670. arXiv:2402.08670.

[7] H. Jeon, S. Koide, Y. Wang, Z. He, J. J. McAuley, Adapting large vision-language models to visually-aware conversational recommendation, in: L. Antonie, J. Pei, X. Yu, F. Chierichetti, H. W. Lauw, Y. Sun, S. Parthasarathy (Eds.), Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025, ACM, 2025, pp. 1037–1048. URL: https://doi.org/10.1145/3711896.3736828. doi:10.1145/3711896.3736828.

[8] J. Ye, Y. Jiang, X. Wang, Y. Li, Y. Li, P. Xie, F. Huang, ProductAgent: Benchmarking conversational product search agent with asking clarification questions, in: S. Potdar, L. M. Rojas-Barahona, S. Montella (Eds.), Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025 - Industry Track, Suzhou, China, November 4-9, 2025, Association for Computational Linguistics, 2025, pp. 383–398. URL: https://doi.org/10.18653/v1/2025.emnlp-industry.25. doi:10.18653/V1/2025.EMNLP-INDUSTRY.25.

[9] D. Tran, Y. Li, H. Clay, N. Golrezaei, S. Beygi, A. Saberi, Entropy guided diversification and preference elicitation in agentic recommendation systems, CoRR abs/2603.11399 (2026). URL: https://doi.org/10.48550/arXiv.2603.11399. doi:10.48550/ARXIV.2603.11399. arXiv:2603.11399.

[10] T. Kim, J. Lee, S. Yoon, S. Kim, D. Lee, Towards personalized conversational sales agents: Contextual user profiling for strategic action, in: C. Christodoulopoulos, T. Chakraborty, C. Rose, V. Peng (Eds.), Findings of the Association for Computational Linguistics: EMNLP

2025, Suzhou, China, November 4-9, 2025, Association for Computational Linguistics, 2025, pp. 5131–5154. URL: https://doi.org/10.18653/v1/2025.findings-emnlp.275. doi:10.18653/V1/2025.FINDINGS-EMNLP.275.

[11] Q. Chen, T. Zhang, M. Nie, Z. Wang, S. Xu, W. Shi, Z. Cao, Fashion-GPT: Integrating LLMs with fashion retrieval system, in: Z. Wang, C. Long, S. Xu, B. Gan, W. Shi, Z. Cao, T. Chua (Eds.), Proceedings of the 1st Workshop on Large Generative Models Meet Multimodal Applications, LGM3A 2023, Ottawa ON, Canada, 2 November 2023, ACM, 2023, pp. 69–78. URL: https://doi.org/10.1145/3607827.3616844. doi:10.1145/3607827.3616844.

[12] P. Zhou, C. Liu, J. Ren, X. Zhou, Y. Xie, M. Cao, Z. Rao, Y. Huang, D. Chong, J. Liu, J. B. Kim, S. Wang, R. C. Wong, S. Kim, When large vision language models meet multimodal sequential recommendation: An empirical study, in: G. Long, M. Blumestein, Y. Chang, L. Lewin-Eytan, Z. H. Huang, E. Yom-Tov (Eds.), Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025, ACM, 2025, pp. 275–292. URL: https://doi.org/10.1145/3696410.3714764. doi:10.1145/3696410.3714764.

[13] K. Pang, X. Zou, W. Wong, FashionM3: Multimodal, multitask, and multiround fashion assistant based on unified vision-language model, CoRR abs/2504.17826 (2025). URL: https://doi.org/10.48550/arXiv.2504.17826. doi:10.48550/ARXIV.2504.17826. arXiv:2504.17826.

[14] L. Wang, J. Zhang, H. Yang, Z. Chen, J. Tang, Z. Zhang, X. Chen, Y. Lin, H. Sun, R. Song, X. Zhao, J. Xu, Z. Dou, J. Wang, J. Wen, User behavior simulation with large language model-based agents, ACM Trans. Inf. Syst. 43 (2025) 55:1–55:37. URL: https://doi.org/10.1145/3708985. doi:10.1145/3708985.

[15] K. Balog, C. Zhai, User simulation for evaluating information access systems, Found. Trends Inf. Retr. 18 (2024) 1–261. URL: https://doi.org/10.1561/1500000098. doi:10.1561/1500000098.

[16] H. Cai, Y. Li, W. Wang, F. Zhu, X. Shen, W. Li, T. Chua, Large language models empowered personalized web agents, in: G. Long, M. Blumestein, Y. Chang, L. Lewin-Eytan, Z. H. Huang, E. Yom-Tov (Eds.), Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025, ACM, 2025, pp. 198–215. URL: https://doi.org/10.1145/3696410.3714842. doi:10.1145/3696410.3714842.

[17] L. Zhu, X. Huang, J. Sang, How reliable is your simulator? analysis on the limitations of current LLM-based user simulators for conversational recommendation, in: T. Chua, C. Ngo, R. K. Lee, R. Kumar, H. W. Lauw (Eds.), Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024, ACM, 2024, pp. 1726–1732. URL: https://doi.org/10.1145/3589335.3651955. doi:10.1145/3589335.3651955.

[18] L. Zhu, X. Huang, J. Sang, A LLM-based controllable, scalable, human-involved user simulator framework for conversational recommender systems, in: G. Long, M. Blumestein, Y. Chang, L. Lewin-Eytan, Z. H. Huang, E. Yom-Tov (Eds.), Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025, ACM, 2025, pp. 4653–4661. URL: https://doi.org/10.1145/3696410.3714858. doi:10.1145/3696410.3714858.

[19] S. Kim, T. Kim, K. Seo, J. Yeo, D. Lee, Stop playing the guessing game! target-free user simulation for evaluating conversational recommender systems, CoRR abs/2411.16160 (2024). URL: https://doi.org/10.48550/arXiv.2411.16160. doi:10.48550/ARXIV.2411.16160. arXiv:2411.16160.

[20] A. Petruzzelli, A. F. M. Martina, C. Musto, M. de Gemmis, P. Lops, G. Semeraro, Beyond the lookup: Simulating realistic user uncertainty for the evaluation of conversational agentic recommenders, Information Systems Frontiers (2026). URL: https://doi.org/10.1007/s10796-026-10787-3. doi:10.1007/s10796-026-10787-3.

[21] A. Petruzzelli, A. F. M. Martina, C. Musto, M. de Gemmis, P. Lops, G. Semeraro, Simulating diverse user behavioral stereotypes for evaluating agentic conversational recommenders,

in: Proceedings of the 20th ACM Conference on Recommender Systems, RecSys 2026, Minneapolis, MN, USA, September 27 - October 2, 2026, ACM, 2026. doi:10.1145/3773078.3831840.

[22] J. Fang, S. Gao, P. Ren, X. Chen, S. Verberne, Z. Ren, A multi-agent conversational recommender system, CoRR abs/2402.01135 (2024). URL: https://doi.org/10.48550/arXiv.2402.01135. doi:10.48550/ARXIV.2402.01135. arXiv:2402.01135.

[23] Y. Zu, X. Cai, C. Yu, H. Peng, J. Duan, L. Chen, K. Wang, Z. Zhang, C. Peng, Z. Lin, C. Law, Think then recommend: An LLM-Powered multi-agent framework for personalized conversational recommender system in E-Commerce, in: H. Hacid, Y. Maarek, F. Bonchi, I. Guy, E. Yilmaz (Eds.), Proceedings of the ACM Web Conference 2026, WWW 2026, Dubai, United Arab Emirates, originally scheduled for April 13-17, 2026, rescheduled for June 29 - July 3, 2026, ACM, 2026, pp. 9160–9168. URL: https://doi.org/10.1145/3774904.3793050. doi:10.1145/3774904.3793050.

[24] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, Y. Cao, ReAct: Synergizing reasoning and acting in language models, in: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X.

[25] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, S. Yao, Reflexion: language agents with verbal reinforcement learning, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

[26] C. G. Ling, ElizabethHMGroup, FridaRim, inversion, J. Ferrando, Maggie, neuraloverflow, xlsrln, H&M Personalized Fashion Recommendations, https://kaggle.com/competitions/h-and-m-personalized-fashion-recommendations, 2022. Kaggle.

[27] A. Panickssery, S. R. Bowman, S. Feng, LLM evaluators recognize and favor their own generations, in: A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, C. Zhang (Eds.), Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024. URL: http://papers.nips.cc/paper_files/paper/2024/hash/7f1f0218e45f5414c79c0679633e47bc-Abstract-Conference.html.

[28] A. F. M. Martina, A design-space approach to robust agentic conversational recommendation, in: Proceedings of the 20th ACM Conference on Recommender Systems, RecSys 2026, Minneapolis, MN, USA, September 27 - October 2, 2026 (Doctoral Symposium), ACM, 2026. doi:10.1145/3773078.3831946.

[29] A. F. M. Martina, A. Petruzzelli, C. Musto, M. de Gemmis, P. Lops, G. Semeraro, Distillrecdial: A knowledge-distilled dataset capturing user diversity in conversational recommendation, in: Proceedings of the Nineteenth ACM Conference on Recommender Systems, 2025, pp. 726–735.