DPLite-CRS: Towards Dynamic and Efficient Conversational Recommendation

Yongsen Wei
School of Physics and Information Engineering, Guilin Normal College, Guilin 541199, China

Abstract: Conversational recommender systems must incorporate new preferences while retaining relevant dialogue history and controlling response cost. These requirements motivate a joint design of preference tracking and computation allocation. This vision paper proposes DPLite-CRS, a framework combining incremental preference updates, parameter-efficient adaptation, and adaptive recommendation routing. A compact state is up-dated from each new dialogue turn; low-rank adapters support domain adaptation; and a drift-aware controller selects between direct item retrieval and a more expensive contextual reranking path. The central hypothesis is that computation should depend on the evidence needed at each turn rather than follow a fixed processing path. We outline an illustrative formulation, identify failure cases, and propose an evaluation protocol covering ranking quality, preference revision, latency, and memory. The framework remains a research proposal: its effectiveness and deployment benefits require empirical validation.

Keywords: Terms—conversational recommendation; Preference tracking; Parameter-efficient adaptation; Adaptive inference; Vision paper

1. Introduction

Conversational recommender systems (CRS) elicit preferences and revise recommendations through natural-language interaction [1]. A movie seeker may first request a thriller, then exclude violent content, and later prioritize a short running time. A useful system must retain compatible constraints, revise conflicting ones, and rank items using only information available at the current turn. Preference clarification and preference change can coexist; neither implies that the user's underlying interests must eventually converge.

This setting raises a practical question: how much computation is necessary to incorporate each new piece of preference evidence? Reprocessing an expanding dialogue can preserve access to context but incur repeated work. Maintaining a compact state can reduce that work but discard details needed for later corrections. Meanwhile, parameter-efficient adaptation can reduce training overhead without necessarily reducing the cost of each backbone forward pass. These are related design choices, but improvements in one dimension do not establish improvements in the others.

We propose DPLite-CRS, a Dynamic Preference-aware Lightweight Conversational Recommendation framework. Its research premise is to coordinate three decisions: what preference information to retain, which parameters to adapt, and when additional contextual processing is justified. The framework combines gated state updates, low-rank adaptation, and a controller that routes turns between direct retrieval and contextual reranking.

The contribution of this vision paper is a concrete design and a falsifiable research agenda. We

specify an illustrative update and routing mechanism, distinguish adaptation efficiency from inference efficiency, and define comparisons that could establish or reject the proposed benefits. We do not claim a new gating or adaptation algorithm, proven memory preservation, or measured performance gains. The intended contribution lies in studying these components jointly under conversational quality and resource constraints.

2. Background and Design Motivation

2.1 Contextual Recommendation

Existing CRS already exploit dialogue context. KBRD connects recommendation and response generation through knowledge-based representations [2]; UniCRS uses knowledge-enhanced prompts and a fixed pretrained language model to integrate the two tasks [3]. A fixed backbone therefore does not imply a fixed user representation. The question addressed here is narrower: whether a persistent preference state can support accurate turn-by-turn ranking while limiting repeated contextual computation.

2.2 Efficient Adaptation and Processing

LoRA freezes pretrained weights and learns low-rank updates [4]. TALLRec studies efficient adaptation of language models to recommendation [5], motivating their use as an adaptation component rather than establishing conversational state tracking. Training fewer parameters reduces adaptation requirements, but the backbone still participates in inference. Consequently, DPLite-CRS treats parameter-efficient adaptation and adaptive processing as separate mechanisms whose resource effects must be measured separately.

2.3 Research Gap

The proposed research concerns the interaction among state compression, adaptation capacity, and per-turn processing cost. An inexpensive update may miss a negation, and a routing decision may overlook an important revision. DPLite-CRS therefore tests the combined quality–cost trade-off through matched component comparisons.

Three requirements guide the design. First, the system should preserve an explicit processing boundary: a newly available turn can update the state without routinely re-encoding the full dialogue. Second, expensive computation should have a defined purpose, such as resolving a reference or checking an earlier constraint. Third, resource accounting should cover the complete recommendation pipeline. A smaller trainable module is useful for adaptation, but a system remains expensive if encoding or retrieval dominates each turn.

These requirements motivate a compact state with recoverable evidence, rather than assuming that one representation can retain every detail.

3. Proposed Dplite-CRS Framework

3.1 Task and Processing Boundary

Let $C_t=(u_1,\ldots,u_t)$ denote the visible dialogue prefix at a recommendation point, including speaker roles, and let V denote the item catalog. The goal is to return a ranked list from V using C_t. The current target recommendation and subsequent user feedback are excluded from the input. We focus on item ranking; response generation can consume the resulting list but is outside the present design.

The proposed system maintains a preference state $\mathbf{h}_t \in \mathrm{R}^d$. A turn encoder processes the newly observed turn together with its speaker role and, if needed, a fixed-size local context window. This boundary matters: feeding the complete growing prefix into the encoder at every turn would undermine the proposed reduction in repeated processing. A separately stored dialogue archive remains available to the expensive path for recovering omitted evidence.

Figure 1 summarizes the proposed data flow. The state supports inexpensive ranking, while the archive preserves access to evidence that state compression may omit. Retrieval and reranking do not modify the user state merely because a candidate receives a high score; only newly observed dialogue provides further preference evidence.

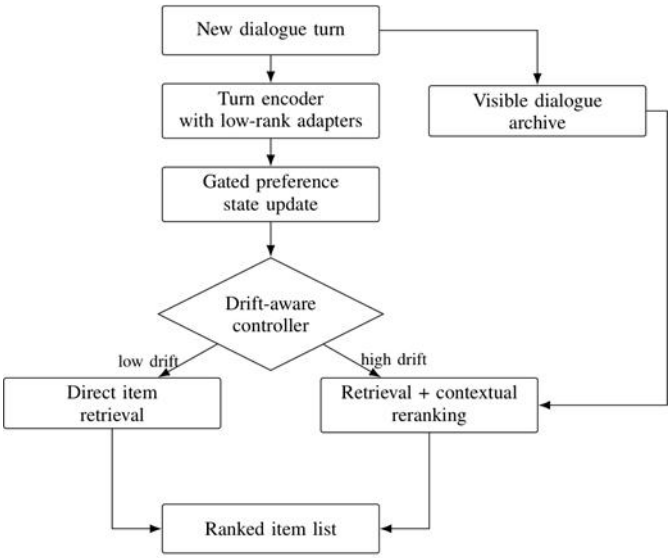


Figure 1: Proposed architecture. Both paths use a shared item catalog. Archive access is limited to information already observed at the current turn.

3.2 Incremental Preference Updates

An illustrative gated update is

$$\mathbf{z}_t = E_{\theta,\phi}(u_t) \tag{1}$$

$$\mathbf{g}_t = \sigma(W_g[\mathbf{h}_{t-1};\mathbf{z}_t] + \mathbf{b}_g) \tag{2}$$

$$\widetilde{\mathbf{h}}_t = \tanh(W_h[\mathbf{h}_{t-1};\mathbf{z}_t] + \mathbf{b}_h) \tag{3}$$

$$\mathbf{h}_t = (\mathbf{1}\text{-}\mathbf{g}_t) \odot \mathbf{h}_{t-1} + \mathbf{g}_t \odot \widetilde{\mathbf{h}}_t \tag{4}$$

where $E_{\theta,\phi}$ is a pretrained encoder with backbone parameters θ and adaptation parameters ϕ, $[\cdot;\cdot]$ denotes concatenation, and $\odot$ denotes elementwise multiplication. The initial state can be zero. Equations (1)–(4) specify one possible implementation, not an implemented or validated model.

The gate determines how strongly new evidence modifies the state. For example, "avoid horror" should revise an earlier request for a frightening film while retaining an unrelated running-time constraint. Learned coordinates, however, need not correspond to interpretable attributes. Gating alone provides no guarantee of semantic selectivity or long-term retention. Tests must therefore distinguish successful constraint revision from simple recency bias and compare the update against ungated recurrent and context-encoding alternatives.

Consider a hypothetical exchange: a seeker requests a thriller under two hours, later excludes violent content, and then asks for "something similar but lighter." The first turn establishes multiple

constraints; the second revises content tolerance; the third introduces a reference that may require earlier evidence. The desired behavior is to preserve the duration constraint throughout. This example illustrates why drift and processing difficulty can differ: an explicit revision may be easy to encode, whereas a small state change may hide an unresolved reference. It is a diagnostic scenario, not evidence that the proposed gate or router succeeds.

3.3 Parameter-Efficient Domain Adaptation

For a selected backbone matrix $W_0 \in \mathrm{R}^{m\times n}$, a LoRA-style update takes the form.

$$W = W_0 + \frac{\alpha}{r} BA, \quad A \in \mathrm{R}^{r\times n}, \quad B \in \mathrm{R}^{m\times r} \tag{5}$$

where $r \ll \min(m,n)$ and α controls scaling. Training updates the low-rank factors and the state-update and ranking modules while freezing the backbone. Candidate item embeddings can be precomputed once the encoder is fixed for deployment.

This component targets adaptation memory and trainable parameter count. It does not remove the backbone computation in (1). A large encoder could dominate latency even when reranking is rarely triggered. The backbone size, adapter rank, item-index storage, and online encoding cost must therefore be included in any claim of lightweight operation.

3.4 Drift-Aware Recommendation Routing

The direct path scores catalog items using the current state, for example through $s_f(t,v) = \mathbf{h}_t^\top \mathbf{e}_v$, where $\mathbf{e}_v$ is an item embedding in the same space. An approximate nearest- neighbor index can retrieve a candidate list. The contextual path additionally retrieves relevant historical evidence or item knowledge and applies a more expensive reranked. Both paths use the same catalog and return the same output format.

A simple routing signal is the change between consecutive states:

$$\delta_t = 1 - \frac{\mathbf{h}_t^\top \mathbf{h}_{t-1}}{\max(\|\mathbf{h}_t\|_2 \|\mathbf{h}_{t-1}\|_2, \epsilon)} \tag{6}$$

For $t>1$, contextual reranking is triggered when $\delta_t > \tau$; the first turn uses a predefined initialization policy. The threshold τ is selected on validation data under a deployment budget. This rule is a starting hypothesis, since a large state change need not require additional reasoning and a small change can conceal unresolved ambiguity. Uncertainty-aware routing and periodic contextual checks are alternatives to test if drift alone is unreliable.

Let c_u be the shared encoding and update cost, c_f the direct retrieval cost, c_s the additional contextual-path cost, and p_s its activation probability. An accounting approximation is

$$\mathrm{E}[c_t] = c_u + c_f + p_s c_s \tag{7}$$

This decomposition identifies where savings could arise; it is not a latency guarantee. Retrieval, archive access, routing overhead, and hardware contention must be included in measured cost. Benefits depend on avoiding sufficient contextual work without excessive ranking degradation.

Under the simplified assumption that shared costs are identical, always-contextual processing costs $c_u + c_f + c_s$ per turn. Selective routing would avoid an expected $(1-p_s)c_s$ of contextual work before accounting for controller overhead. If that overhead is c_r, a necessary condition for a net saving in this accounting model is $(1-p_s)c_s > c_r$. This condition explains why frequent switching or a costly detector can eliminate the intended benefit. It also motivates reporting activation rates together with measured latency rather than treating a low activation rate as sufficient evidence of efficiency.

3.5 Learning and Turn-Level Operation

An initial implementation could train the encoder adapters, state updater, and direct scorer with a next-item ranking objective on visible prefixes. The contextual reranked would use the same target definition. A validation sweep would then select routing thresholds and adapter ranks; no test feedback would influence these choices. This staged procedure provides a simple reference before considering joint controller training.

For a single target y_t associated with a visible prefix, an illustrative direct-path objective is

$$L_{\text{rank}}(t) = -\log \frac{\exp s_f(t, y_t)}{\sum_{v \in V} \exp s_f(t, v)} \tag{8}$$

Training would backpropagate through the state updates and adapter parameters while leaving backbone weights fixed. If sampled candidates replace the full denominator, the sampling rule and candidate count must be documented. The contextual scorer can be trained on candidates retrieved by the direct path, with training and inference candidate construction kept consistent. A reranked cannot recover a target absent from its candidate list; candidate recall must therefore be reported separately from reranking quality.

The archive interface should also be bounded. A prototype could retrieve at most b prior turns and rerack at most M candidates, with both limits selected using validation data. Their effects should be examined alongside the routing threshold. Increasing b may recover missing constraints but increase encoder work; increasing M may improve candidate coverage but increase reranking cost. Fixed limits make these trade- offs observable without implying constant total storage: the archived dialogue still grows with the interaction.

At inference, the system encodes each new turn, updates the state, retrieves candidates, and conditionally reracks them. Subsequent user feedback enters only when it becomes observable. The recommender's own suggestion is not treated as evidence that the user likes an item. The archive and item index are shared resources whose storage and retrieval costs must be reported alongside the compact state.

4. Research Agenda and Evaluation Plan

4.1 Testable Hypotheses

Table I translates the design into three research questions. Each concerns a distinct claimed benefit, allowing the combined framework to fail in one dimension even if it succeeds in another.

Table 1: Proposed hypotheses and their evaluation. No results are reported.

Hypothesis	Required evidence
H1: Incremental states retain useful constraints.	Compare ranking quality and constraint violations against full-prefix, bounded-window, and ungated recurrent encoders.
H2: Low-rank adaptation is sufficient.	Compare quality, trainable parameters, and peak training memory against frozen and fully fine-tuned versions of the same backbone.
H3: Routing allocates computation effectively.	Compare quality–latency curves against always-direct, always-contextual, and random routing at matched contextual activation rates.

4.2 Data and Ranking Protocol

Movie recommendation offers a suitable initial setting. ReDial [6] can support ranking evaluation, supplemented by a manually checked subset of dialogues containing explicit revisions, negation, or references to earlier preferences. If controlled edits are introduced to increase revision coverage, they should be labeled as constructed examples, checked for consistency, and reported separately from naturally occurring dialogue.

Splits should be disjoint at the dialogue level. For each query, construct the visible prefix before the target recommendation and rank against a fixed catalog. Specify target selection, entity mapping, and whether the task is single-target or multi-target. Report Recall@K and NDCG@K, aggregate consistently across queries, and use dialogue-level resampling for confidence intervals. Fixed-dialogue ranking can evaluate state tracking, but cannot establish improvements in interactive user satisfaction or conversation length.

Beyond aggregate ranking, a checked diagnostic subset should distinguish retention from revision. For constraints with verifiable catalog attributes, report the fraction of top-K recommendations violating each currently active constraint. After a correction, evaluate the replacement constraint while checking whether unrelated constraints remain satisfied. State- vector distance is unsuitable as a standalone measure of either behavior, since representations can change without corresponding changes in item rankings. Human annotation should resolve ambiguous references and distinguish temporary requests from explicit preference reversals. Attribute coverage and annotation agreement should accompany these diagnostic results.

4.3 Efficiency and Ablations

Measure median and 95th-percentile turn latency, peak inference memory, trainable parameters, training memory, and contextual-path activation rate. Use identical hardware, precision, batch size, candidate pools, and retrieval settings for matched comparisons. Report preprocessing and indexing separately, and include archive and index storage. State whether contextual processing includes external-service latency.

Ablations should remove the gate, replace low-rank adaptation with frozen or full tuning, and force either recommendation path. Threshold sweeps should show the quality– latency frontier rather than a single favorable operating point. A random router with the same activation rate tests whether preference drift identifies useful turns beyond merely reducing expensive calls. Long-prefix and preference-reversal subsets test whether apparent savings come from forgetting difficult evidence.

For a deployment latency budget B, validation should select the configuration with the highest ranking quality among those satisfying the budget. The selected configuration is then evaluated once on the held-out test split. Comparisons should also fix an allowable quality loss and measure the resulting resource reduction. Report results across multiple training seeds and separate short from long prefixes. This procedure tests whether routing offers a useful operating range rather than an isolated threshold advantage. A later interactive study could measure acceptance and turns to a satisfactory recommendation, but offline ranking results alone should not be used to infer those benefits.

4.4 Failure Cases and Deployment Boundaries

Compression errors: A compact state may lose rare but decisive constraints. Archive retrieval

offers a recovery mech- anism, but its success depends on the stored evidence and retrieval query. Retention should be measured through later- turn ranking and explicit constraint checks, not inferred from gate values.

Routing errors: Noise can trigger unnecessary contextual calls, whereas unresolved ambiguity can leave the state nearly unchanged. Evaluation should report both activation cost and quality loss relative to always-contextual processing. If the drift signal is ineffective, the routing premise requires revision.

Adaptation and domain limits: Adapter rank and back- bone capacity may interact with dialogue length and domain shift. Results from movie ranking would not establish suitability for education or clinical decision support. Edge deployment also requires separate memory, energy, and connectivity measurements. The immediate research priority is to validate the proposed quality–cost trade-off in one well-defined setting.

5. Conclusion

DPLite-CRS proposes coordinating incremental preference tracking, low-rank adaptation, and drift-aware recommendation routing. Its central question is whether a compact state and selective contextual processing can preserve ranking quality while reducing resource use. The framework offers an illustrative design and a controlled evaluation plan, rather than evidence of superiority. The next step is to implement matched baselines and test when state compression and adaptive routing help, when they fail, and whether their savings remain after accounting for online encoding, retrieval, and storage.

References

[1] D. Jannach, A. Manzoor, W. Cai, and L. Chen, "A survey on conversational recommender systems," ACM Computing Surveys, vol. 54, no. 5, 2021, doi: 10.1145/3453154.

[2] Q. Chen, J. Lin, Y. Zhang, M. Ding, Y. Cen, H. Yang, and J. Tang, "Towards knowledge-based recommender dialog system," in Proc. EMNLPIJCNLP, 2019, pp. 1803–1813, doi: 10.18653/v1/D19-1189.

[3] X. Wang, K. Zhou, J.-R. Wen, and W. X. Zhao, "Towards unified conversational recommender systems via knowledge-enhanced prompt learning," in Proc. ACM SIGKDD, 2022, doi: 10.1145/3534678.3539382.

[4] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models,"in Proc. ICLR, 2022. [Online]. Available: https://arxiv.org/abs/2106.09685

[5] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He, "TALLRec: An effective and efficient tuning framework to align large language model with recommendation," in Proc. ACM RecSys, 2023, pp. 1007–1014, doi: 10.1145/3604915.3608857.

[6] R. Li, S. Ebrahimi Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal, "Towards deep conversational recommendations," in Advances in Neural Information Processing Systems, vol. 31, 2018.