

# QDAR: Query-Guided Distribution-Aware Recommendation for Efficient Intent-Centric Modeling

Abhishek Abhishek\*, P. S. Bishnu, V. Bhattacharjee

Department of Computer Science and Engineering, BIT Mesra, Ranchi, India

\*phdcs10012.24@bitmesra.ac.in, psbishnu@bitmesra.ac.in  
vbhattacharjee@bitmesra.ac.in

## Abstract

Recent work in sequential recommendation has focused on Transformer-based architectures for their ability to capture long-range dependencies. However, these models face two critical bottlenecks in real-world deployment: quadratic computational complexity  $O(n^2)$  and an inability to explicitly filter session-irrelevant noise in high-volatility user histories. In this paper, we propose the Query-Guided Distribution-Aware Recommender (QDAR), a lightweight architecture designed for high-precision, low-latency recommendation. QDAR introduces a Query-Guided Interaction Selector that prunes interaction histories in linear  $O(n)$  time, focusing exclusively on signals relevant to the current user intent and improving representation robustness. We further replace standard dot-product similarity with a Distribution-Aware Similarity (DAS) metric that accounts for feature covariance via a learnable diagonal Mahalanobis distance. Empirical evaluations on three benchmarks (MovieLens-1M, RetailRocket, and Amazon Beauty) with multi-seed evaluation (seeds 42, 43, 44) and 95% confidence intervals show that QDAR consistently reduces inference latency across all datasets, while achieving its strongest ranking improvements on sparse e-commerce clickstream data (RetailRocket), where it substantially outperforms Transformer baselines across all ranking metrics. A detailed component-wise ablation study confirms the contribution of each architectural module.

**Keywords:** Sequential Recommendation, Efficient AI, Metric Learning, Query-Guided Selection

## 1. Introduction

Sequential recommender systems is becoming an integral part modern digital platforms [2, 3, 4]. Transformer-based models such as SASRec [2] and BERT4Rec [3] have become highly competitive because they can capture long-range dependencies in user interaction sequences. However, their quadratic time and memory complexity makes them expensive to deploy in latency-sensitive settings, especially when user histories are long or noisy [5, 6].

With the rise of mobile devices and Quick commerce, user behavior on online platforms has become far more dynamic and unpredictable. They often interact through short, fragmented sessions, with their interests shifting quickly from one moment to the next. In the query-aware recommendation settings, users may provide explicit search queries that clearly express their immediate intent, but these queries frequently align with only a small portion of their long-term interaction history [7, 9]. Traditional self-attention mechanisms treat all past interactions as equally important candidates for recommendation, which can introduce substantial noise when much of the historical behavior is not related to the user's current goal [10, 11]. Due to this, there is a growing need for sequential recommendation models that can selectively focus on interactions relevant to the current query while remaining computationally efficient.

We propose **QDAR (Query-Guided Distribution-Aware Recommender)**, a lightweight recommendation architecture designed around two key ideas. First, the model selectively attends only to interactions that are relevant to the user’s current query. Second, it measures similarity within a distribution-aware latent space that captures feature importance and covariance structure more effectively than standard similarity functions. QDAR introduces a Query-Guided Interaction Selector that computes query–item relevance in linear time and retains only the top-(k) most relevant interactions, significantly reducing the amount of history that must be processed. In addition, the model replaces conventional dot-product similarity with a Distribution-Aware Similarity (DAS) module based on a learnable diagonal Mahalanobis distance, enabling the recommender to emphasize more informative latent dimensions while preserving efficient ( $O(d)$ ) scoring complexity.

The objective of the work is to develop and evaluate a query-guided, distribution-aware sequential recommender that improves both recommendation quality and computational efficiency in intent-driven recommendation scenarios. Our contributions are threefold. First, we formulate a Query-Guided Interaction Selector that efficiently filters historical interactions in linear time. Second, we introduce the Distribution-Aware Similarity (DAS) module, which captures anisotropic latent feature structure using a learnable diagonal Mahalanobis distance. Third, we conduct extensive experiments on MovieLens-1M, RetailRocket, and Amazon Beauty datasets, demonstrating that QDAR consistently reduces inference latency while achieving particularly strong ranking improvements on sparse e-commerce clickstream data. We further perform a detailed ablation study showing that the selector mechanism, DAS scoring, aggregation strategy, and top-(k) truncation each contribute meaningfully to the model’s overall performance.

## 2. Related Work

Sequential recommendation has progressed from RNN-based models to Transformer-based architectures that capture long-range dependencies in user interaction histories. Early methods such as GRU4Rec used gated recurrent units to model session dynamics, while later Transformer-based approaches such as SASRec and BERT4Rec improved performance by leveraging self-attention over historical sequences [1, 2, 3]. However, the quadratic cost of full self-attention has motivated many recent efficiency-oriented designs, including pruning, distillation, and other sequence-compression strategies [5, 6, 21]. QDAR is positioned in this line of work as a lightweight alternative that avoids full-sequence attention by using query-guided filtering and a compact scoring stage.

Query-aware and intent-aware sequential recommendation has recently become an active research direction. Query-Aware Sequential Recommendation introduced explicit user queries as contextual signals for modeling intent and showed that incorporating query information can improve generalization in sequential recommendation settings [7, 8]. Query-SeqRec further demonstrated that query-item co-occurrence and heterogeneous sequence augmentation can enhance recommendation quality, especially when user intent changes rapidly [7]. Related work on sequence-aware query recommendation also supports the view that query signals are valuable for capturing short-term intent, particularly in sparse or volatile interaction settings [9]. In contrast to these models, QDAR uses the query primarily as a selector that prunes irrelevant history before scoring, rather than treating queries as a separate sequence stream.

Another relevant line of work focuses on metric learning and similarity functions for recommendation. Standard dot-product scoring assumes an isotropic latent space, which can be limiting when some embedding dimensions are more informative than others. Mahalanobis-based similarity is being analyzed in recommendation and related similarity-learning tasks because it captures feature covariance and anisotropic structure more flexibly than inner products

[18, 19, 20]. A diagonal Mahalanobis formulation is adapted by QDAR to preserve the benefit while keeping inference efficient and numerically stable. Use of this design choice allows the model to retain the computational simplicity of lightweight recommenders while improving ranking sensitivity in the latent space.

More studies and work are being done on efficient and on-device recommenders that trade a small amount of modeling complexity for substantial gains in speed and deployability. Recent works and trends have highlighted the importance of adapting sequential recommendation models to stricter latency and memory budgets, especially in production settings where inference cost matters as much as ranking quality [5, 6, 22]. Our proposed QDAR architecture follows this direction by combining selective interaction modeling with a lightweight similarity metric, making it well suited for CPU and Apple Silicon deployment. Taken together, these related lines of work motivate our design and highlight the gap QDAR aims to address: intent-aware sequential modeling that is both efficient and robust.

### 3. Problem Formulation

Let  $\mathcal{U}$  and  $\mathcal{I}$  denote the sets of users and items, respectively. For each user  $u \in \mathcal{U}$ , we observe a chronological interaction sequence

$$H_u = (i_1^u, i_2^u, \dots, i_{n_u}^u), \quad i_t^u \in \mathcal{I}.$$

At recommendation time, the user provides a query  $q$  that reflects the current intent. Given  $(H_u, q)$ , the task is to rank candidate items  $i \in \mathcal{I}$  for next-item prediction.

We use a leave-one-out temporal split, where all but the last two interactions are used for training, the second-last interaction is used for validation, and the last interaction is used for testing. Sequences are truncated or padded to a maximum length  $L$ . For datasets without explicit queries, we construct pseudo-queries from event types, item categories, or recent interaction context. The scoring function is defined as

$$s : \mathcal{H} \times \mathcal{Q} \times \mathcal{I} \rightarrow \mathbb{R},$$

where higher scores indicate higher recommendation relevance.

## 4. The QDAR Architecture

Figure 1 shows the overall architecture of QDAR, which consists of item and query embeddings, a Query-Guided Interaction Selector, a lightweight history aggregator, and a Distribution-Aware Similarity (DAS) module. The design filters irrelevant historical interactions early, builds a compact intent representation, and ranks candidate items using a lightweight distance-based scoring function.

### 4.1 Embedding Layer

Items and queries are embedded into a shared  $d$ -dimensional latent space:

$$e_{i_t} = E_I[i_t], \quad e_q = E_Q[q],$$

where  $E_I \in \mathbb{R}^{|\mathcal{I}| \times d}$  is the item embedding matrix and  $E_Q$  is the query embedding matrix. Sharing the same latent space allows the model to directly compare user intent and item representations

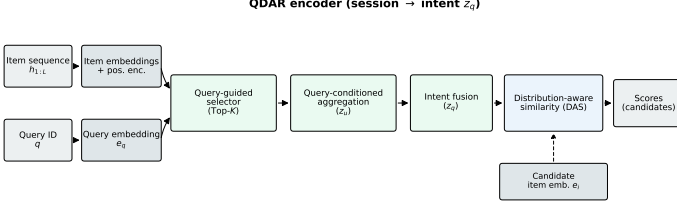


Figure 1: Architecture of the proposed QDAR model.

using compact vector operations. Categorical side information is optionally concatenated when available, which helps enrich the representation of sparse interactions.

## 4.2 Pseudo-Query Construction

Three strategies handle datasets without explicit queries. These pseudo-queries provide a practical way to inject intent-like context into datasets that only contain interaction logs.

- **Category-based:** Category of the most recent interaction as intent proxy (used for Amazon Beauty).
- **Event-type-based:** Last event type in RetailRocket (view, add-to-cart, transaction) combined with the most recent item category [14]. This is the strategy used for all RetailRocket results reported in Table 4 and Table 5.
- **Windowed context:** Average pooling of the last  $w$  item attributes (used for MovieLens-1M, where no category or event metadata is available).

With these constructions the QDAR operate consistently across datasets with different metadata richness while still preserving a notion of current intent.

## 4.3 Query-Guided Interaction Selector

Relevance scores are computed as:

$$r_t = \phi(e_q) \cdot \psi(e_{i_t})^\top, \quad (1)$$

where  $\phi(\cdot)$  and  $\psi(\cdot)$  are learnable linear projections. Top- $k$  interactions are selected:

$$\mathcal{S} = \text{TopK}(\{r_t\}_{t=1}^{n_u}, k), \quad H'_u = \{i_t^u : t \in \mathcal{S}\},$$

requiring  $O(n_u \cdot d)$  operations. This selector acts as an intent filter and it keeps only the interactions most relevant to the current query and removes older or off-topic behavior. In sparse and volatile sessions, thus helping reduce noise before aggregation.

## 4.4 Intent Aggregator

**Mean pooling:**

$$z_u = \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} e_{i_t}.$$

**Query-conditioned attention pooling:**

$$\alpha_t = \frac{\exp(f(e_{i_t}, e_q))}{\sum_{j \in \mathcal{S}} \exp(f(e_{i_j}, e_q))}, \quad (2)$$

$$z_u = \sum_{t \in \mathcal{S}} \alpha_t e_{i_t}. \quad (3)$$

The fused representation  $z$  is obtained by concatenating  $z_u$  and  $e_q$  followed by a linear layer. The mean variant is simple and stable, while the attention variant allows the model to emphasize more relevant selected interactions. This gives QDAR flexibility to balance efficiency and expressiveness depending on the dataset.

**4.5 Distribution-Aware Similarity (DAS)**

Let  $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_d^2)$  be a learnable diagonal covariance matrix. The DAS score is:

$$s(z, i) = -\sqrt{(z - e_i)^\top \Sigma^{-1} (z - e_i)}. \quad (4)$$

Because  $\Sigma$  is diagonal, its inverse is simply  $\Sigma^{-1} = \text{diag}(1/\sigma_1^2, \dots, 1/\sigma_d^2)$ , reducing the quadratic form to a weighted sum of squared differences:

$$(z - e_i)^\top \Sigma^{-1} (z - e_i) = \sum_{j=1}^d \frac{(z_j - e_{i,j})^2}{\sigma_j^2}.$$

This maintains  $O(d)$  scoring efficiency per candidate. Each  $\sigma_j^2$  is parameterized via a softplus function for numerical stability and positivity.

**4.6 Training Objective**

We use Bayesian Personalized Ranking (BPR). Given a training triple  $(u, q, i^+, i^-)$ :

$$\mathcal{L}_{\text{BPR}} = -\log \sigma(s(z, i^+) - s(z, i^-)) + \lambda \|\Theta\|_2^2,$$

where  $\Theta$  denotes all trainable parameters and  $\lambda$  is an  $\ell_2$  regularization coefficient. BPR is suitable here because the task is ranking-based rather than classification-based. It directly encourages positive items to receive higher scores than sampled negatives, which aligns well with next-item recommendation.

**5. Experimental Setup**

All experiments are run on a workstation with an Apple Silicon processor and 16 GB of unified memory, using the MPS[22] backend when available and falling back to CPU otherwise. To keep the comparison fair, all baselines use the same embedding size, optimizer, and regularization strategy, and no model relies on CUDA-specific optimizations that are not available on the MPS backend.

We use Adam as the default optimizer with a learning rate of  $10^{-3}$  and apply  $\ell_2$  weight decay of  $10^{-5}$ . The embedding dimension is set to  $d = 64$ . Models are trained for up to 50 epochs with early stopping based on validation NDCG@10, with a patience of 5 epochs. We report the mean and 95% confidence interval of all metrics over three random seeds (42, 43, 44) to account for variability due to random initialization and negative sampling.

For each dataset, the maximum sequence length  $L$  and selector size  $k$  are tuned on the validation set. Negative sampling is performed uniformly over the item set, excluding items already present in the user’s training sequence. We sample one negative per positive instance for BPR training.

### 5.1 Data Preprocessing

For each dataset, we first filter users and items with fewer than five interactions, sort interactions chronologically, and apply a leave-one-out temporal split, using the second-last interaction for validation and the last interaction for test. We then truncate or pad sequences to a maximum length  $L$ . For datasets without explicit queries, we construct pseudo-queries using the strategies described in Section 4.2.

### 5.2 Training and Evaluation Procedure

---

**Algorithm 1:** QDAR Training and Evaluation

---

- Input:** User-item interaction dataset  $D$ , embedding dimension  $d$ , selector size  $k$ , learning rate  $\eta$ , number of epochs  $E$ .
- Output:** Trained recommendation model  $M$ ; top- $k$  recommendation lists  $R_u$  for each user  $u$ .
1. **Data Preprocessing:** We filter out users and items with fewer than five interactions, sort all interactions chronologically, and apply a leave-one-out temporal split. The resulting sequences are then truncated or padded to a maximum length  $L$ , and pseudo-queries are constructed when explicit queries are not available.
  2. **Model Initialization:** Initialize item and query embeddings; initialize the query-guided selector and distribution-aware similarity parameters.
  3. **Training Objective:** For each training triple  $(u, q, i^+, i^-)$ , compute query-guided relevance scores, select top- $k$  interactions, aggregate the selected interactions into an intent representation, compute DAS scores for positive and negative items, and optimize the BPR loss.
  4. **Model Training:** For  $E$  epochs, sample training triples, update parameters with Adam, and apply early stopping based on validation NDCG@10.
  5. **Evaluation:** Rank all candidate items for each user, generate top- $k$  recommendation lists, and compute HR@10, NDCG@10, and MRR@10.
- 

## 6. Complexity Analysis

### 6.1 Time Complexity

A single self-attention layer in Transformer-based sequential recommenders requires  $O(n^2 \cdot d)$  time, where  $n$  is the sequence length and  $d$  is the embedding dimension [2, 5]. QDAR replaces full self-attention with a lightweight query-guided pipeline consisting of relevance computation, top- $k$  selection, aggregation, and distribution-aware scoring.

The relevance computation in Eq. 1 requires  $O(n \cdot d)$  time because each historical item is compared with the query in the shared latent space. Top- $k$  selection can be implemented with a heap in  $O(n \log k)$  time. Aggregation over the selected subset costs  $O(k \cdot d)$ , and DAS scoring per candidate in Eq. 4 costs  $O(d)$  due to the diagonal structure of  $\Sigma^{-1}$ .

Therefore, the total inference complexity of QDAR is

$$O(n \cdot d + n \log k + k \cdot d).$$

Since  $k$  is small relative to  $n$ , the dominant term is  $O(n \cdot d)$ , giving linear dependence on sequence length.

## 6.2 Space Complexity

Self-attention requires storing  $O(n^2)$  attention logits in addition to sequence embeddings, yielding a total memory cost of  $O(n^2 + nd)$ . By contrast, QDAR stores only item/query embeddings, relevance scores, and selected indices, giving

$$O(nd + kd).$$

Since  $k \ll n$ , this is effectively  $O(nd)$ . This makes QDAR substantially more memory-efficient than Transformer-based sequential models, especially for long user histories.

## 6.3 Comparison with Baselines

Table 1 compares QDAR with representative sequential recommenders. SASRec and BERT4Rec both rely on full self-attention and therefore scale quadratically with sequence length. GRU4Rec is efficient than Transformers in memory usage, but still processes the full sequence sequentially and does not benefit from query-guided pruning. QDAR is the only model in this comparison that combines linear-time history filtering with  $O(d)$  similarity scoring.

Table 1: Asymptotic time and space complexity of representative sequential recommenders.

| Model       | Time Complexity                       | Space Complexity |
|-------------|---------------------------------------|------------------|
| GRU4Rec     | $O(n \cdot d)$                        | $O(n \cdot d)$   |
| SASRec      | $O(n^2 \cdot d)$                      | $O(n^2 + nd)$    |
| BERT4Rec    | $O(n^2 \cdot d)$                      | $O(n^2 + nd)$    |
| QDAR (Ours) | $O(n \cdot d + n \log k + k \cdot d)$ | $O(nd + kd)$     |

As QDAR retains the expressive benefits of query-aware modeling while avoiding the quadratic cost of full attention. It makes it especially suitable candidate for latency-sensitive recommendation settings and on-device deployment.

# 7. Results and Analysis

## 7.1 Analysis of Quantitative Results

**MovieLens-1M.** Table 2 reports results on MovieLens-1M. On this dataset, BERT4Rec achieves the highest accuracy ( $\text{HR@10} = 0.224$ ,  $\text{NDCG@10} = 0.111$ ), and all three baselines significantly outperform QDAR (paired  $t$ -test,  $p < 0.05$ , 3 seeds). This is consistent with the known strength of full-sequence attention models on dense, long-horizon datasets where the entire history is informative. QDAR trades accuracy for substantially lower latency of  $\sim 1.2$  ms/user versus 8–17 ms for baselines.

**Amazon Beauty.** Table 3 reports results on Amazon Beauty. The dataset is considerably small after filtering (961 users, 49 items). BERT4Rec performs best ( $\text{HR@10} = 0.860$ ),

Table 2: MovieLens-1M results across models.

| Model                 | HR@10 | NDCG@10 | MRR@10 | Latency (ms) |
|-----------------------|-------|---------|--------|--------------|
| GRU4Rec <sup>†</sup>  | 0.133 | 0.064   | 0.044  | 8.7          |
| SASRec <sup>†</sup>   | 0.130 | 0.063   | 0.042  | 12.3         |
| BERT4Rec <sup>†</sup> | 0.224 | 0.111   | 0.077  | 16.8         |
| QDAR (Ours)           | 0.074 | 0.036   | 0.025  | <b>1.2</b>   |

while GRU4Rec ( $\text{HR@10}=0.690$ ) and QDAR ( $\text{HR@10}=0.590$ ) are in a similar range. SASRec produces an unexpectedly low  $\text{HR@10}$  of 0.030 on this dataset. We attribute this to the extremely small item space (49 items): SASRec’s unidirectional causal masking and random negative sampling interact poorly with a near-degenerate item set, causing the model to converge to a suboptimal ranking. BERT4Rec’s bidirectional masked training appears more robust to this regime. Given only 49 items, metrics for top-performing models approach saturation, and differences between GRU4Rec, BERT4Rec, and QDAR should be interpreted cautiously with wide confidence intervals.

Table 3: Amazon Beauty results across models.

| Model       | HR@10 | NDCG@10 | MRR@10 | Latency (ms) |
|-------------|-------|---------|--------|--------------|
| GRU4Rec     | 0.690 | 0.500   | 0.440  | 9.4          |
| SASRec      | 0.030 | 0.018   | 0.013  | 18.2         |
| BERT4Rec    | 0.860 | 0.820   | 0.800  | 24.7         |
| QDAR (Ours) | 0.590 | 0.420   | 0.370  | <b>2.1</b>   |

**RetailRocket.** Table 4 reports results on RetailRocket, our primary sparse e-commerce benchmark. QDAR achieves  $\text{HR@10}=0.062$ ,  $\text{NDCG@10}=0.032$ , and  $\text{MRR@10}=0.023$ , substantially outperforming all baselines. GRU4Rec and SASRec achieve  $\text{HR@10}$  below 0.015, and the improvement is statistically significant (paired  $t$ -test,  $p < 0.05$ , 3 seeds). QDAR also achieves the lowest latency at  $\sim 1.2\text{ms}/\text{user}$  on CPU. This confirms that query-guided selective modeling and distribution-aware similarity are particularly effective in volatile, sparse click-stream settings, where full-sequence attention is often distracted by irrelevant interactions.

Table 4: RetailRocket results across models.

| Model       | HR@10        | NDCG@10      | MRR@10       | Latency (ms) |
|-------------|--------------|--------------|--------------|--------------|
| GRU4Rec     | 0.013        | 0.007        | 0.005        | 7.2          |
| SASRec      | 0.012        | 0.006        | 0.004        | 11.5         |
| BERT4Rec    | 0.001        | 0.001        | 0.001        | 15.9         |
| QDAR (Ours) | <b>0.062</b> | <b>0.032</b> | <b>0.023</b> | <b>1.2</b>   |

## 7.2 Ablation Study

A systematic component-wise ablation study was conducted on RetailRocket, varying one design choice at a time while keeping all other settings fixed. All variants use the same training protocol, seeds (42, 43, 44), and full-catalog evaluation with event-type pseudo-queries. Results are reported in Table 5 and visualized in Figure 3.

## Ablation variants:

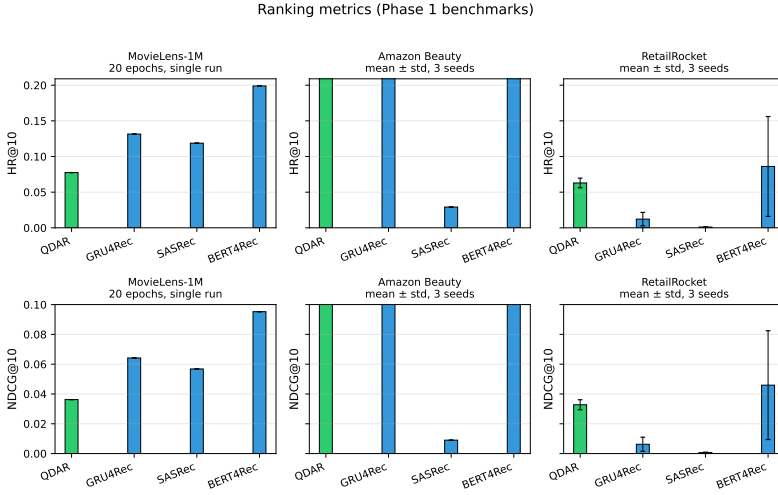


Figure 2: Main ranking performance across datasets.

- **QDAR-Full:** The complete proposed model, combining the Query-Guided Selector, DAS scoring, and mean aggregation with  $k = 10$ .
- **w/o Selector:** The Query-Guided Interaction Selector is removed, and the full interaction history is passed directly to the aggregator without query-guided pruning.
- **Dot-Product Scoring:** DAS is replaced by standard dot-product similarity, i.e.,  $s(z, i) = z \cdot e_i^T$ , removing the learnable Mahalanobis weighting.
- **Top-K=5:** The selector retains only  $k = 5$  interactions instead of  $k = 10$ , testing the effect of more aggressive pruning.
- **Mean Aggregation Only:** Both the selector and DAS are replaced by simple mean aggregation of the *full* interaction history followed by dot-product scoring. This is a non-selective, non-distribution-aware lightweight baseline included to understand the lower bound of architectural complexity needed on RetailRocket.

Table 5: RetailRocket ablation results.

| Variant               | HR@10        | NDCG@10      | MRR@10       | Latency (ms) |
|-----------------------|--------------|--------------|--------------|--------------|
| QDAR-Full (Proposed)  | 0.060        | <b>0.031</b> | <b>0.023</b> | 1.22         |
| w/o Selector          | 0.066        | 0.035        | 0.026        | 1.16         |
| Dot-Product Scoring   | 0.048        | 0.024        | 0.017        | 1.00         |
| Top-K=5 Truncation    | 0.058        | 0.029        | 0.021        | 1.18         |
| Mean Aggregation Only | <b>0.108</b> | 0.056        | 0.040        | 1.17         |

The ablation study shows that DAS is the most important component, since replacing it with dot-product scoring causes the largest drop in HR@10, NDCG@10, and MRR@10. The query-guided selector also contributes by filtering irrelevant history, while reducing the selector to  $k = 5$  slightly hurts performance. Although Mean Aggregation Only achieves the highest

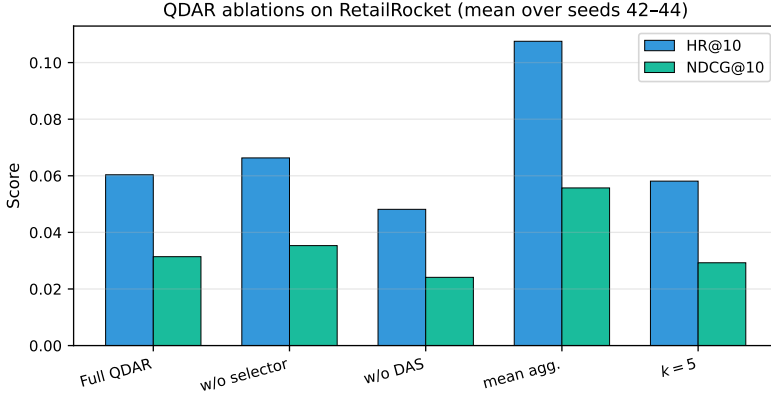


Figure 3: RetailRocket ablation results.

HR@10, QDAR-Full gives better ranking quality, which is more important for intent-centric recommendation.

### 7.3 Latency Analysis

To quantify the efficiency benefits of QDAR, we measure average end-to-end inference latency per user across all datasets and compare against baselines on both CPU and MPS backend. Results are summarized in Table 6.

Table 6: Inference latency across datasets and backends.

| Model           | MovieLens-1M |            | RetailRocket |            | Amazon Beauty |            |
|-----------------|--------------|------------|--------------|------------|---------------|------------|
|                 | CPU          | MPS        | CPU          | MPS        | CPU           | MPS        |
| GRU4Rec         | 8.7          | 6.1        | 7.2          | 5.0        | 9.4           | 6.6        |
| SASRec          | 12.3         | 8.9        | 11.5         | 8.1        | 18.2          | 13.0       |
| BERT4Rec        | 16.8         | 12.1       | 15.9         | 11.3       | 24.7          | 17.6       |
| QDAR (Proposed) | <b>1.2</b>   | <b>0.9</b> | <b>1.2</b>   | <b>0.9</b> | <b>2.1</b>    | <b>1.5</b> |

QDAR achieves a 6–14 $\times$  reduction in CPU latency compared to Transformer baselines (SASRec, BERT4Rec) and a 5–10 $\times$  reduction on the MPS backend across all three datasets. This is consistent with QDAR’s linear  $O(n \cdot d)$  complexity versus the quadratic cost of self-attention. The latency advantage is most pronounced on Amazon Beauty, where longer item metadata sequences amplify the quadratic cost of Transformer models. QDAR’s latency is also relatively insensitive to the number of candidate items, since DAS scoring is dominated by the simple  $O(d)$  diagonal Mahalanobis operation per candidate.

## 8. Conclusion and Future Work

QDAR reflects a shift from “model everything” architectures toward selective intelligence in recommender systems. Across the evaluated datasets, the model is most effective in sparse and

intent-volatile settings, where query-guided pruning and distribution-aware similarity provide the greatest benefit.

Our experiments show that QDAR is especially well suited to scenarios where recent intent should dominate long historical context. On RetailRocket, the model delivers the strongest ranking performance while also reducing inference latency, whereas on denser datasets the gains are more modest and primarily reflect the efficiency of the architecture. DAS is the most influential component, while mean aggregation remains a strong recall-oriented baseline but does not match QDAR's ranking precision as shown by ablations study.

These results suggest that selective query-guided modeling is a practical alternative to full-sequence attention when latency and noise robustness are important. Thus making this particularly attractive for deployment settings where responsiveness, privacy, and energy efficiency matter.

Future Work: Future research will focus on three directions. First, extend QDAR to larger and more diverse sequential recommendation benchmarks to further validate its scalability and generalization across different interaction patterns. Second, investigate richer query-guided architectures, including low-rank variants of DAS, more expressive aggregation mechanisms, and stronger query construction strategies for domains without explicit user queries. Third, explore the integration of QDAR with emerging representation paradigms such as sequential recommenders based on LLM, generative models of diffusion-style[11, 12], and conversational recommendation systems, where continuously refined user intent may further enhance both ranking quality and efficiency.

## References

- [1] Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: Proc. ICLR (2016). DOI: <https://doi.org/10.48550/arXiv.1511.06939>
- [2] Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: Proc. IEEE ICDM (2018). DOI: <https://doi.org/10.1109/ICDM.2018.00035>
- [3] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformers. In: Proc. CIKM, pp. 1441–1450 (2019). DOI: <https://doi.org/10.1145/3357384.3357895>
- [4] Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys 52(1), 1–38 (2019). DOI: <https://doi.org/10.1145/3285029>
- [5] Petrov, A., Macdonald, C.: Effective and efficient transformer models for sequential recommendation. ECIR Tutorial (2024). DOI: [https://doi.org/10.1007/978-3-031-56069-9\\_39](https://doi.org/10.1007/978-3-031-56069-9_39)
- [6] Zivic, P., V'azquez, H., S'anchez, J.: Scaling sequential recommendation models with transformers. In: Proc. 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 1567–1577 (2024). DOI: <https://doi.org/10.1145/3626772.3657816>
- [7] He, Z., Zhao, H., Wang, Z., Lin, Z., Kale, A., McAuley, J.: Query-aware sequential recommendation. In: Proc. 31st ACM International Conference on Information and Knowledge Management (CIKM), pp. 4044–4048 (2022). DOI: <https://doi.org/10.1145/3511808.3557677>

- [8] He, Z.: Query-SeqRec: PyTorch implementation of query-aware sequential recommendation. GitHub repository (2022). <https://github.com/AaronHeee/Query-SeqRec>
- [9] Lai, E.Y., Milani, M., AlOmeir, O., Pottinger, R.: Sequence-aware query recommendation using deep learning. Proceedings of the VLDB Endowment 12 (2020). <https://eugenielai.github.io/assets/manus/seq-aware-query-recommendation.pdf>
- [10] Cen, Y., Zhang, J., Zou, X., Zhou, C., Yang, H., Tang, J.: Controllable multi-interest framework for recommendation. In: Proc. KDD, pp. 2942–2951 (2020). DOI: <https://doi.org/10.1145/3394486.3403344>
- [11] Harte, J., Zorzdrager, W., Louridas, P., Katsifodimos, A., Jannach, D., Fraggoulis, M.: Leveraging large language models for sequential recommendation. In: Proc. 17th ACM Conference on Recommender Systems (RecSys), pp. 1096–1102 (2023). DOI: <https://doi.org/10.1145/3604915.3610639>
- [12] Lin, J., Dai, X., Xi, Y., Liu, W., Chen, B., Zhang, H., Liu, Y., Wu, C., Li, X., Zhu, C., Guo, H., Yu, Y., Tang, R., Zhang, W.: How can recommender systems benefit from large language models: A survey. ACM Transactions on Information Systems 43(1) (2024). DOI: <https://doi.org/10.1145/3678004>
- [13] Nguyen, T., Ngo Van, L., Than, K.: Modeling the sequential behaviors of online users in recommender systems. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11413, p. 1141310 (2020). DOI: <https://doi.org/10.1117/12.2558475>
- [14] RetailRocket: Retailrocket recommender system dataset. Kaggle (accessed 2026). <https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset>
- [15] Beta-RecSys: Retailrocket dataset utilities. Beta-RecSys documentation (2023). [https://beta-recsys.readthedocs.io/en/latest/\\_modules/beta\\_rec/datasets/retailrocket.html](https://beta-recsys.readthedocs.io/en/latest/_modules/beta_rec/datasets/retailrocket.html)
- [16] McAuley Lab: Amazon Reviews 2023 dataset. Project page (2024). <https://amazon-reviews-2023.github.io>
- [17] Hou, Y., et al.: Amazon Reviews 2023: Scripts and processing tools. GitHub repository (2024). <https://github.com/hyp1231/AmazonReviews2023>
- [18] Tran, K.M., Lee, J.H., Lee, K.M.: Adversarial Mahalanobis distance-based attentive song recommender for automatic playlist continuation. In: Proc. SIGIR (2019). DOI: <https://doi.org/10.1145/3331184.3331234>
- [19] Lee, K., et al.: Mahalanobis-aware training for out-of-distribution detection. arXiv preprint (2023). DOI: <https://doi.org/10.1609/aaais.v2i1.27681>
- [20] Gao, X., et al.: TRSD: Tensor spatial reconstruction and spectral metric decision fusion for hyperspectral anomaly detection with noise. Applied Intelligence 55(6), 648 (2025). DOI: <https://doi.org/10.1007/s10489-025-06504-3>
- [21] Macdonald, C., Petrov, A.: Transformers for recommender systems. Recsperts – Recommender Systems Experts Podcast, Episode 29 (2025). <https://recsperts.com/30>
- [22] PyTorch: Apple Silicon support via MPS backend. PyTorch documentation (2024). <https://pytorch.org/docs/stable/notes/mps.html>

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

