Pairwise Ranking Outperforms Single-Action RL for Offline Explanation Selection: A Practical Lesson

Tanay Chowdhury*
Data Scientist
Amazon Web Services
Seattle, USA
tanaycho@amazon.com

Saeideh Shahrokh Esfahani*
Applied Scientist
Amazon Web Services
Mountain View, USA
saeidesh@amazon.com

Abstract

We report a practical lesson from building a GPU-free explainable-recommendation serving stack: explanations are pre-generated offline into a per-item candidate pool, and a small CPU-resident model selects one at request time. Every candidate in the pool of size K carries an offline BERTScore-F1 label against a reference explanation, so we compare a pairwise learning-to-rank model (LightGBM LambdaRank) against a single-action RL formulation (DPO) and a distilled selector on the XRec Google Local benchmark (2,958 pairs, 5 seeds). LambdaRank outperforms both by 0.019–0.025 F1, a gap more than fifteen times the across-seed standard deviation. The advantage appears structural rather than algorithmic: LambdaRank's objective consumes the label of every candidate in the pool, whereas DPO's rollout only observes the label of the sampled side of each preference pair. A separate candidate-source design, selecting from knowledge-graph-grounded paths instead of the cached pool, trades this reference alignment for a Unique-Sentence Ratio (USR) of 1.000. We also encountered two failure modes: further RL fine-tuning on top of an already-distilled policy regresses F1, and an end-to-end RL fine-tune of the generator itself reward-hacks the metric within a few hundred steps. The practical takeaway is that whenever an offline metric can label every candidate in a fixed action set, a pairwise learning-to-rank baseline is worth evaluating before reaching for single-action RL. One caveat: LambdaRank's training signal differs in form from the evaluation metric we report, since it trains on quintile-binned labels while DPO trains on a blended reward, though both derive from the same underlying BERTScore-F1; a fully decorrelated evaluation remains future work.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Reinforcement learning*.

*Both authors contributed equally to this research.

RecSys '26, Minneapolis, MN, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords

explainable recommendation, learning-to-rank, reinforcement learning, offline candidate pools

ACM Reference Format:
Tanay Chowdhury and Saeideh Shahrokh Esfahani. 2026. Pairwise Ranking Outperforms Single-Action RL for Offline Explanation Selection: A Practical Lesson. In *Proceedings of 20th ACM Conference on Recommender Systems – Research and Practice Notes (RecSys '26).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Motivation and Hypothesis

Serving an LLM-generated explanation on every recommendation request is expensive, adding hundreds of milliseconds of latency and a cost that scales linearly with traffic. Prior explainable-recommendation approaches, including transformer-based generation from user/item IDs [5] and RL-based multi-hop reasoning over a knowledge graph [9], both run this generation step at request time. We instead decouple generation from selection: explanations for each (u, i) pair are pre-generated offline into a pool of K candidates, and a small model selects one at request time from a fixed 30-dimensional feature vector per candidate (retrieval similarity, query–candidate relevance, linguistic signals, and prompt/generator provenance). Selection then becomes a one-step decision over a set with a fully labelled reward, since every candidate's BERTScore-F1 [10] against the reference explanation is already known offline. Caching K=40 candidates, rather than one, is not about reuse across requests: it gives enough spread across prompt styles and models for the offline metric to discriminate a strong candidate from a weak one, while keeping generation an offline batch job outside the request-time budget.

This raises a specific, testable hypothesis: when an offline metric can label every candidate in a fixed action set, a pairwise learning-to-rank objective that consumes all K labels should outperform single-action RL formulations that only observe the label of the one action sampled per rollout. This note reports the result of testing that hypothesis, along with two failure modes that surfaced during the same study, on the XRec Google Local benchmark [7] (2,958 review-covered test pairs, byte-verified against G-Refer's [6] published predictions).

2 Methods Compared

Table 1 compares four methods against two published baselines. Three (LambdaRank, DPO, Distillation A) select from the same cached LLM pool (K=40 candidates per pair, six prompt styles over two commodity LLMs) using the same 30-dim feature vector; the fourth (KG-path) builds candidates offline from knowledge-graph

Table 1: BERTScore-F1 and USR on XRec Google Local. Our four rows use the n=2,958 subset; XRec/G-Refer rows are published numbers on their own n=3,000 set, not a strict apples-to-apples comparison. Published baselines and KG-path have no seed variance.

Method	BERTScore-F1 ↑	USR ↑
XRec [7] (published)	0.4311	0.9993
G-Refer 8B [6] (published)	0.4592	**1.000**
KG-path (edge-disjoint)	0.3265 ± 0.074	**1.000**
DPO	0.4749 ± 0.001	0.909
Distillation A only	0.4817 ± 0.000	0.865
LambdaRank	**0.5003**	0.808

paths instead. LambdaRank is a LightGBM pairwise ranker [1] trained on quintile-binned per-candidate F1 labels; at inference we take the argmax within each test group. DPO samples preference pairs from the pool ($F1(a^+) > F1(a^-) + \delta$) and trains against a frozen reference policy [8]; it was the best-performing single-action RL formulation we tried. Distillation A is an MLP student trained via knowledge distillation [4] to match LambdaRank's soft per-candidate distribution. The KG-path variant (edge-disjoint enumeration) builds candidates from a different source: up to five edge-disjoint (u, i) paths from a heterogeneous knowledge graph via Menger-style max-flow decomposition, each rendered to text, with a PPO policy selecting one of ten action slots. Because every selected explanation is grounded on a path structurally unique to its query, this variant reaches USR = 1.000 by construction, at a substantial cost to reference-aligned F1. We omit full hyperparameters here for space and will release code and configs at camera-ready.

3 Results

Table 1 reports BERTScore-F1 and USR (Unique-Sentence Ratio, the fraction of selected outputs that are token-unique across the test set). LambdaRank reaches F1 = 0.500, ahead of Distillation-A (0.482) by 0.019 and ahead of DPO, the strongest single-action RL variant we tested, by 0.025. Both DPO and Distillation-A have an across-seed standard deviation of at most 0.001 (5 seeds), so the 0.019–0.025 gap to LambdaRank is more than fifteen times either method's standard deviation.

We attribute the F1 gap to how much of the pool each objective uses: LambdaRank's pairwise ΔNDCG objective sees the F1 label of all 40 candidates per query, while DPO observes only the sampled side of each preference pair, leaving most labels unused by the gradient. Distillation closes most of this gap by transferring LambdaRank's full ranking into a compact student; the residual 0.019 gap looks like a softmax-compression artefact, since the temperature-softened teacher distribution is a lower-fidelity target than the LightGBM argmax itself.

USR runs the opposite direction within the offline-pool family: LambdaRank has the highest F1 but lowest USR (0.808), while DPO trades F1 for higher USR (0.909)—the sharpest decision boundaries collapse onto a narrower slice of the pool, trading diversity for reference alignment. The KG-path variant sits at the extreme of this trade-off: because every explanation is grounded on a path structurally unique to its (u, i) query, it reaches USR = 1.000 by construction, but its candidates are not optimised against the reference text, and F1 falls to 0.327. Which end of this trade-off matters depends on the deployment: reference-aligned F1 is the right target when explanations are compared against a ground truth, while USR matters more when avoiding visibly repeated explanations is itself a product requirement. A lighter-weight, untested mitigation is a post-hoc MMR-style diversity penalty [2] on LambdaRank's scores, discounting candidates similar to explanations already selected for related items, rather than the KG-path variant's all-or-nothing trade.

Failure mode 1. We further fine-tuned the Distillation-A student with GRPO, expecting a small gain. Instead, F1 regressed (0.4767 vs. 0.4817, $>10\sigma$ across 5 seeds), likely because the entropy bonus pushes a near-converged policy back toward exploration, away from the teacher-induced argmax; removing the entropy term recovered about half the loss. RL fine-tuning after distillation seems to help only when the distilled policy is still far from optimal, which was not the case here.

Failure mode 2. In a separate experiment we fine-tuned a 3B-parameter generator end-to-end with BERTScore-F1 as the RL reward. Within a few hundred steps it reward-hacked the metric, converging on repeating "user enjoys" since that phrase is frequent in the reference corpus. Decoupling generation from selection avoids this by construction: the generator is never in the training loop.

4 Training Signal vs. Evaluation Metric

Every number in Table 1 is the same continuous BERTScore-F1, computed identically across methods at evaluation time. What differs is the form of the training signal each method actually optimises. LambdaRank never sees continuous F1 during training: labels are quintile-binned (5 ordinal bins) before fitting the pairwise ΔNDCG objective. DPO instead trains on a blended, rescaled reward $\alpha R_{\text{struct}} + (1-\alpha) R_{\text{sem}}$, where R_{sem} is F1 rescaled to $[0, 100]$ and mixed with a structural term. Since both training signals derive from the same underlying BERTScore-F1, this is not an independent-metric comparison, and we cannot rule out that some of LambdaRank's advantage comes from optimising a coarser, more robust target rather than from consuming every candidate's label. A cleaner ablation would train LambdaRank on continuous F1 and the RL variants on unblended, unbinned F1 to isolate the two effects; we see this, along with a decorrelated automatic metric or a human preference study, as the natural next step. Separately, BERTScore is an imperfect quality proxy, missing factual errors and diverging from human judgments in some settings [3]; our F1 gap is a claim about label utilization, not human-perceived quality.

5 Practical Takeaway

When a selection problem admits a reward that can be computed offline for every candidate in a fixed action set, which is common whenever candidates are pre-generated and scored against a reference metric, a pairwise learning-to-rank baseline is worth evaluating before adopting single-action RL formulations such as DPO: single-action formulations leave most supervision unused, unlike a dense pairwise objective.

Supplementary: Prompt Styles and Reproducibility Artifacts

The offline pool is built from six prompt styles applied to two commodity LLMs (Amazon Nova Lite and Claude Haiku): (A) retrieve-grounded paraphrase, (B) retrieve-grounded synthesis, (C) 2-shot retrieval from nearest training neighbours, (D) review-grounded chain-of-thought, (E) adversarial refinement, and (F) length-tuned (25–33 word) synthesis. Styles A and B ground on same-business review text, and C–F are generated at training-set featurisation time.

- **Code and configs:** the code for this work, including prompt templates, pool-generation code, and trained selectors, is in the process of being open-sourced.
- **Pool rebuild cost:** generation, featurisation, 5-seed training, and 3-metric scoring for the Google Local pool cost ~$15 in LLM API spend and ~14 CPU-hours. Analytical estimate from measured offline compute and public cloud pricing, not a production measurement.
- **Serving latency:** the LambdaRank selector (~1.7 MB) returns in <1 ms per query on a single CPU core; end-to-end latency stays <100 ms at p99, with no GPU on the request path. A live A/B or shadow-traffic test is a natural next step.

References

[1] Christopher J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview.* Technical Report MSR-TR-2010-82. Microsoft Research.

[2] Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *SIGIR*. 335–336.

[3] Michael Hanna and Ondřej Bojar. 2021. A Fine-Grained Analysis of BERTScore. In *WMT*. 507–517.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).

[5] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *ACL*. 4947–4957.

[6] Yuhan Li, Xinni Zhang, Linhao Luo, Heng Chang, Yuxiang Ren, Irwin King, and Jia Li. 2025. G-Refer: Graph Retrieval-Augmented Large Language Model for Explainable Recommendation. In *WWW*. 240–251. doi:10.1145/3696410.3714727

[7] Qiyao Ma, Xubin Ren, and Chao Huang. 2024. XRec: Large Language Models for Explainable Recommendation. In *Findings of EMNLP*. 391–402. doi:10.18653/v1/2024.findings-emnlp.22

[8] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *NeurIPS*.

[9] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR*. 285–294.

[10] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *ICLR*.