# GraphReduce: Coverage-Preserving LLM Aggregation for E-commerce Review Insights

**Anonymous ACL submission**

## Abstract

In e-commerce, product detail pages (PDPs) often contain hundreds or thousands of customer reviews, yet the interface can surface only a small set of concise and reliable insights. A useful review aggregation system must therefore abstract recurring themes while preserving which reviews support each insight. We study this problem as structured review-insight mining: review-level tuples are first extracted and then aggregated into ranked product-level insight clusters with explicit member traceability. We propose GRAPHREDUCE, a coverage-preserving framework for product-scale review insight aggregation. It combines a contrastively trained projection layer over frozen sentence embeddings, graph-based clustering on local neighborhood structure, and an LLM reducer/ranker for naming and ordering canonical clusters. By keeping membership accounting outside the generative step, GRAPHREDUCE preserves complete tuple coverage and avoids a common failure mode of direct LLM summarization: producing plausible high-level themes while silently dropping long-tail evidence. Across three product-category benchmarks, GRAPHREDUCE improves recovery of gold insight clusters over strong lossless embedding-based baselines while maintaining comparable ranking quality. We further show that flat LLM and topic-modeling baselines can appear competitive under ranking metrics despite omitting substantial input evidence, and that an anti-copy reinforcement objective improves the quality of upstream tuple extraction. These results highlight the importance of coverage-aware aggregation for faithful review insight mining in e-commerce settings.

## 1 Introduction

Product review summarization has increasingly moved beyond short free-form summaries toward more structured forms of opinion aggregation, including aspect-guided summaries (Boytsov et al., 2025), aspect-controllable opinion summaries (Amplayo et al., 2021), learned review selection, synthetic-data pipelines for e-commerce opinions (Siledar et al., 2023), and facet-based faithful summaries (Wang et al., 2025). These methods address a central question: *what should a summary say?* In e-commerce product detail pages (PDPs), however, a deployed review insight system must also answer a second question: *which reviews support each displayed insight?* Without explicit member traceability, a system may generate fluent and useful-looking themes while silently ignoring long-tail evidence needed for user trust, analytics, and auditability.

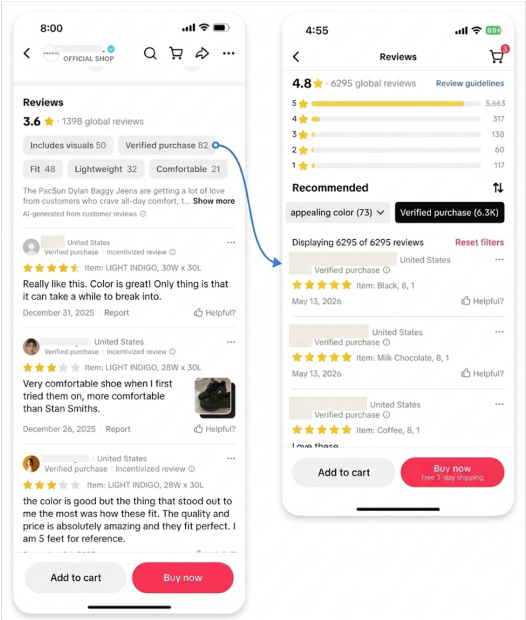


Figure 1: Demo of review insight

We study this setting as *structured review-insight mining*, a two-stage problem. In the first stage, each review is converted into one or more structured tuples, each consisting of an abstracted insight, an aspect, a sentiment label, a language tag, and the supporting review span. In the second stage, tu-

ples are grouped by product and aggregated into a ranked list of canonical product-level insight clusters. Each output cluster has a human-readable description and a complete set of member tuple ids, linking the displayed insight back to the evidence that supports it. This explicit coverage requirement changes the nature of the task. A method that only emits salient head themes can score well under ranking-oriented metrics such as NDCG, while discarding the long-tail evidence needed for downstream analysis, verification, and auditability.

We propose GRAPHREDUCE, a coverage-preserving aggregation framework for product-scale review insights. GRAPHREDUCE first maps tuple representations into a product-cluster-aware space using a lightweight contrastive projection over frozen sentence embeddings. It then constructs local-neighborhood graphs, applies spectral clustering, and uses an LLM reducer/ranker to name and order the resulting canonical clusters. Crucially, semantic generation is separated from membership accounting: the LLM produces cluster labels and rankings, while tuple membership is propagated deterministically through the aggregation pipeline. As a result, every input tuple remains assigned to exactly one output cluster.

This formulation differs from subset-based opinion summarization (Jiang et al., 2023; Bražinskas et al., 2021), which first selects a small set of representative reviews and then summarizes them. Our goal is not to summarize a chosen subset, but to aggregate all extracted evidence into a compact set of traceable product insights. This distinction is especially important for long-context LLM aggregation: direct prompting can often produce plausible high-level themes, but as the number of tuples grows, it may omit substantial portions of the input without an explicit failure signal.

Our contributions are threefold:

1. **Task and benchmark.** We formulate product review insight mining as ranked clustering with explicit member-id coverage, and evaluate systems using ranking quality, cluster recovery, coverage, and duplicate assignment metrics across three e-commerce product categories.

2. **Method.** We introduce GRAPHREDUCE, a projection-plus-graph-clustering-plus-LLM aggregation pipeline that preserves member coverage by construction while producing canonical ranked insight clusters.

3. **Findings.** Across three category benchmarks, GRAPHREDUCE recovers gold insight clusters more effectively than strong lossless embedding-based baselines while maintaining comparable ranking quality. We also show that flat LLM and topic-modeling baselines can appear competitive under ranking metrics despite dropping input evidence, and that anti-copy reinforcement learning improves the upstream tuple extractor.

## 2 Related Work

**Opinion and product review summarization.** Opinion summarization has moved from free-form summaries toward structured and controllable representations of user opinions, including aspect-controllable summarization (Amplayo et al., 2021), review and subset selection (Jiang et al., 2023), synthetic e-commerce data (Siledar et al., 2023), LLM-based facet induction (Wang et al., 2025), extractive aspect-based summarization (Gunel et al., 2023), latent-facet discovery (Lakkaraju et al., 2011), and industrial aspect-guided review summarization (Boytsov et al., 2025; Soltan et al., 2022). These methods mainly address what a summary should contain and how it should be expressed. Our setting adds evidence accounting: each displayed insight must remain linked to its supporting review-level evidence. We therefore aggregate extracted evidence into ranked canonical clusters with explicit member assignments, rather than producing only a free-form summary or a selected review subset.

**Aspect-based sentiment analysis and tuple extraction.** Aspect-based sentiment analysis identifies opinion targets and sentiment in reviews (Pavlopoulos and Androutsopoulos, 2014; Tang et al., 2016; Chen et al., 2017; Xue and Li, 2018). Our upstream extractor extends this setup with tuples containing an abstracted insight phrase, aspect, sentiment label, language tag, and verbatim evidence span. The span grounds the tuple, while the insight should abstract the sentiment-bearing content. This creates a copying failure mode, where a model satisfies grounding by reusing the evidence span as the insight. We use DPO (Rafailov et al., 2023) and GRPO (Shao et al., 2024) to improve abstraction, but the main focus of this paper is aggregation over fixed extracted tuples.

**Long-context LLMs and evidence coverage.** Long-context LLMs allow many review tuples to be placed in one prompt, but context length does not ensure faithful or complete source use (Laban et al., 2024; Liu et al., 2024; Bai et al., 2024; Kim et al., 2024). In review insight aggregation, a model may generate plausible high-level themes and strong rankings while omitting less frequent evidence. We therefore separate ranking quality from evidence accounting, reporting NDCG and cluster recovery alongside tuple coverage and duplicate assignments.

**Contrastive embeddings and graph-based clustering.** Sentence encoders such as SimCSE (Gao et al., 2021) and BGE (Xiao et al., 2024) provide strong semantic spaces, while contrastive learning adapts embeddings to task-specific similarity (Wang et al., 2022). Because generic similarity may not match product-level insight boundaries, we learn a lightweight projection over a frozen encoder and apply spectral clustering over local-neighborhood graphs. Unlike recent LLM-guided hierarchical or graph-based clustering work (Pattnaik et al., 2024; Ashkenazi et al., 2025), our setting requires canonical descriptions and rankings while assigning every tuple id exactly once.

## 3 Method: GRAPHREDUCE

Figure 2 summarizes the GRAPHREDUCE aggregation pipeline. The key design principle is to separate semantic abstraction from exact membership accounting: language models describe and rank clusters, while tuple assignments are maintained by graph clustering and deterministic set union.

### 3.1 The GRAPHREDUCE inference pipeline

Given the $N$ extracted tuples for a product and a target of $K$ output insights, GRAPHREDUCE proceeds hierarchically. Each tuple is encoded with a frozen sentence encoder and mapped through a lightweight projection head into a product-cluster-aware representation space. At each aggregation level, GRAPHREDUCE constructs a $k$-nearest-neighbor graph over the current pool and applies spectral clustering to obtain a hard partition. The number of clusters is chosen so that each reducer call receives a bounded-size group, while the pool is gradually reduced toward the target output size.

For each cluster, an LLM reducer generates a canonical phrase describing the shared insight. The reducer output replaces the original items in the next-level pool, together with the cluster centroid and the union of all member ids contained in the cluster. This process repeats until the pool contains at most $K$ items. The same LLM is then prompted to rank the final candidates and assign importance scores.

**Algorithm 1** GRAPHREDUCE inference. Member-id sets are propagated by deterministic union outside the LLM.

**Input** tuples $T = \{t_i\}_{i=1}^{N}$; target size $K$; group size $B$; projection $\phi$; reducer/ranker $\pi_\theta$; embedder Emb
**Output** ranked clusters with complete member-id assignments

$x_i \leftarrow \mathrm{Emb}(t_i)$; $z_i \leftarrow \phi(x_i)$ for all $t_i$
$pool \leftarrow \{(t_i, z_i, \{\texttt{row\_id}_i\})\}_{i=1}^{N}$
**while** $|pool| > K$ **do**
  $K' \leftarrow \max(K, \lceil |pool|/B \rceil)$
  $\{C_j\}_{j=1}^{K'} \leftarrow \textsc{SpectralKNN}(\{z\}, K')$
  $p_j \leftarrow \pi_\theta(C_j \rightarrow \textsc{Name})$ for each $C_j$
  $R_j \leftarrow \bigcup_{u \in C_j} R_u$; $\bar{z}_j \leftarrow |C_j|^{-1} \sum_{u \in C_j} z_u$
  $pool \leftarrow \{(p_j, \bar{z}_j, R_j)\}_{j=1}^{K'}$
**end while**
$(top_K, scores) \leftarrow \pi_\theta(pool \rightarrow \textsc{Rank}, K)$
**return** $(top_K, scores, \{R_j\})$

Because each intermediate clustering step returns a partition, and because parent member sets are computed by deterministic union, every input tuple remains assigned to exactly one final cluster. The coverage guarantee therefore applies to membership accounting over the extracted tuple inventory; it does not depend on the LLM reproducing row ids or maintaining global state in text.

A direct LLM aggregation prompt must jointly decide which insights are salient, generate their descriptions, and return the exact set of member ids for each cluster. GRAPHREDUCE decomposes these roles. The LLM is used where linguistic abstraction is needed, while exact membership is computed outside the model as a graph operation. This makes cluster assignments auditable before language generation and isolates the effect of the learned projection on the geometry used by the graph partitioner.

The number of LLM calls grows with the number of bounded reducer groups, rather than relying on a single long-context prompt:

$$1 + \sum_{\ell=0}^{L-1} \max\left(K, \left\lceil \frac{N}{B^{\ell+1}} \right\rceil\right),$$

where the final 1 corresponds to the ranking prompt, $B$ is the maximum reducer group size, and $L$ is the number of aggregation levels.

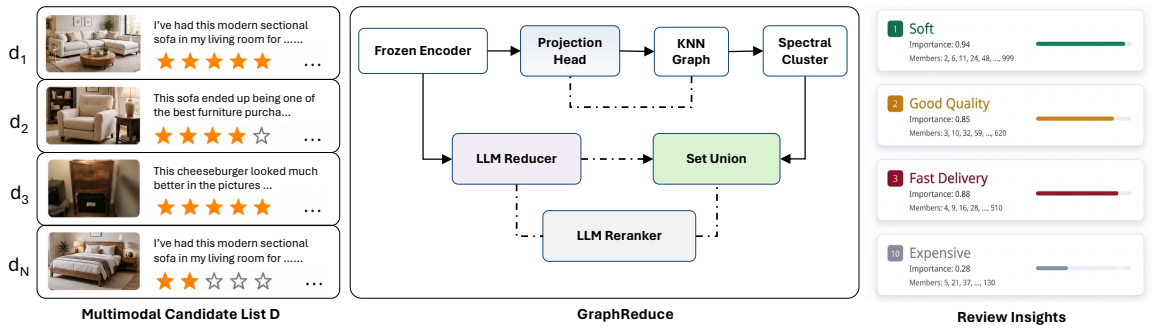


Figure 2: Review tuples are projected into a cluster-aware embedding space and partitioned by local graph clustering. LLMs name and rank the final insight clusters, while member ids are propagated outside the model by deterministic set union to preserve full evidence coverage.

### 3.2 Training the projection head

Generic sentence embeddings provide a useful starting point for tuple similarity, but their neighborhood structure is not always aligned with product-level insight clusters. Tuples that mention the same product component may be close in the embedding space even when they describe different issues, while lexically different phrases may express the same underlying concern. We therefore learn a lightweight projection head over a frozen sentence encoder to adapt tuple representations to the cluster boundaries observed within products.

The projection head PROJHEAD is trained with an InfoNCE objective (Oord et al., 2018). For each training product, anchors and positives are sampled from the same gold insight cluster. Hard negatives are sampled from other clusters of the same product, and easy negatives from other products. Product-local hard negatives force the projection to distinguish semantically adjacent but operationally different issues in the same product context. The base encoder remains frozen, so the learned component only reshapes the embedding space used by the downstream graph partitioner. In our implementation, we use BGE-small-en-v1.5 and a 384→256→128 MLP projection head.

The product-level gold clusters used for projection training are constructed from embedding-based candidate clusters, teacher-LLM validation and renaming, and LLM-based importance ranking (Appendix C). Test products are excluded from both projection-head training and reducer fine-tuning.

### 3.3 K-way reducer supervision

The reducer converts a group of tuples or intermediate clusters into canonical insight descriptions. A reducer trained only on single-cluster cleanup examples can name coherent groups, but it does not learn what to do when a reducer call receives a mixed group. This mismatch is problematic in hierarchical aggregation, where early groups may contain several related themes and require controlled splitting rather than a single over-general label.

We therefore construct $K$-way reducer supervision from training products. For each example, we sample several gold clusters from the same product, draw a bounded number of tuples from their union, shuffle the inputs, and train the reducer to recover the corresponding partition and canonical cluster names. This matches the reducer's inference-time role: a call may receive either a coherent cluster that should be summarized as one insight, or a mixed group that should be separated into multiple canonical insights. In our experiments, each synthetic reducer example samples $k_{\text{in}} \in \{2, 3, 5, 7, 10\}$ gold clusters and at most 200 tuples.

The same LLM checkpoint is used for local naming and final ranking. At the final stage, each candidate already carries a deterministic member set, so the ranker assigns an ordering and importance scores without modifying membership assignments.

## 4 Experiments

We evaluate GRAPHREDUCE on three held-out product-category benchmarks built from Amazon product reviews (Hou et al., 2024): Software, Fashion, and Baby. Each benchmark contains 100 products with product-level gold insight clusters and importance rankings. All test products are excluded from both projection-head training and reducer fine-tuning.

### 4.1 Baselines and metrics

We compare GRAPHREDUCE against four baselines that represent different trade-offs between semantic abstraction and evidence accounting.

- **flat_llm**: a single LLM call over all tuples for a product, prompted to produce ranked insight clusters and member ids directly.
- **BERTopic** (Grootendorst, 2022): a topic-modeling pipeline using UMAP, HDBSCAN, and c-TF-IDF naming over the same frozen BGE embeddings. HDBSCAN outliers are treated as unassigned.
- **hier_agglom (BGE raw)**: a lossless hierarchical aggregation pipeline with the same LLM naming and ranking steps as GRAPHREDUCE, but using raw frozen BGE embeddings without the learned projection head.
- **embed_cluster**: deterministic embedding clustering with a modal raw phrase used as the cluster name, without LLM-based naming.

We report four complementary metrics. **Recall@10** measures the fraction of gold top-10 insight clusters recovered by the predicted top-10 list. **Precision@10** measures the fraction of predicted top-10 clusters that match a gold cluster. **NDCG@10** evaluates the ordering of matched predicted clusters using the gold importance ranking as relevance. Predicted and gold cluster names are matched one-to-one using a phrase-token Jaccard rule; duplicate aliases do not receive repeated credit.

Because ranking metrics do not measure whether the input evidence has been fully accounted for, we also report **coverage** and **duplicate assignments**. Coverage is the fraction of input tuple ids that appear in at least one predicted top-10 cluster, and duplicate assignments count tuple ids that appear in more than one predicted cluster. These metrics separate two questions that are often conflated: whether a system names the most important insights, and whether it preserves the structured evidence inventory supporting those insights.

### 4.2 Main results

Table 1 shows the main Stage-2 aggregation results. Among methods with complete member coverage, GRAPHREDUCE obtains the highest recall@10 in all three categories. Relative to the strongest lossless baseline, hier_agglom with raw BGE embeddings, GRAPHREDUCE improves recall@10 from 0.575 to 0.609 on Software, from 0.502 to 0.536 on Fashion, and from 0.618 to 0.674 on Baby. The gains are statistically significant under paired Wilcoxon tests over products. At the same time, GRAPHREDUCE preserves 100% coverage and produces no duplicate member assignments.

The NDCG results show a different pattern: GRAPHREDUCE is comparable to the raw-BGE lossless baseline, with differences within bootstrap uncertainty. We therefore interpret ranking quality as parity rather than a consistent NDCG improvement. The main gain is improved gold cluster recovery under exact evidence coverage, not substantially better top-insight ordering.

The non-lossless baselines show why coverage must be reported separately. Flat LLM prompting and BERTopic can be competitive on NDCG, and in some categories achieve the highest NDCG or precision, but they do so with incomplete evidence accounting. Flat prompting omits tuple ids and produces duplicate assignments, while BERTopic discards HDBSCAN outliers. Across the 100-product benchmarks, omitted inventory ranges from 4.9% to 39.9%. Thus, ranking metrics alone can overstate methods that capture salient head themes while leaving input evidence unassigned.

Overall, GRAPHREDUCE is the only method that improves cluster recovery over the strongest lossless baseline while retaining complete coverage and zero duplicates.

### 4.3 Large-$N$ stress test

We further evaluate five Fashion products with $N \in [800, 1700]$ tuples (Appendix G). This stress test is not intended to support broad statistical claims, but to examine how aggregation behaves when the tuple inventory exceeds the size of the main benchmark products. In this setting, flat prompting keeps only 22.8% of tuple ids, despite producing plausible insight names. By contrast, GRAPHREDUCE maintains 100% coverage and zero duplicate assignments. The result highlights the central limitation of treating review insight aggregation as a single long-context generation problem: the output may look coherent while failing to preserve the input inventory.

### 4.4 Ablations

We ablate the main aggregation components on a 30-product Software development benchmark (Ta-

| Category | Strategy | recall@10 | prec@10 | NDCG@10 | **coverage** | **dup** | calls |
|---|---|---|---|---|---|---|---|
| Software ($N \leq 297$) | flat_llm | 0.532 | 0.573 | 0.619 | 0.942 (–5.8%) | 107 | 1.0 |
| | BERTopic | 0.471 | 0.723 | 0.599 | 0.883 (–11.7%) | 0 | 0.0 |
| | hier_agglom (BGE raw) | 0.575 | 0.520 | 0.648 | **1.000** | 0 | 10.0 |
| | **GRAPHREDUCE** | **0.609**$^{*}$ | 0.551 | **0.667** | **1.000** | **0** | 11.0 |
| Fashion ($N \leq 486$) | flat_llm | 0.516 | 0.517 | 0.602 | 0.872 (–12.8%) | 88 | 1.0 |
| | BERTopic | 0.562 | 0.562 | **0.631** | 0.601 (–39.9%) | 0 | 0.0 |
| | hier_agglom (BGE raw) | 0.502 | 0.502 | 0.561 | **1.000** | 0 | 10.0 |
| | **GRAPHREDUCE** | 0.536$^{**}$ | 0.536 | 0.581 | **1.000** | **0** | 11.0 |
| Baby ($N \leq 500$) | flat_llm | 0.618 | 0.539 | **0.676** | 0.951 (–4.9%) | 51 | 1.0 |
| | BERTopic | 0.430 | **0.792** | 0.575 | 0.879 (–12.1%) | 0 | 0.0 |
| | hier_agglom (BGE raw) | 0.618 | 0.465 | 0.652 | **1.000** | 0 | 10.0 |
| | **GRAPHREDUCE** | **0.674**$^{**}$ | 0.508 | 0.669 | **1.000** | **0** | 10.6 |

Table 1: Stage-2 results on fixed 100-product held-out benchmarks. Coverage is the fraction of input tuple ids assigned to predicted top-10 clusters; **dup** counts tuple ids assigned to multiple clusters. $^{*}/^{**}$ denote GRAPHREDUCE recall gains over hier_agglom significant at $p < 0.05$ / $p < 0.01$ by paired Wilcoxon tests ($n = 100$). NDCG differences are within bootstrap uncertainty (Appendix H).

ble 7).

First, adapting the embedding geometry helps. Replacing raw BGE embeddings with the contrastively trained projection head improves cluster recovery and ranking quality under the same downstream pipeline, indicating that generic semantic similarity is insufficient for product-level insight boundaries.

Second, the hierarchical chunk-plus-rank pipeline matters. Using the projection with a simpler clustering backend degrades performance, showing that the gain is not from the projection alone. Bounded reducer calls and the final ranking step are both needed to convert local cluster structure into a stable top-$K$ output.

Third, more expressive learned clustering is not automatically better. A GraphPool-style model with GraphSAGE (Hamilton et al., 2017) and Gumbel-Softmax assignments underperforms the explicit spectral partition, even as its training reward improves. This points to a reward-alignment issue: local differentiable clustering objectives do not necessarily optimize top-$K$ ranking and recovery metrics. We therefore limit the learned component to representation projection and retain explicit graph partitioning for cluster assignment.

### 4.5 Qualitative analysis

Appendix J presents a worked example and a manual analysis of common errors. We observe four recurring patterns. First, the reducer sometimes produces phrase aliases that are semantically close to the gold label but fail the automatic phrase match, lowering NDCG even when the member partition is reasonable. Second, long-tail clusters can compete with broader head themes for limited top-$K$ slots. Third, spectral clustering can over-merge related issues when the product has few tuples. Finally, some clusters have mostly correct member sets but receive overly generic or partially incorrect names. These errors suggest that future work should improve semantic matching in evaluation, calibrate the head–tail trade-off, and make cluster naming more robust.

## 5 Conclusion

We introduced GRAPHREDUCE, a coverage-preserving framework for aggregating e-commerce review tuples into ranked, traceable product-level insight clusters. GRAPHREDUCE uses LLMs for naming and ranking, while graph clustering and deterministic set propagation preserve member assignments. Across three product-category benchmarks, it improves cluster recovery over strong lossless baselines while maintaining comparable ranking quality and complete coverage. Our results show that ranking metrics alone can obscure evidence loss: flat LLM and topic-modeling baselines may look competitive while leaving input tuples unassigned. Faithful review insight mining should therefore evaluate not only the displayed insights, but also the coverage and consistency of their supporting evidence.

## Limitations

First, the product-level gold is LLM-assisted: Gemini participates in importance scoring and evaluation. We use held-out product splits to avoid train/test leakage, but human audits with inter-

annotator agreement are needed before treating the benchmark as a human gold standard. Second, some evaluation axes remain limited. The large-$N$ Fashion stress test covers only five products, Stage-1 extraction is fully evaluated only on Software, and the phrase-token matching metric may miss semantic equivalence or over-credit generic aliases. Broader human validation and semantic matching would make the evaluation more robust.

## References


Reinald Kim Amplayo, Stefanos Angelidis, and Mirella Lapata. 2021. Aspect-controllable opinion summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6578–6593.

Yoel Ashkenazi, Dekel Cohen, Etzion Harari, Naphtali Abudarham, Regev Yehezkel Imra, and Yoram Louzoun. 2025. D2cs-documents graph clustering using llm supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23606–23623.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Ilya Boytsov, Vinny DeGenova, Mikhail Balyasin, Joseph Walt, Caitlin Eusden, Marie-Claire Rochat, and Margaret Pierson. 2025. End-to-end aspect-guided review summarization at scale. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 463–471.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2021. Learning opinion summarizers by selecting informative reviews. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9424–9442.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6894–6910.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

Beliz Gunel, Sandeep Tata, and Marc Najork. 2023. Strum: Extractive aspect-based contrastive summarization. In *Companion proceedings of the ACM web conference 2023*, pages 28–31.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*.

Han Jiang, Rui Wang, Zhihua Wei, Yu Li, and Xinpeng Wang. 2023. Large-scale and multi-perspective opinion summarization with diverse review subsets. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5641–5656.

Yekyung Kim, Yapei Chang, Marzena Karpinska, Aparna Garimella, Varun Manjunatha, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. Fables: Evaluating faithfulness and content selection in book-length summarization. *arXiv preprint arXiv:2404.01261*.

Philippe Laban, Alexander Richard Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context llms and rag systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9885–9903.

Himabindu Lakkaraju, Chiranjib Bhattacharyya, Indrajit Bhattacharya, and Srujana Merugu. 2011. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 498–509. SIAM.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Anup Pattnaik, Cijo George, Rishabh Kumar Tripathi, Sasanka Vutla, and Jithendra Vepa. 2024. Improving hierarchical text clustering with llm-guided multi-view cluster representation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 719–727.

John Pavlopoulos and Ion Androutsopoulos. 2014. Aspect term extraction for sentiment analysis: New datasets, new evaluation measures and an improved unsupervised method. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 44–52.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Tejpalsingh Siledar, Suman Banerjee, Amey Patil, Sudhanshu Singh, Muthusamy Chelliah, Nikesh Garera, and Pushpak Bhattacharyya. 2023. Synthesize, if you do not have: Effective synthetic dataset creation strategies for self-supervised opinion summarization in e-commerce. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13480–13491.

Saleh Soltan, Victor Soto, Ke M Tran, and Wael Hamza. 2022. A hybrid approach to cross-lingual product review summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 18–28.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers*, pages 3298–3307.

Jian Wang, Yanjie Liang, Yuqing Sun, and Bin Gong. 2025. Inducing argument facets for faithful opinion summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 16153–16166.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

# A Data Provenance, Models, and Privacy

**Source data.** All experiments use publicly released Amazon product reviews (Hou et al., 2024), restricted to three categories: Software, Fashion, and Baby Products. We use the dataset as distributed and do not collect any additional user data. The annotation pool contains 4,828 Software, 1,112 Fashion, and 8,090 Baby products; the main benchmarks use fixed 100-product held-out slices per category (Appendix C), and Stage-1 extraction is evaluated on 5,000 held-out Software reviews (Appendix B).

**Privacy and anonymization.** The source corpus is a public benchmark that is already de-identified at release: reviews are keyed by opaque, anonymized identifiers and contain no names, contact details, or account information beyond the public review text. We therefore introduce no additional personal data and perform no re-identification. We neither attempt to recover author identities nor link reviews to external profiles, and we release only aggregate, product-level insight clusters rather than user-level records. No new human data was collected for this work.

**Models used and their roles.** Our pipeline uses several models, each confined to a specific role (Table 2). Generic semantic representations come from a frozen BGE-small-en-v1.5 encoder (Xiao et al., 2024). Stage-1 tuple labels are produced by a frontier teacher LLM, and extractor students are Qwen3.5-9B variants (Yang et al., 2025) trained with SFT, DPO (Rafailov et al., 2023), and GRPO (Shao et al., 2024). Gemini 2.5 Pro serves as the Stage-1 quality judge and, in gold construction, as the Layer-3 importance ranker and member auditor (Appendix C). The same fine-tuned Qwen reducer is used for both cluster naming and final ranking at inference time. Crucially, membership accounting is deterministic and uses no LLM. As noted in the Limitations, Gemini participates in both gold construction and evaluation, which we treat as a limitation rather than an anti-circularity guarantee.

# B Stage-1 Extraction Details

**Setup.** We sample product-keyed reviews from Amazon Software, Fashion, and Baby Products (Hou et al., 2024) and label per-review tuples with a frontier teacher LLM. We train Qwen3.5-9B (Yang et al., 2025) variants with full-parameter

| Component | Model | Role |
|---|---|---|
| Sentence encoder | BGE-small-en-v1.5 | frozen embeddings |
| Stage-1 teacher | frontier LLM | tuple annotation |
| Stage-1 student | Qwen3.5-9B | tuple extraction |
| Stage-1 judge | Gemini 2.5 Pro | quality scoring |
| Gold Layer-2 | teacher LLM | validate / rename |
| Gold Layer-3 | Gemini 2.5 Pro | rank / audit |
| Reducer / ranker | Qwen3.5-9B (SFT) | name / rank clusters |

Table 2: Models used in the pipeline and their roles. Membership accounting is deterministic and uses no LLM.

SFT, DPO, and GRPO. The held-out Stage-1 evaluation set contains 5,000 Software reviews. Gemini 2.5 Pro judges five dimensions from 1 to 5: correctness, completeness, diversity, abstraction, and overall quality. We report the production quality rate overall $\geq 4$.

**Reward design.** The dominant SFT failure is *slice copying*: the model places the verbatim evidence span in both *slice* and *insight*. The insight is then validly grounded but not abstracted. The per-tuple insight score starts at $+1$ and is clipped below at $-2$ after four penalties: $-1.5$ if the generated insight exactly equals the generated slice, $-0.5$ if the insight is a non-slice verbatim substring of the source review, $-0.5$ if the insight is not 1–3 words, and $-0.7$ if it is a vague single-token phrase such as "good". The second penalty targets non-slice verbatim copying from the review; it is not a penalty on the evidence slice itself. Auxiliary terms score aspect quality, slice validity, and sentiment/language membership. The final GRPO reward is a fixed weighted sum with the insight term carrying weight 5.0.

**Extraction results.** Table 3 reports the four-way comparison. The main movement is in abstraction, the dimension targeted by the anti-copy reward. Overall production quality rises from 59.3% for the base model to 93.1% after GRPO.

## C Gold Construction and Leakage Controls

We construct product-level gold clusters in three layers. Layer 1 builds candidate clusters from BGE-small embeddings using the intersection of Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) and agglomerative clustering, keeping a high-precision core. Layer 2 uses a teacher LLM to validate members, rewrite canonical names, merge near-duplicates, and reassign orphans. Layer 3 uses Gemini 2.5 Pro to rank the resulting clusters with 1–5 importance scores and to audit a sample of member assignments.

The gold is used in three different ways: training PROJHEAD, synthesizing $K$-way reducer SFT examples, and evaluating held-out products. To avoid product-level leakage, all reported test products are excluded from both PROJHEAD training and reducer SFT. We do not describe this as a purely human gold standard: Gemini participates in both gold construction and evaluation, and we treat that dependency as a limitation rather than an anti-circularity guarantee.

The annotation pool contains 4,828 Software products, 1,112 Fashion products, and 8,090 Baby Products. The main reported benchmarks use fixed 100-product held-out slices per category, with a separate small large-$N$ Fashion stress test.

## D Projection-Head Training Details

PROJHEAD is a 384→256→128 MLP trained per category for 3 epochs with the information noise-contrastive estimation (InfoNCE) loss ($\tau = 0.07$). For each training product, we sample anchors from Layer-3 product clusters, choose positives from the same cluster, hard negatives from other clusters of the same product, and easy negatives from random products. The projection is small ($< 200$K trainable parameters) and leaves the frozen BGE encoder unchanged. This lets the system adapt to product-review cluster boundaries without overwriting the broad-domain similarity prior.

The hard negatives are product-local by design. Across products, many phrases are trivially different because they refer to different items, brands, or domains; those negatives teach little about the boundary between two themes within the same product page. Within one product, however, lexically similar tuples such as *installation failure*, *device compatibility*, and *performance lag* often share the same app or product name. These are exactly the cases where raw BGE over-merges and where the projection head should reshape the space.

## E K-way Reducer SFT Details

A first reducer checkpoint was trained on Layer-2 cleanup examples where each input was already one coherent cluster and the target was a single canonical phrase. Sampling 5,000 examples showed that 100% of targets were $K = 1$. The

| Model | overall≥4 | overall | correctness | completeness | diversity | **abstraction** |
|---|---|---|---|---|---|---|
| base (Qwen3.5-9B) | 59.3% | 3.69 | 4.70 | 4.69 | 4.79 | 3.40 |
| SFT (190K teacher labels) | 88.7% | 4.52 | 4.80 | 4.78 | 4.89 | 4.49 |
| DPO (chosen=teacher/rej=SFT) | 90.8% | 4.62 | 4.85 | 4.80 | 4.94 | 4.64 |
| **GRPO (anti-copy)** | **93.1%** | **4.71** | 4.85 | 4.82 | 4.95 | **4.78** |

Table 3: Stage-1 extraction quality on 5,000 held-out Software reviews, judged by Gemini 2.5 Pro.

checkpoint therefore over-collapsed mixed-theme chunks.

We replace this with $K$-way SFT data synthesized from training products: for each product, sample $k_{\text{in}} \in \{2, 3, 5, 7, 10\}$ gold clusters, draw at most 200 tuples, shuffle them, and target the corresponding gold partition. For Software this yields 18,304 train and 956 validation examples. The same trained reducer is used for cluster naming and final ranking.

This data format matches the reducer's inference-time role. A reducer call may receive a pure cluster or a mixed chunk produced by an imperfect spectral partition, so the target must teach both naming and controlled splitting. In contrast, the earlier $K = 1$ cleanup data taught the model that every input should collapse to one phrase. The $K$-way data fixes that supervision mismatch without changing the inference algorithm.

## F Evaluation Protocol Details

**Why coverage is reported separately.** The task has two different desiderata that can conflict. NDCG rewards a system for naming the most important gold themes near the top of the list, but it does not check whether all input evidence was assigned to some output cluster. A flat LLM can therefore obtain a strong NDCG by selectively naming head themes and omitting low-confidence member ids. This behavior is useful to measure rather than hide: for shopper-facing copy it may be acceptable, but for product analytics, debugging, and trust-and-safety audits the system must account for the full evidence inventory. We therefore treat coverage and duplicate member assignments as first-class metrics rather than implementation details.

**Matching predicted and gold clusters.** We match predicted canonical names to gold names only after the model has produced its full top-$K$ list. Matching is one-to-one and greedy by phrase-token overlap, so duplicate aliases do not receive repeated credit. This deliberately conservative protocol penalizes two common mistakes: generic cluster names that cover many different gold themes, and repeated names assigned to distinct member sets. The protocol is still imperfect, because two semantically equivalent names can have low token overlap; we discuss this in the Limitations section.

| Strategy | recall | prec | NDCG | **cov** | **dup** |
|---|---|---|---|---|---|
| flat_llm | 0.480 | 0.491 | 0.481 | 0.228 | 10 |
| batch_direct | 0.380 | 0.380 | 0.478 | 0.490 | 1 |
| embed_cluster_llm | 0.400 | 0.400 | **0.486** | 0.977 | 0 |
| embed_cluster | 0.340 | 0.340 | 0.394 | 1.000 | 0 |
| **GRAPHREDUCE** | 0.360 | 0.360 | 0.475 | **1.000** | **0** |

Table 4: Large-$N$ Fashion stress test (5 products, $N \approx 1{,}500$, all metrics @10). The flat LLM keeps plausible head themes but drops most member assignments (coverage $0.228$).

**Split discipline.** All Stage-2 test products are held out before training either the projection head or the reducer. This is stricter than tuple-level splitting: the model cannot see other tuples from the same product during training and then be evaluated on a held-out subset of that product.

## G Large-N Stress Test

Table 4 reports the full large-$N$ Fashion stress test referenced in Section 4.

## H Significance and Cost

**Significance.** Table 5 gives paired tests for the key lossless comparison, GRAPHREDUCE vs. hier_agglom, over the $n$=100 products each category shares. We report the mean per-product metric difference, a 95% confidence interval from a deterministic 10,000-sample paired bootstrap, and a two-sided Wilcoxon signed-rank $p$-value. The recall@10 gain is significant in all three categories with CIs that exclude zero; the NDCG@10 gain is positive everywhere but its CI includes zero, so we report NDCG parity rather than improvement.

**Cost and latency.** Table 6 reports per-product LLM calls and wall-clock latency (median and 90th percentile). Although GRAPHREDUCE issues

| Category | Metric | Δ (95% CI) | Wilcoxon |
|---|---|---|---|
| Software | recall@10 | +0.034 [+0.006, +0.062] | $p$=0.015 * |
| | NDCG@10 | +0.020 [−0.005, +0.045] | $p$=0.16 ns |
| Fashion | recall@10 | +0.034 [+0.009, +0.059] | $p$=0.010 ** |
| | NDCG@10 | +0.020 [−0.005, +0.045] | $p$=0.29 ns |
| Baby | recall@10 | +0.056 [+0.023, +0.090] | $p$=0.005 ** |
| | NDCG@10 | +0.017 [−0.016, +0.052] | $p$=0.30 ns |

Table 5: Paired significance of GRAPHREDUCE over hier_agglom ($n$=100 products/category). Recall gains are significant; NDCG gains are within bootstrap noise.

| Category | Strategy | calls | med. (s) | p90 (s) |
|---|---|---|---|---|
| Software | flat_llm | 1.0 | 13.4 | 21.0 |
| | BERTopic | 0.0 | 0.2 | 0.3 |
| | hier_agglom | 10.0 | 16.4 | 21.7 |
| | GRAPHREDUCE | 11.0 | 14.8 | 19.5 |
| Fashion | flat_llm | 1.0 | 27.2 | 37.8 |
| | BERTopic | 0.0 | 0.6 | 1.1 |
| | hier_agglom | 10.0 | 27.9 | 39.5 |
| | GRAPHREDUCE | 11.0 | 26.8 | 40.1 |
| Baby | flat_llm | 1.0 | 21.9 | 34.5 |
| | BERTopic | 0.0 | 0.1 | 0.2 |
| | hier_agglom | 10.0 | 29.2 | 37.8 |
| | GRAPHREDUCE | 10.6 | 26.7 | 32.8 |

Table 6: Per-product cost and latency. GRAPHREDUCE's many small reducer calls match or beat the single-call flat LLM in wall-clock time.

10–11 reducer calls versus the flat LLM's single call, its median latency is comparable to or below flat prompting and below hier_agglom, because each call operates on a bounded chunk ($\leq B$=200 tuples) rather than one prompt over all $N$ tuples. BERTopic is by far the cheapest (no LLM calls, sub-second after a one-time UMAP/HDBSCAN warm-up) but, as Table 1 shows, only by dropping 11.7–39.9% of the tuple inventory. Losslessness therefore carries no latency penalty over the flat baseline.

## I Full Ablation Results

Table 7 isolates the aggregation components on a 30-product Software development benchmark. The trained projection head and the chunk-plus-rank pipeline are both necessary. We call the learned soft-clustering negative baseline GraphPool; it uses a GraphSAGE encoder (Hamilton et al., 2017) and Gumbel-Softmax straight-through assignments. Training GraphPool with GRPO under a local reward hurts NDCG.

The negative result is important for the final system design. The local Stage-C reward combined coverage, balance, phrase distinctness, and phrase quality, but it was not aligned with NDCG@10. During GRPO, training reward rose while NDCG drifted down. We therefore keep the contrastive projection and remove the differentiable clustering head.

## J Qualitative Analysis

Table 8 shows one Software product. The example is not presented as a win on every gold phrase: GRAPHREDUCE recovers *device compatibility* and a more specific gameplay phrase, but it still misses the gold's fine-grained *blue crystal inaccessible* wording. This illustrates the measured pattern: the method improves recall and coverage over raw embeddings, while phrase aliasing remains a source of NDCG loss.

Our manual error analysis of low-NDCG GRAPHREDUCE predictions finds four recurring issues: reducer phrase aliasing, long-tail themes competing with head themes, spectral over-merge at very small $N$, and incorrect cluster names despite correct member sets. The first issue is the most common: two semantically distinct clusters can receive the same generic name, such as *app crashes*. This hurts phrase-match NDCG even when the underlying member partition is reasonable.

## K Extended Discussion

**Why the projection helps.** Raw BGE embeddings are strong but generic. In our products they often place device compatibility, installation, and performance tuples near one another because the same product names appear in all three. The projection head is trained on product-level cluster boundaries, so it can separate themes that are lexically similar but operationally different for shoppers.

Three recurring examples drive the gain. First, compatibility complaints often mention the same device names as installation complaints, even though shoppers treat them as different purchase risks. Second, paid-app frustration and in-app-purchase frustration share many tokens but map to different gold clusters. Third, generic predicates such as "works" or "can't open" dominate raw sentence similarity even when the theme-bearing noun phrase differs. The projection head does not need to learn review language from scratch; it only needs to rotate the frozen embedding space toward these product-level boundaries.

**Why flat LLMs can win NDCG.** Phrase-level NDCG rewards matching high-importance gold themes. A flat LLM can implicitly focus on those

| Configuration | recall@10 | prec@10 | NDCG@10 | cov |
|---|---|---|---|---|
| BGE raw + sklearn agglom + size rank | 0.595 | 0.507 | 0.651 | 0.947 |
| BGE raw + chunk + spectral + LLM rank | 0.589 | 0.503 | 0.680 | 1.000 |
| +PROJHEAD (Layer-2 pairs) + sklearn | 0.533 | 0.522 | 0.606 | 1.000 |
| +PROJHEAD (Layer-2 pairs) + chunk+rank | 0.632 | 0.533 | 0.686 | 1.000 |
| +PROJHEAD (Layer-3 train pairs) + sklearn | 0.529 | 0.532 | 0.602 | 1.000 |
| **+PROJHEAD (Layer-3 train pairs) + chunk+rank** | **0.652** | **0.550** | **0.700** | **1.000** |
| *Projection dimension sweep, full pipeline* | | | | |
| $d = 64$ | 0.622 | 0.527 | 0.678 | 1.000 |
| $d = 128$ | **0.652** | **0.550** | **0.700** | **1.000** |
| $d = 256$ | 0.628 | 0.533 | 0.680 | 1.000 |
| *Training length sweep* | | | | |
| epochs $= 3$ | 0.652 | 0.550 | 0.700 | 1.000 |
| epochs $= 5$ (union L2+L3 pairs) | 0.618 | 0.523 | 0.676 | 1.000 |
| epochs $= 10$ | 0.609 | 0.507 | 0.664 | 1.000 |
| *Negative:* GraphPool (GraphSAGE) | 0.460 | 0.390 | 0.518 | 1.000 |
| *Negative:* GraphPool + GRPO | 0.478 | 0.404 | 0.537 | 1.000 |

Table 7: Software development ablations. A small learned projection helps; a learned soft-clustering head trained with a local reward does not.

| Method | Top-3 themes (asin B008BBB388) |
|---|---|
| flat_llm | paid app, app crashes, gameplay |
| hier_agglom | gameplay, inaccessible items, app crashes |
| **GRAPHREDUCE** | **hidden object gameplay, device compatibility, paid app** |
| Gold top-3 | progression blocker, device compatibility, blue crystal inaccessible |

Table 8: Example top-3 cluster names for one Software product ($N = 57$ tuples, 12 gold clusters).

head themes and omit lower-confidence member assignments. GRAPHREDUCE must assign every tuple to one of the output clusters, including long-tail evidence. This makes coverage and recall necessary companion metrics to NDCG.

**Where the method breaks.** The current fixed $k = 20$ graph can be too dense for products with $N < 40$, and large clusters in Fashion sometimes absorb finer themes such as *high waist* and *tummy control* into a broad *good fit* phrase. Adaptive graph degree and a diversity-aware ranking objective are natural next steps.

The other recurring issue is phrase aliasing. Two clusters can have reasonable member sets but receive the same generic name from the reducer, such as *app crashes*. This hurts NDCG because only one alias can match the corresponding gold theme, and it weakens the shopper-facing output because the top-$K$ list becomes less diverse. A future reducer should condition on already selected names or use a lightweight diversity penalty at ranking time.