Fast Retrieval, Unequal Exposure: Evaluating Efficiency and Exposure Bias in Vector-Based Recommender Systems

Enock Okorno Ayiku, Tarikul Islam Papon, Navkar Jain, and Kenneth Fletcher
University of Massachusetts Boston
United States
Email: enock.ayiku001@umb.edu, t.papon@umb.edu, n.jain001@umb.edu, kenneth.fletcher@umb.edu

***Abstract*—Vector retrieval is central to modern recommender systems because it determines which items enter the candidate set before ranking. While vector search is often evaluated by speed and retrieval accuracy, it also shapes catalog exposure and can preserve popularity bias. This paper studies the efficiency exposure trade-off across three retrieval architectures: exhaustive SQL dot-product scan in PostgreSQL, exact external vector retrieval using FAISS, and database-integrated vector retrieval using PostgreSQL with pgvector. Using the same two-tower model trained on MovieLens-1M, we compare latency, throughput, Recall@10, NDCG@10, catalog coverage, long-tail exposure, and Gini exposure inequality. FAISS produces the same recommendations as exhaustive SQL retrieval while reducing retrieval time from 89.74 seconds to 0.67 seconds. Exact retrieval preserves recommendation quality and exposure concentration, showing that retrieval acceleration alone does not reduce popularity-driven exposure inequality. In contrast, pgvector index configuration affects both relevance and exposure: HNSW remains highly concentrated, while IVFFLAT increases catalog coverage and reduces Gini exposure inequality at the cost of lower Recall@10 and NDCG@10. Across the main retrieval backends, exposure remains highly skewed, with Gini above 0.99 and near-zero long-tail exposure. Even when IVFFLAT reduces Gini to 0.9467, exposure remains concentrated. These results show that retrieval backends should be evaluated not only by latency and throughput, but also by catalog coverage, long-tail exposure, and exposure inequality before downstream ranking is applied.**

***Index Terms*—Recommender systems, vector retrieval, candidate generation, FAISS, pgvector, PostgreSQL, exposure bias, popularity bias.**

I. Introduction

Modern recommender systems are often organized as multi-stage pipelines in which a retrieval stage first selects candidate items and a ranking stage then orders them for final recommendation. Because items excluded during retrieval cannot be recovered by later ranking or re-ranking, candidate generation acts as both an efficiency mechanism and a visibility gatekeeper for the item catalog. Embedding-based retrieval is commonly used for this stage. Users and items are represented as dense vectors, and candidate items are selected using dot-product or nearest-neighbor search. This search can be implemented through different retrieval architectures, including exhaustive SQL dot-product scans, external vector-search libraries such as FAISS, and database-integrated vector retrieval systems such as PostgreSQL with pgvector. These

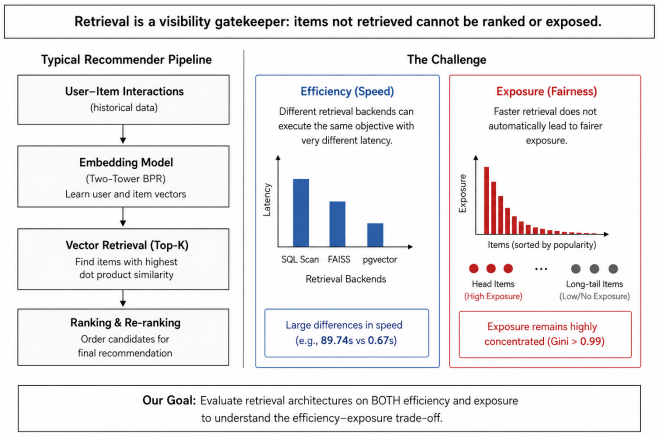

Fig. 1: Overview of the efficiency exposure problem in candidate generation. The left panel shows a typical recommender pipeline; the right panel illustrates that retrieval speed and exposure fairness are independent properties of the backend.

three architectures represent distinct points in the retrieval design space: exhaustive relational execution without a vector index, exact external vector search optimized for speed, and database native vector retrieval with support for both exact and approximate indexing. These systems differ in speed, database integration, and retrieval behavior. However, improving retrieval speed does not necessarily improve item exposure. If the learned embeddings reflect popularity skewed historical interactions, a faster retrieval backend may simply retrieve the same popular items more efficiently. Similarly, indexed or approximate retrieval may alter the candidate set in ways that affect both ranking quality and catalog coverage. This creates a need to evaluate retrieval architectures not only by efficiency, but also by their effect on exposure concentration. We measure exposure concentration using catalog coverage, long-tail exposure, and Gini exposure inequality, which together capture how broadly recommendations are distributed across the item catalog and how much exposure is concentrated among a small number of popular items.

This paper provides a controlled retrieval-layer evaluation of embedding-based candidate generation. Rather than proposing a new recommendation model, we hold the learned embeddings, scoring function, dataset split, and seen-item filtering

fixed, and vary only the retrieval backend. This design separates retrieval execution effects from model-training effects.

Using MovieLens-1M and a two-tower BPR model, we compare three retrieval architectures: PostgreSQL SQL dot-product scan, FAISS exact vector retrieval [1], and PostgreSQL with pgvector. We evaluate each architecture using efficiency metrics, recommendation quality metrics, and exposure concentration metrics.

This paper addresses three questions. First, how do retrieval backends differ in efficiency when generating candidates from the same learned embeddings? Second, does the retrieval backend choice affect recommendation quality and catalog exposure concentration? Third, can retrieval acceleration alone reduce popularity driven exposure inequality?

The main contributions of this paper are as follows:

- We frame vector retrieval as a retrieval-layer audit problem, where candidate generation is evaluated not only by speed and relevance, but also by catalog exposure and concentration.
- We design a controlled comparison of PostgreSQL SQL scan, FAISS exact retrieval, and pgvector while holding the embeddings, scoring function, dataset split, and seen-item filtering fixed.
- We show that retrieval acceleration alone does not reduce exposure inequality: FAISS achieves a $133\times$ speedup over SQL scan while producing identical recommendation quality and exposure concentration.
- We show that pgvector index choice changes the relevance exposure trade-off: IVFFLAT broadens coverage and lowers Gini exposure inequality, but reduces Recall@10 and NDCG@10.

II. Related Work

A. Candidate Generation and Vector Retrieval

Large-scale recommender systems typically use multi-stage pipelines: a retrieval stage selects a small candidate set and a ranking stage applies more expensive models to order them [2]. Because items not retrieved cannot be ranked or exposed later, candidate generation is a critical visibility control point. Embedding-based recommendation methods, including matrix factorization, BPR, Neural Collaborative Filtering, NGCF, and LightGCN, learn user and item representations that can be searched using dot product or cosine similarity [3], [4], [5], [6], [7]. This makes top-k recommendation closely related to maximum inner-product search.

Vector search systems such as FAISS support efficient exact and approximate similarity search over dense embeddings [8], [1]. Approximate methods, including product quantization and HNSW, improve scalability by trading exactness for speed and memory efficiency [9], [10]. At the same time, practical recommender systems often require metadata filtering, joins, transactions, and hybrid search, motivating SQL-integrated vector retrieval systems and vector databases [11], [12]. PostgreSQL with pgvector provides an accessible example of database-native vector retrieval with support for exact search and approximate indexes such as IVFFLAT and HNSW [13]. While prior work mainly emphasizes speed, recall, and scalability, this paper evaluates retrieval architectures using both efficiency and exposure outcomes.

B. Popularity Bias and Fairness of Exposure

Popularity bias is a persistent challenge in recommender systems because historical interactions are often concentrated around popular items, causing models to over-recommend head items and under-expose long-tail items. Prior work shows that popularity bias can create unfair visibility outcomes and that improving long-tail coverage often involves accuracy–coverage trade-offs [14], [15]. Recent surveys identify popularity bias as a central issue in recommender-system evaluation and debiasing [16], [17]. Related work on diversity and long-tail recommendation studies aggregate diversity, catalog coverage, and beyond-accuracy objectives [18], [19], [20].

Fairness aware and multi-sided recommendation research further examines how recommendation outcomes affect users, items, providers, and platforms [21], [22], [23], [24], [25], [26], [27]. Fairness-of-exposure work focuses on how attention is allocated across ranked outputs, including exposure constraints, equity of attention, and fair ranking methods [28], [29], [30], [31]. This paper connects these concerns to the retrieval layer by asking whether different retrieval backends affect catalog coverage, long-tail exposure, and exposure concentration before ranking or re-ranking can intervene.

III. Methodology

Figure 2 summarizes the experimental pipeline used in this study. The same dataset, preprocessing procedure are used across retrieval backends so that differences in efficiency, recommendation quality, and exposure concentration can be attributed to the retrieval architecture.

A. Dataset

We evaluate on MovieLens-1M, a standard benchmark dataset for recommender-system research [32]. The dataset contains approximately 1 million ratings from 6,040 users on 3,706 movies. We treat ratings as implicit feedback and retain all user-item interactions. User and item identifiers are converted to zero based indices. The data is randomly shuffled and split into 80% training and 20% testing, yielding 6,033 users with at least one test interaction. The training set is used to learn embeddings and to identify already seen items.

B. Embedding Model

We train a simple two-tower embedding model. Each user u is represented by an embedding vector $\mathbf{u} \in \mathbb{R}^{64}$, and each item i is represented by an embedding vector $\mathbf{v}_i \in \mathbb{R}^{64}$. The relevance score is the inner product:

$$s(u, i) = \mathbf{u}^\top \mathbf{v}_i. \tag{1}$$

The model is trained using Bayesian Personalized Ranking (BPR) loss [4]. For a user u, observed item i, and sampled negative item j, the loss is:

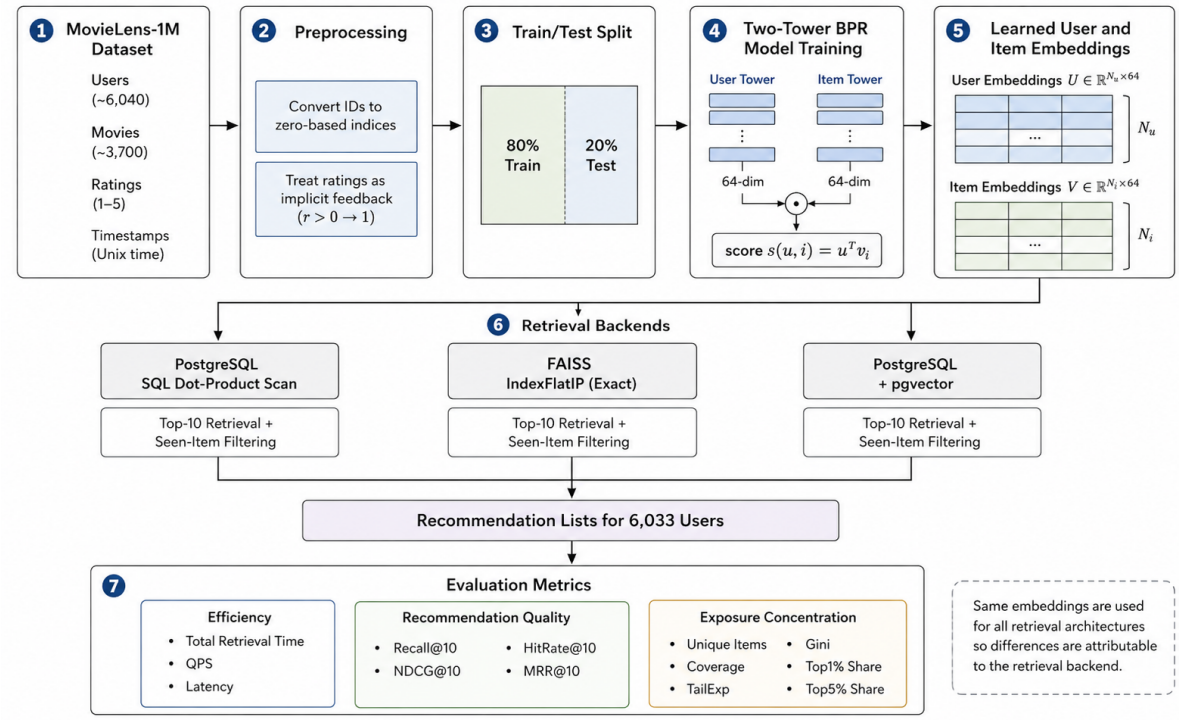

Fig. 2: Overview of the experimental pipeline. MovieLens-1M interactions are used to train a two-tower BPR model. The same learned embeddings are then queried using SQL scan, FAISS exact retrieval, and pgvector, and evaluated using efficiency, relevance, and exposure metrics.

$$\mathcal{L}_{BPR} = -\log \sigma(s(u,i) - s(u,j)). \tag{2}$$

After training, the same learned user and item embeddings are used for every retrieval architecture. This ensures that differences in performance and exposure are caused by the retrieval backend, not by different models.

C. Retrieval Architectures

1) PostgreSQL SQL Dot-Product Scan: The first architecture stores each item's embedding dimension as a separate numeric column in PostgreSQL:

$$(e_{i,1}, e_{i,2}, \ldots, e_{i,64}). \tag{3}$$

For each user, PostgreSQL computes the exact dot product:

$$s(u,i) = \sum_{d=1}^{64} u_d e_{i,d}. \tag{4}$$

The query orders items by this score and returns the top-10 items, excluding those observed in the user's training history. This architecture represents exhaustive relational execution of the embedding retrieval objective without a vector index.

2) FAISS Exact Vector Retrieval: The second architecture uses a FAISS IndexFlatIP index to retrieve the top 10 items by exact inner product. Because IndexFlatIP is exact, it optimizes the same ranking objective as the PostgreSQL SQL dot-product scan. We compare the two to isolate the efficiency benefit of vector-optimized retrieval.

3) PostgreSQL + pgvector Retrieval: The third architecture stores item embeddings in PostgreSQL using the pgvector extension and generates recommendations through database-native inner-product vector search. In the main retrieval comparison, we use the pgvector configuration labeled in Tables I and II.

D. Implementation Environment

Experiments were run in a local Paperspace environment with PostgreSQL 14.22. The pgvector extension was installed from source and enabled inside the MovieLens database. FAISS retrieval used IndexFlatIP. All retrieval-time results measure top-10 recommendation generation for 6,033 evaluated users after embeddings were trained and after tables or indexes were created. Retrieval time excludes model training time. Index construction time is not included in the retrieval-time column. This design focuses the comparison on online candidate-generation cost.

E. Evaluation Metrics

We evaluate three classes of metrics.

TABLE I: Retrieval efficiency across architectures.

System	Time (s)	QPS	Latency (ms)
PostgreSQL SQL Scan	89.7379	67.23	14.8745
FAISS Exact Vector	0.6736	8956.40	0.1117
PostgreSQL + pgvector (HNSW)	66.1431	91.21	10.9636

1) Efficiency Metrics: We report:

- **Total retrieval time**: wall-clock time to generate top-10 recommendations for all evaluated users.
- **Queries per second (QPS)**: number of users processed per second.
- **Latency per user**: average retrieval time per user in milliseconds.

2) Recommendation Quality Metrics: We evaluate recommendation quality using:

- **Recall@10**: fraction of held-out test items recovered in the top-10 recommendation list.
- **NDCG@10**: ranking-sensitive relevance score.
- **HitRate@10**: fraction of users for whom at least one held-out item appears in the top-10 list.
- **MRR@10**: reciprocal rank of the first relevant item in the top-10 list.

3) Exposure Concentration Metrics: Let I denote the item catalog, R_u^K the top-K recommendation list for user u, and c_i the number of times item i appears across all recommendation lists:

$$c_i = \sum_u \mathbb{I}(i \in R_u^K). \tag{5}$$

Coverage@K is defined as:

$$\text{Coverage@K} = \frac{|\{i \in I : c_i > 0\}|}{|I|}. \tag{6}$$

TailExp@K is the fraction of total recommendation exposure assigned to tail items:

$$\text{TailExp@K} = \frac{\sum_{i \in I_{\text{tail}}} c_i}{\sum_{i \in I} c_i}. \tag{7}$$

We also report UniqueItems, Gini exposure inequality. These capture the number of distinct exposed items, inequality in item exposure, and the concentration of exposure among the most exposed 1% and 5% of items. We define head items as the top 20% most frequent items in the training set and tail items as the remaining 80%.

IV. Results

A. Retrieval Efficiency

Table I reports retrieval efficiency. All systems use the same trained embeddings, inner-product scoring function, and seen-item filtering; only the retrieval backend changes.

FAISS gives the largest efficiency gain, reducing retrieval time from 89.74 seconds to 0.67 seconds. This is a 133.2× speedup over SQL scan and a 98.2× speedup over pgvector. In this setup, pgvector is only 1.36× faster than SQL scan.

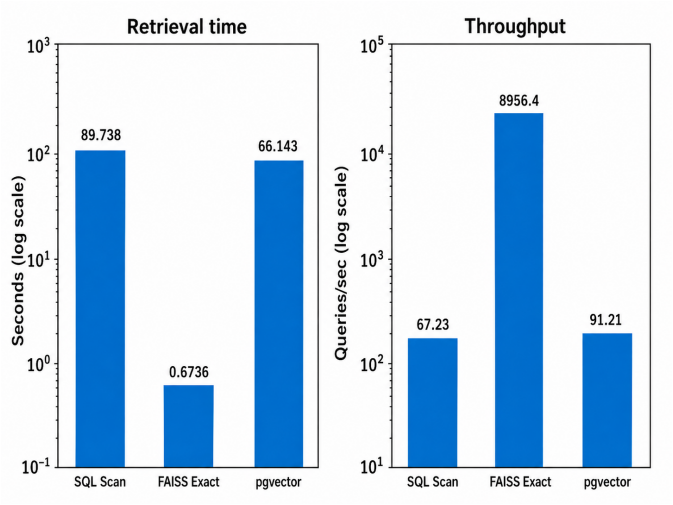

Fig. 3: Efficiency comparison across retrieval architectures.

B. Recommendation Quality and Exposure

Table II reports recommendation quality and exposure concentration. SQL scan and FAISS exact retrieval have identical values because both compute the same inner-product ranking under the same filtering rule. Thus, FAISS improves efficiency without changing the recommendation output. pgvector has slightly lower Recall@10 and NDCG@10 and exposes fewer unique items than SQL scan and FAISS. More importantly, all systems show severe exposure concentration: TailExp@10 is approximately 0.0005 and Gini exceeds 0.99 across all backends. These results show that retrieval acceleration alone does not reduce popularity driven exposure inequality.

V. Ablation and Sensitivity Analysis

We conduct three ablation analyses to test whether the main findings are sensitive to retrieval depth, embedding model, or pgvector index configuration. Latency values in this section are not directly comparable to Table I, as ablations were run in a separate controlled script. Table III compares $K = 10$ and $K = 20$. Increasing retrieval depth improves Recall@K and Coverage@K, but TailExp@K remains near zero and Gini remains high, suggesting that larger candidate sets alone do not meaningfully improve long-tail exposure.

Table IV compares the two-tower BPR model with MF-BPR and LightGCN. MF-BPR achieves the strongest relevance and exposure results in this experiment, but all models remain highly exposure concentrated.

Table V compares HNSW, IVFFLAT, and exact/no-index pgvector retrieval. HNSW preserves recommendation quality but remains concentrated, while IVFFLAT improves catalog coverage and reduces Gini at the cost of lower Recall@10 and NDCG@10. Overall, the ablations show that retrieval depth, model architecture, and index configuration affect the quality–exposure trade-off, but none fully resolve long-tail under-exposure.

TABLE II: Recommendation quality and exposure concentration across retrieval architectures.

System	Recall@10	NDCG@10	HitRate@10	MRR@10	Unique	Coverage	TailExp	Gini↓
PostgreSQL SQL Scan	0.0784	0.2340	0.7202	0.4154	229	0.0579	0.0005	0.9926
FAISS Exact Vector	0.0784	0.2340	0.7202	0.4154	229	0.0579	0.0005	0.9926
PostgreSQL + pgvector	0.0778	0.2300	0.7184	0.4151	182	0.0461	0.0005	0.9930

TABLE III: Retrieval-depth sensitivity results. Best values are bolded and second-best values are underlined within each metric. Increasing K from 10 to 20 improves Recall@K and Coverage@K, but TailExp@K remains very low and Gini remains high.

System	K	Recall@K	NDCG@K	Coverage@K	TailExp@K	Gini↓	Latency (ms)↓
PostgreSQL SQL Scan	10	0.0784	**0.2340**	0.0579	0.0005	0.9926	4.4716
PostgreSQL SQL Scan	20	**0.1265**	0.2196	**0.0929**	**0.0007**	**0.9876**	4.5219
FAISS Exact Vector	10	0.0784	**0.2340**	0.0579	0.0005	0.9926	**0.1422**
FAISS Exact Vector	20	**0.1265**	0.2196	**0.0929**	**0.0007**	**0.9876**	0.1447
PostgreSQL + pgvector	10	0.0778	0.2300	0.0466	0.0005	0.9930	0.4954
PostgreSQL + pgvector	20	0.1186	0.1977	0.0585	**0.0007**	0.9901	0.5854

TABLE IV: Best values are bolded and second-best values are underlined within each metric. MF-BPR gives the strongest relevance and exposure results in this experiment, while FAISS remains much faster than SQL scan.

Model	Retrieval Backend	Recall@10	NDCG@10	Coverage@10	TailExp@10	Gini↓	Time (s)↓
Two-Tower BPR	PostgreSQL SQL Scan	0.0784	0.2340	0.0579	0.0005	0.9926	28.4363
Two-Tower BPR	FAISS Exact Vector	0.0784	0.2340	0.0579	0.0005	0.9926	0.8226
Two-Tower BPR	PostgreSQL + pgvector	0.0778	0.2300	0.0461	0.0005	0.9930	3.2093
MF-BPR	PostgreSQL SQL Scan	**0.0843**	**0.2538**	**0.0987**	0.0027	**0.9892**	29.8310
MF-BPR	FAISS Exact Vector	**0.0843**	**0.2538**	**0.0987**	0.0027	**0.9892**	0.8653
MF-BPR	PostgreSQL + pgvector	0.0837	0.2494	0.0863	**0.0028**	0.9900	3.3282
LightGCN	PostgreSQL SQL Scan	0.0678	0.2012	0.0230	0.0003	0.9952	29.1451
LightGCN	FAISS Exact Vector	0.0678	0.2012	0.0230	0.0003	0.9952	**0.8135**
LightGCN	PostgreSQL + pgvector	0.0676	0.1997	0.0157	0.0003	0.9953	2.9727

TABLE V: pgvector index-configuration sensitivity. Best values are bolded and second-best values are underlined within each metric.

Index Type	Time (s)↓	Recall @10	NDCG @10	Coverage @10	Gini↓
HNSW	2.9981	0.0778	0.2300	0.0458	0.9930
IVFFLAT	**2.3149**	0.0676	0.1880	**0.2619**	**0.9467**
Exact/No Index	7.1548	**0.0784**	**0.2340**	0.0579	0.9926

VI. Discussion

A. Integrated Vector Retrieval

The pgvector experiment represents database-integrated vector retrieval, where vector search can operate alongside SQL filters, joins, metadata constraints, and transactional data [11]. However, integration did not imply superior speed in the local experimental setup: pgvector was faster than an exhaustive SQL scan but much slower than FAISS. It also slightly reduced catalog coverage and increased exposure concentration, showing that integrated vector databases should be evaluated by both efficiency and exposure effects.

B. Retrieval Speed and Popularity Bias

All three retrieval architectures remain highly head-dominated. FAISS greatly improves speed, but because it uses the same embeddings and exact scoring objective as SQL scan, it preserves the same concentrated exposure pattern. Thus, faster retrieval can accelerate popularity-biased candidate generation without reducing the underlying exposure imbalance.

C. Exact and Approximate Retrieval Trade-offs

Exact retrieval mainly changes efficiency, not exposure. PostgreSQL SQL scan and FAISS exact retrieval return identical relevance and exposure metrics because both compute the same inner-product ranking under the same filtering rule. The pgvector index results show a different trade-off. IVFFLAT increases Coverage@10 from 0.0579 to 0.2619 and reduces Gini from 0.9926 to 0.9467, but lowers Recall@10 from 0.0784 to 0.0676 and NDCG@10 from 0.2340 to 0.1880. This suggests that approximate indexing can broaden exposure, but at a relevance cost.

VII. Limitations and Future Work

This study provides a controlled comparison of retrieval architectures, but it has several limitations. First, the main experiment uses a random train-test split; future work should use chronological or leave-last-out evaluation to better preserve temporal recommendation validity. Second, the experiments are conducted on MovieLens1M only, so additional datasets, including sparse and impression log datasets, are needed to test generalizability. Third, results are reported from a single run; future work should report multiple random seeds with variance. Future work should also include stronger baselines, such as popularity only retrieval, random retrieval, and exposure-aware retrieval methods. In addition, the current embeddings

use interaction IDs only and do not incorporate item content features such as genre or release year, which may contribute to exposure concentration. Finally, the model-ablation results suggest that embedding architecture can affect exposure more strongly than retrieval backend choice. Future retrieval-layer audits should therefore study model and index choices jointly.

VIII. Conclusion

This paper compared SQL dot-product retrieval, FAISS exact vector retrieval, and PostgreSQL with pgvector for recommender-system candidate generation. Using the same two-tower embeddings on MovieLens-1M, FAISS produced identical recommendations to SQL retrieval while achieving about 133× faster retrieval. This confirms that exact retrieval can substantially improve efficiency while preserving both ranking quality and exposure concentration.

The pgvector index analysis shows a different trade-off. IVFFLAT increased catalog coverage and reduced Gini exposure inequality, but lowered Recall@10 and NDCG@10. This suggests that approximate vector indexing can broaden exposure by perturbing the retrieved candidate set, although with a relevance cost. Across all configurations, exposure remains highly skewed, with Gini above 0.99 and near-zero long-tail exposure, showing that retrieval acceleration alone does not address popularity driven exposure inequality. Based on these, we recommend that retrieval-layer audits evaluate backends not only by latency and throughput, but also by catalog coverage, long-tail exposure, exposure Gini, and top-item exposure share before downstream ranking is applied.

References

[1] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," *arXiv preprint arXiv:2401.08281*, 2024.

[2] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 173–182.

[6] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 165–174.

[7] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2020, pp. 639–648.

[8] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[9] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[10] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[11] C. Chen, C. Jin, Y. Zhang, S. Podolsky, C. Wu, S.-P. Wang, E. Hanson, Z. Sun, R. Walzer, and J. Wang, "Singlestore-v: An integrated vector database system in singlestore," *Proceedings of the VLDB Endowment*, vol. 17, no. 12, pp. 3772–3785, 2024.

[12] Y. Wang, Y. Zhang, Z. Sun, L. Li, C. Jin, and J. Wang, "Are there fundamental limitations in supporting vector data management in relational databases?" in *Proceedings of the IEEE International Conference on Data Engineering*, 2024.

[13] pgvector contributors, "pgvector: Open-source vector similarity search for postgresql," https://github.com/pgvector/pgvector, 2024.

[14] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The unfairness of popularity bias in recommendation," in *Proceedings of the Workshop on Recommendation in Multi-Stakeholder Environments*, 2019.

[15] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 2017, pp. 42–46.

[16] A. Klimashevskaia, D. Jannach, M. Elahi, and C. Trattner, "A survey on popularity bias in recommender systems," *User Modeling and User-Adapted Interaction*, 2024.

[17] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: A survey and future directions," *ACM Transactions on Information Systems*, vol. 41, no. 3, pp. 1–39, 2023.

[18] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 896–911, 2012.

[19] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4511–4515, 2010.

[20] M. Kunaver and T. Požrl, "Diversity in recommender systems–a survey," *Knowledge-Based Systems*, vol. 123, pp. 154–162, 2017.

[21] R. Burke, "Multisided fairness for recommendation," *arXiv preprint arXiv:1707.00093*, 2017.

[22] H. Abdollahpouri and M. Mansoury, "Multi-sided exposure bias in recommendation," in *Proceedings of the KDD Workshop on Industrial Recommendation Systems*, 2020.

[23] S. Yao and B. Huang, "Beyond parity: Fairness objectives for collaborative filtering," in *Advances in Neural Information Processing Systems*, 2017, pp. 2921–2930.

[24] Y. Li, H. Chen, Z. Fu, Y. Ge, and Y. Zhang, "User-oriented fairness in recommendation," in *Proceedings of the Web Conference*, 2021, pp. 624–632.

[25] Y. Wang, W. Ma, M. Zhang, Y. Liu, and S. Ma, "A survey on the fairness of recommender systems," *ACM Transactions on Information Systems*, vol. 41, no. 3, pp. 1–43, 2023.

[26] D. Jin, L. Wang, H. Zhang, Y. Zheng, W. Ding, F. Xia, and S. Pan, "A survey on fairness-aware recommender systems," *Information Fusion*, vol. 100, p. 101906, 2023.

[27] Y. Zhao, J. Chen, Z. Zhang, X. Wang, and X. He, "Fairness and diversity in recommender systems: A survey," *arXiv preprint arXiv:2406.11323*, 2024.

[28] A. Singh and T. Joachims, "Fairness of exposure in rankings," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018, pp. 2219–2228.

[29] A. J. Biega, K. P. Gummadi, and G. Weikum, "Equity of attention: Amortizing individual fairness in rankings," in *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2018, pp. 405–414.

[30] E. Pitoura, K. Stefanidis, and G. Koutrika, "Fairness in rankings and recommendations: An overview," *The VLDB Journal*, vol. 31, pp. 431–458, 2022.

[31] N. Usunier, V. Do, and E. Dohmatob, "Fast online ranking with fairness of exposure," *arXiv preprint arXiv:2209.13019*, 2022.

[32] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, 2015.