Real-Time Hard Negative Sampling via LLM-based Clustering for Large-Scale Two-Tower Retrieval

Ivan Ji*
Meta
Menlo Park, CA, USA
ivanji@meta.com

Liuyi Hu*
Meta
Menlo Park, CA, USA
liuyihu@meta.com

Harrison (Zihao) Zhao*
Meta
Menlo Park, CA, USA
harrisonzhao@meta.com

Lei Huang
Meta
Menlo Park, CA, USA
leihuang@meta.com

Qunshu Zhang
Meta
Menlo Park, CA, USA
qunshuzhang@meta.com

Max (Xiangjun) Fan
Meta
Menlo Park, CA, USA
maxfan@meta.com

Aameek Singh
Meta
Menlo Park, CA, USA
aameeksingh@meta.com

Abstract

The two-tower model is widely used in the retrieval stage of large-scale recommendation systems, where training typically relies on in-batch and/or out-of-batch negative sampling. These methods, however, tend to produce easy negatives that the model learns quickly and that provide little training signal. This paper proposes a self-supervised, cluster-based hard negative sampling technique that draws negatives from the same semantic cluster as the positive item; in our production deployment the clusters are derived from large language model (LLM) based multimodal content representations, so that intra-cluster items are genuinely similar and yield informative negatives. To make this deployable at industrial scale, we realize the technique in a real-time, end-to-end framework that maintains a live in-memory item pool and draws cluster-based negatives from it on the fly via global out-of-batch sampling (GOOBS). The framework integrates directly into production two-tower training and serving and scales to billions of training examples with minimal computational overhead. Experiments on four public datasets and a 14-day online A/B test in a large-scale production system show that the proposed technique outperforms widely used industry methods, while also helping to break recommendation feedback loops and substantially reducing popularity bias.

CCS Concepts

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

*Three authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
OARS @ RecSys '26, Minneapolis, MN, USA
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/26/09
https://doi.org/10.1145/XXXXXXX.XXXXXXX

Keywords

recommender systems, two-tower models, retrieval, negative sampling, hard negatives, contrastive learning, popularity bias, large language models

ACM Reference Format:
Ivan Ji, Liuyi Hu, Harrison (Zihao) Zhao, Lei Huang, Qunshu Zhang, Max (Xiangjun) Fan, and Aameek Singh. 2026. Real-Time Hard Negative Sampling via LLM-based Clustering for Large-Scale Two-Tower Retrieval. In *Proceedings of RecSys 2026 Workshop on Online and Adaptive Recommender Systems (OARS @ RecSys '26).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 Introduction

In the era of big data, large-scale recommendation systems have become increasingly important in various applications, including e-commerce, social media, and entertainment. These systems aim to provide users with personalized recommendations that cater to their interests and preferences. However, designing an efficient and effective recommendation system is a challenging task due to the vast number of candidates eligible for selection. To address this challenge, modern recommendation systems typically adopt a multi-stage design, consisting of retrieval and ranking stages [8, 9, 18]. The retrieval stage aims to retrieve a small subset of relevant candidates from a large corpus, while the ranking stages further refine the selection to provide the most relevant recommendations. The two-tower model design is widely used as retrieval models in large scale applications due to its serving efficiency [9, 14].

Retrieval model training is often formulated as an extreme classification problem in which negative samples play a critical role. Various sampling-based techniques have been proposed to enhance training efficiency [3, 4, 9]. The widely used sampling techniques are in-batch negative sampling [11, 12] and out-of-batch (OOB) negative sampling and also mixed negative sampling [13, 27]. However, in-batch negatives are constrained by the mini-batch size, which can cause recommendation bias and limited exposure to a diverse corpus during training. Meanwhile, random OOB negatives are

often too easy for the model to learn, especially when the OOB item pool is large and diverse.

Besides, retrieval models typically use user engagement such as click as positives. However, whether user clicks or not depends on what item we surface to the user which is controlled by the multi-stage system. The strong feedback loop can result in popularity bias in the recommendation [5, 6, 19, 20].

To solve the problems of negatives being too easy and also popularity bias, this paper proposes a self-supervised cluster-based hard negative sampling technique for the two-tower model, together with a practical real-time framework that makes it deployable at industrial scale. The technique generates hard negatives from item clusters on the fly during training, while the framework integrates directly into a production two-tower training and serving pipeline, scaling to billions of examples with minimal overhead. On public datasets, experiments demonstrate that the proposed negative sampling technique yields significant performance improvements, particularly on large-scale datasets such as Amazon Reviews, which feature a vast number of unique items. When deployed as a retrieval source in a production recommendation system comprising multiple candidate sources, the proposed model achieves a +53% CTR improvement on its served impressions relative to the prior source model. The technique also helps mitigate popularity bias.

The contributions of this paper are summarized below.

- **Technique.** A self-supervised cluster-based hard negative sampling method for two-tower models that draws negatives from the same semantic cluster, using clusters derived from LLM-based multimodal content representations.
- **A practical real-time realization** that makes cluster-based sampling deployable in production two-tower training and serving, generating negatives on the fly with negligible overhead.
- Experiments on public and industrial datasets demonstrate consistent gains over the widely used in-batch and out-of-batch negative sampling methods.
- A production deployment demonstrates that the approach also mitigates popularity bias at industrial scale.

2 Related Work

2.1 Multi-stage System

For large scale recommendation systems, there are millions of candidates eligible for selection. Given the latency constraint, it is infeasible to rank all the candidates with a complicated ranking model. To balance recommendation efficiency and effectiveness, modern recommendation system typically adopts a multi-stage design: retrieval and ranking. The retrieval stage aims to retrieve a small subset from a large corpus. Then several ranking stages are deployed to rerank the retrieved subset. Such multi-stage system has been widely used in both industry and academia [8, 9, 18].

2.2 Two-Tower Model

The Two-Tower model, a variant of the neural network architecture, has been a powerful tool in recommendation systems since [15] introduced it. It is widely used in the retrieval stage of industry applications [9, 14]. One of the towers is typically used to model user interactions, and the other is used to model item characteristics. The user and item are represented by the embedding from the user and item tower. Then the retrieval problem is converted into a nearest neighbor (NN) search problem in the embedding space. The biggest advantage for this architecture is execution efficiency since the item embedding for the entire corpus can be precomputed and indexed so that online inference in real time is not required.

2.3 Negative Sampling

Retrieval model training is often formulated as an extreme classification problem, with various sampling-based techniques being proposed to enhance training efficiency [3, 4, 9].

2.3.1 In-batch Negative Sampling. It treats positive items of other users in the same mini-batch as negative items, rather than selecting from the corpus. It is commonly used owing to its time and memory efficiency [11, 12]. However, in-batch negatives are inherently limited by the mini-batch size. This can cause recommendation bias and limits the model's exposure to diverse candidates during training. To mitigate this, there are several variants being proposed. For example, [23] has proposed cross-batch negative sampling which takes advantage of the encoded items' embeddings from recent mini-batches to boost model training.

2.3.2 LogQ Correction. LogQ correction is a technique used in large-scale recommendation systems to correct for bias introduced by sampled softmax during training, where popular items are more likely to be sampled as negative examples. It has been well adopted in the industry by companies like Google [27, 28], ByteDance [26], Kuaishou [17]. It works by adjusting the model's logits (raw output scores) by subtracting the log-probability of a negative item's occurrence in the training batch, thereby penalizing popular items less and improving the model's ability to recall less frequent items. This correction helps ensure that the model learns from true preference signals rather than statistical artifacts of the sampling process.

2.3.3 Out-of-batch (OOB) Negative Sampling. Out-of-batch negative samples are selected from a pool of items that are not present in the current mini-batch of data being processed. By sampling from the items out of the current mini-batch, it provides a wider range of negative samples, potentially leading to better model generalization as the model encounters a broader variety of items that could be considered "not relevant". It also makes the training and serving more consistent since during serving time, the model needs to retrieve relevant items from the whole corpus instead of just items from the mini-batch. However, it can be computationally more expensive compared to in-batch sampling as it might require accessing and processing data outside the current mini-batch. Also, random OOB negative samples are very easy negatives for the model to distinguish. To overcome this obstacle, many different hard negative sampling methods have been proposed in research, such as Dynamic Negative Sampling [31], Adaptive Sampling [7, 22], and robust negative sampling [10]. But they are not widely adopted for large scale industrial applications due to the computational cost. In contrast, the method proposed in this paper samples hard negatives from semantic clusters in real time, requiring neither a global approximate-nearest-neighbor index (as in ANCE [25]) nor reliance on batch recency (as in cross-batch sampling [23]), which makes it well suited to industrial-scale streaming training.

2.3.4 Mixed Negative Sampling. Mixed Negative Sampling uses a mixture of in-batch and out-of-batch sampled negatives to tackle the selection bias of implicit user feedback. This technique is also widely used in various applications [13, 27].

2.4 Popularity Bias

The ideal state of recommendation system is to help find the most relevant items to users in a personalized way. However, in real applications, we often see popularity bias in the recommendation. Popularity bias means that the recommendation system tends to recommend rather popular items to users at the expense of less popular items that users may find relevant. The problem of popularity bias in recommendation systems has garnered significant attention from researchers over the past decade due to its practical importance [5, 6, 19, 20].

Overall, popularity bias in the recommendation system has the following negative effects: (1) making users interact with a limited number of items and therefore hurting user experiences; (2) no exploration on the long tail items to learn the embedding; (3) causing strong system feedback loop that is hard to break since already popular items will receive more exposures and become even more popular.

To mitigate, some methods directly try to correct the bias in the model training itself [1, 21, 24], while others try to correct in a post-processing strategy via adjusting the predictions of the model with a bias factor [2, 32]. Another typical approach is to balance the popular and unpopular items in the exposure data by assigning weights that are inversely proportional to item popularity in the loss function [21].

3 Methodology

3.1 Problem Formulation

We denote the labeled samples as $\{x_i, y_i, r_i\}_{i=1}^n$ where x_i indicates the user side information, y_i indicates the item side information and r_i indicates the label for ith example pair (x_i, y_i).

The retrieval model is typically modeled as an extreme multi-class classifier. The goal of the model is to predict the probability of user x_i engaging with item y_i

$$P(y_i|x_i;\theta) = \frac{e^{s(x_i,y_i|\theta)}}{\sum_{j\in\mathcal{I}} e^{s(x_i,y_j|\theta)}}$$

where $\mathcal{I}$ is the eligible item set, $s(x_i, y_i|\theta)$ is some function based on x_i and y_i and θ is the model parameter. For example, in the basic two-tower model set-up, $s(x_i, y_i) = v_i^T u_i$ where v_i and u_i are the embedding output from the user and item tower respectively. In the industrial applications, the cardinality of the item set $\mathcal{I}$ is at least on the order of millions.

Considering the widely adopted cross-entropy loss, the loss function to optimize is

$$\mathcal{L}(\{x_i, y_i, r_i\}_{i=1}^n) = -\frac{1}{n}\sum_{i=1}^{n} r_i \cdot \log(P(y_i|x_i;\theta))$$

3.2 Self-Supervised Cluster-based Negative Sampling

To enhance the training efficiency, the proposed method relies on a self-supervised cluster-based negative sampling technique to sample hard negatives from the item pool $\mathcal{I}$. First, clustering is performed on the item pool $\mathcal{I}$ and each y_j is assigned a cluster id. The clustering can be based on various dimensions, e.g. item category, item topics etc. Then for each example (x_i, y_i) in the labeled samples, $y_{i1}^{-n}, y_{i2}^{-n}, \ldots, y_{iK}^{-n}$ negatives are sampled from the same cluster as y_i. $\{y_{ik}^{-n}\}$ are hard negatives for the (x_i, y_i) because they are similar to y_i in some way.

For each example the cross-entropy loss is minimized for the true label and the sampled negative classes, i.e.

$$\mathcal{L}(\{x_i, y_i, r_i\}, \{x_i, y_{ik}^{-n}, 0\}_{k=1}^K) = -r_i \cdot \log\left(\frac{e^{s(x_i,y_i|\theta)}}{\sum_{j\in\{y_i\}\cup\{y_{ik}^{-n}\}_{k=1}^K} e^{s(x_i,y_j|\theta)}}\right)$$

3.3 A Gradient Perspective on Cluster-based Negatives

The efficacy of cluster-based negative sampling can be understood intuitively through the lens of contrastive learning and gradient analysis. In a standard two-tower model trained with InfoNCE or softmax cross-entropy loss, the gradient contribution of a sampled negative y_{ik}^{-n} with respect to the user anchor x_i is proportional to its exponentiated similarity $\exp\big(s(x_i, y_{ik}^{-n} \mid \theta)\big)$. Uniformly sampled negatives typically reside far from the anchor in the embedding space, resulting in vanishingly small similarities and, consequently, near-zero gradients that provide minimal learning signal.

By sampling negatives from the same semantic cluster as the positive item y_i, the proposed method guarantees that y_{ik}^{-n} shares latent characteristics with y_i, ensuring a higher similarity score $s(x_i, y_{ik}^{-n} \mid \theta)$ prior to full convergence. This pushes the negative sample closer to the model's current decision boundary. Consequently, the gradient magnitude is significantly larger, forcing the model to learn fine-grained distinctions between intra-cluster items rather than relying on trivial inter-cluster differences. This approach aligns with the theoretical findings in Approximate Nearest Neighbor Negative Contrastive Estimation (ANCE) [25], which demonstrates that negatives drawn from the local neighborhood of the query yield higher gradient norms and better approximate the oracle importance sampling distribution. This analysis also predicts a failure mode we observe empirically: because in-cluster negatives carry disproportionately large gradients, sampling too many of them per positive over-weights the contrastive term and destabilizes training. Indeed, varying the number of in-cluster hard negatives per positive from 1 to 64, we find that a small number performs best while larger counts degrade performance.

3.4 LLM-based Cluster Generation

In the cluster-based negative sampling process, selecting the appropriate clusters is crucial. Traditionally, media genres or categories have been used as clustering options, as discussed in Section 5.1. However, for the industrial application described in this paper, item clusters are derived from an in-house multimodal content-embedding model that captures semantic relationships across text,

image, audio, and video, going beyond surface-level genre or category.

The content embedding model is constructed as a set of fine-tuned encoders built on top of a large language model (LLM), designed to produce semantically rich item representations across text, image, audio, and video modalities. Figure 1 illustrates the workflow, which includes the following steps:

- *Pre-training*: The pre-trained LLM is utilized to achieve robust general language understanding. This foundational step ensures that the model captures complex semantic relationships that are often missed by traditional clustering methods based solely on surface-level features like genre or category.
- *Multimodal Encoder*: This module processes diverse input types, including text, images, audio, and videos, using a transformer-based architecture to encode them into fixed-size vector representations. This multimodal capability allows for a more nuanced and comprehensive understanding of media content, surpassing the limitations of traditional clustering that typically relies on single-modal data.
- *Fine-tuning*: The model is then fine-tuned to adapt to specific tasks, ensuring that the clusters generated are highly relevant and contextually informed. This fine-tuning process allows the model to capture subtle distinctions and patterns in user interests that traditional clustering methods might overlook.

Given these embeddings, item clusters are formed by applying k-means to the content-embedding space; each item is assigned to its nearest centroid, and newly encoded items are assigned online as they arrive. Because items are grouped by semantic proximity in this LLM-derived representation—rather than by surface genre or category—intra-cluster items tend to be genuinely similar across modalities, which yields harder and more informative negatives for cluster-based sampling.

The clustering granularity, controlled by the number of clusters k, plays a crucial role. Overly fine-grained clusters increase the risk of false negatives—items so similar to the positive that they are effectively relevant—whereas overly coarse clusters dilute the hardness of the sampled negatives. We therefore adopt a relatively coarse granularity in the production deployment.

The risk of sampling false negatives from within a cluster is small: a user is genuinely interested in at most thousands of items out of billions, so within a single cluster of order 10^4 items the fraction truly relevant to a given user—and thus false negatives if sampled—is well below 1%, and uniform random sampling further disperses these rare collisions across training steps, making their effect on the gradient negligible.

4 A Scalable Real-Time Sampling System

To deploy this technique at scale, we realize it in a real-time, end-to-end framework that performs cluster-based negative sampling in production, integrating into two-tower training and serving while handling billions of training examples with minimal computational overhead. The framework performs global out-of-batch sampling (GOOBS): as illustrated in Figure 2, it maintains an item pool that stores tensors for OOB samples. The maximum number of items is predefined, and each item is assigned to a "slot," which is a set of tensors storing the item's features.

During training, in-batch samples are passed through an update engine that employs a customized hashing function to determine the slot where the item should be stored. In parallel, a sampling engine takes in-batch items' cluster IDs as input to sample corresponding OOB samples from the item pool. These OOB samples are then combined with in-batch samples for training.

To ensure a high sampling hit rate during the early stages of training, the OOB item pool is pre-loaded with item features from an item data table derived from previous training data. Although this data may be slightly delayed compared to the latest training data, in-batch training samples capture the latest available items and continuously update the OOB item pool to maintain its freshness.

The core functionality of the cluster-based negative sampling framework is embedded in the update engine and the sampling engine. As shown in Figure 3 and Algorithm 1, during the update process, the item ID and cluster ID are passed through a hashing function to determine the dedicated cluster segment in the item pool. Each cluster segment consists of multiple slots to store item features, with each segment having an equal number of slots. When multiple items hash to the same slot, the later item overwrites the earlier one by design; this keeps the pool continuously refreshed with recent items and bounds its memory footprint.

Algorithm 1 Update engine

Input: Item IDs in ith training batch $x_i = [x_{i1}, x_{i2}, \ldots, x_{iB}]$, size B, slots per cluster segment S
for $j = 1$ **to** B **do**
1. Get cluster id of x_{ij}: c_{ij}
2. Hash x_{ij} to index $c_{ij} \cdot S + (x_{ij} \bmod S)$ in the item pool
end for

During sampling, as illustrated in Figure 4 and Algorithm 2, the in-batch samples' cluster IDs are used to identify the target cluster segment to sample from. Within the targeted cluster segment, an existing item and its features are randomly selected to be used as OOB samples.

Algorithm 2 Sample engine

Input: Item IDs in ith training batch $x_i = [x_{i1}, x_{i2}, \ldots, x_{iB}]$, size B, slots per cluster segment S
for $j = 1$ **to** B **do**
1. Get cluster ids of the in-batch samples: c_{ij}
2. Random sample $r_{ij} \in \text{rand}(0, S)$
3. Sample item N_{ij} in index $c_{ij} \cdot S + r_{ij}$ from the item pool
end for

5 Experiments

5.1 Public Datasets

5.1.1 Datasets. The performance of the cluster based negative sampling is first evaluated on the public datasets MovieLens-1M and Amazon Reviews (subsets Grocery, Electronics, Home) with full-shuffle similar to previous work on recommendations [29, 30].

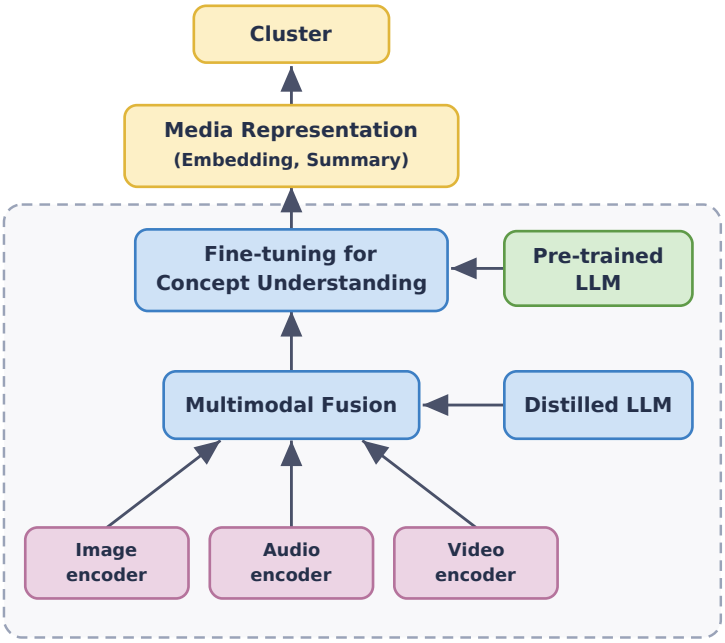

Figure 1: LLM-based cluster generation

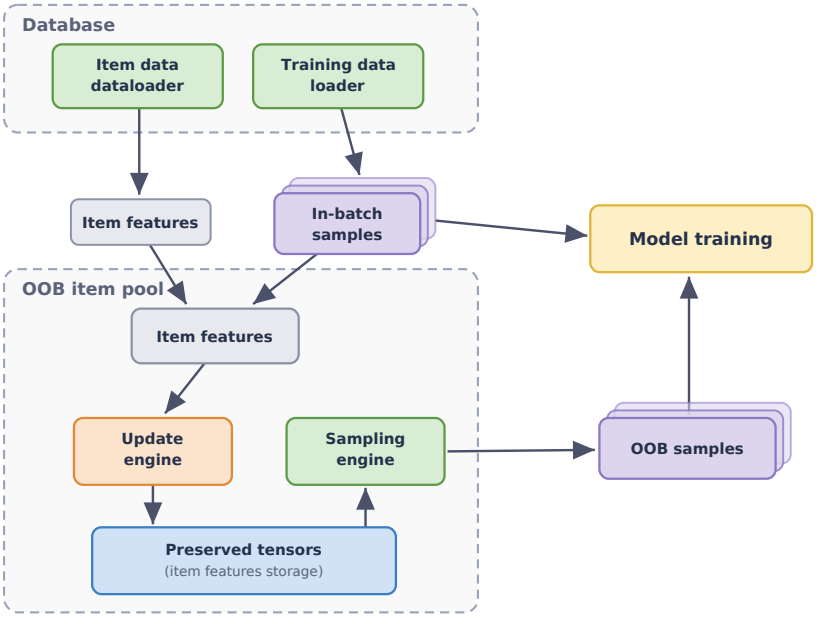

Figure 2: The real-time cluster-based negative sampling system

For pre-processing, timestamp-based splitting is used, ordering data by timestamps and taking the first 80% for training and 20% for evaluation. The rating label is converted to a binary label where rating 1–2 are negatives and 3–5 are positives. Features include user id, item id, and cluster id (e.g. genre id, category id) directly available from the datasets. Each user's historical interaction sequence

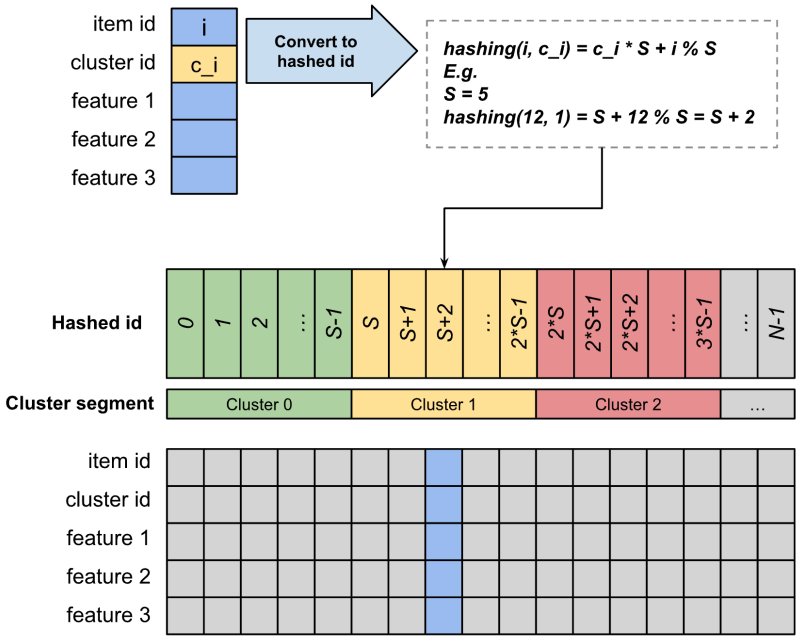

Figure 3: Cluster-based pool update

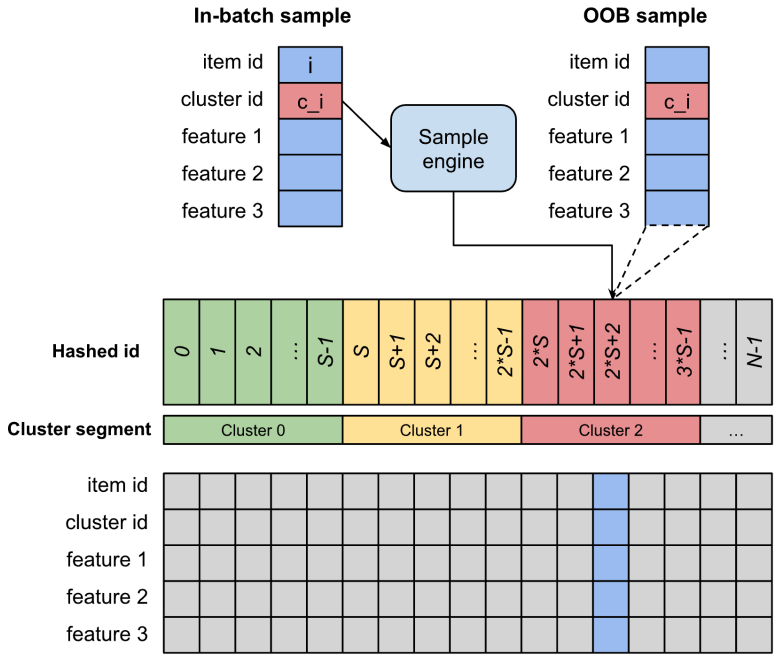

Figure 4: Cluster-based sampling

is extracted as user features. On the public datasets we deliberately use the readily available category and genre labels as the cluster assignment, and reserve the LLM-based clustering of Section 3.4 for the production deployment (Section 5.2). The public corpora are small—MovieLens-1M and the Amazon Reviews subsets contain on the order of 10^4–10^5 items, three to four orders of magnitude fewer than the production corpus, with only a short title and a few sparse attributes per item—so an LLM-derived partition of that space largely recovers the category taxonomy the datasets already ship with, at additional cost and with no additional information.

Table 1: Comparison of negative sampling methods on public datasets. HR@50 and HR@100 are reported. Bold indicates best performance. Relative improvements over Baseline are shown in parentheses.

Dataset	Metric	Baseline (In-batch)	DNS	CBNS	ANCE	GOOBS	Cluster GOOBS
Movielens-1M	hr@50	.2253	.2298 (+2.0%)	.2331 (+3.5%)	.2380 (+5.6%)	.2346 (+4.2%)	**.2415 (+7.2%)**
	hr@100	.3588	.3618 (+0.8%)	.3608 (+0.6%)	.3661 (+2.0%)	.3611 (+0.7%)	**.3682 (+2.7%)**
Amazon-Grocery	hr@50	.0254	.0261 (+2.8%)	.0271 (+6.7%)	.0288 (+13.4%)	.0279 (+10.1%)	**.0301 (+18.5%)**
	hr@100	.0406	.0419 (+3.2%)	.0432 (+6.4%)	.0451 (+11.1%)	.0440 (+8.3%)	**.0470 (+15.7%)**
Amazon-Electronics	hr@50	.0084	.0090 (+7.1%)	.0099 (+17.9%)	.0108 (+28.6%)	.0110 (+30.9%)	**.0131 (+55.6%)**
	hr@100	.0154	.0163 (+5.8%)	.0176 (+14.3%)	.0186 (+20.8%)	.0190 (+23.5%)	**.0201 (+30.2%)**
Amazon-Home	hr@50	.0050	.0054 (+8.0%)	.0061 (+22.0%)	.0065 (+30.0%)	.0067 (+34.2%)	**.0074 (+47.3%)**
	hr@100	.0082	.0088 (+7.3%)	.0094 (+14.6%)	.0099 (+20.7%)	.0102 (+23.9%)	**.0118 (+42.3%)**

The public experiments are therefore intended to isolate the contribution of the cluster-based sampling mechanism itself, holding the clustering source fixed at a simple and fully reproducible signal; the LLM-based clustering is evaluated as part of the complete system in the production deployment, where no curated taxonomy is available. We make no claim that LLM-based clustering outperforms category labels on the public benchmarks.

It is important to note that the evaluation protocol intentionally avoids k-core filtering (e.g., removing users or items with fewer than 5 or 10 interactions) and evaluates the target item against the entire global corpus rather than a small sample of 100 random negatives. As demonstrated by Krichene and Rendle [16], sampled evaluation metrics are often inconsistent with exact global ranking and artificially inflate absolute Hit Rate numbers. By retaining the full sparsity of the original datasets and performing exact global ranking, this experimental setup more accurately reflects the difficulty of real-world cold-start deployment scenarios. Consequently, while the absolute HR@50 and HR@100 values reported here are lower than those in studies employing heavy k-core filtering and sampled metrics, the relative performance improvements between sampling strategies remain robust and unbiased.

5.1.2 Models and Negative Sampling Strategies. A standard two-tower model is used as the backbone with six negative sampling strategies implemented on top of it to enable a comprehensive comparison with the state of the art:

- **Baseline** (in-batch negative sampling w/ LogQ correction): the standard industry baseline using in-batch negatives with LogQ correction to reduce over-penalization of frequent items. A false-negative mask is applied to suppress logits of known positives appearing as in-batch negatives.
- **DNS** [31]: Dynamic Negative Sampling selects negatives with the highest predicted relevance scores from the current model checkpoint, i.e., items the model currently believes are most relevant but are actually negative. This is the canonical hard negative baseline in the recommendation literature.
- **CBNS** [23]: Cross-Batch Negative Sampling caches item embeddings from recent mini-batches and reuses them as negatives for the current batch. Like GOOBS, CBNS leverages out-of-batch items; however, it selects negatives based on recency of appearance rather than semantic cluster membership.
- **ANCE** [25]: Approximate Nearest Neighbor Negative Contrastive Estimation builds a global ANN index over the entire item corpus and asynchronously refreshes it during training. Negatives are selected as the approximate nearest neighbors of each query in the embedding space, making them the most geometrically hard negatives available globally.
- **GOOBS** (Global Out-of-Batch Sampling): extends the Baseline with random out-of-batch (OOB) samples drawn uniformly from a maintained item pool in real-time. A pre-training pool loading step ensures high OOB hit rates from the start of training.
- **Cluster GOOBS**: augments the random OOB samples with hard negatives drawn from the same semantic cluster as the positive item, using a ratio of cluster to random OOB samples of 1:15 for Movielens-1M and 1:31 for Amazon Reviews datasets.

5.1.3 Results. Table 1 reports HR@50 and HR@100 across all four datasets. Cluster GOOBS achieves the best performance on every dataset and metric, with gains over the in-batch baseline ranging from +7.2% (Movielens-1M HR@50) to +55.6% (Amazon-Electronics HR@50), confirming that cluster-based hard negatives provide a consistent and substantial training signal across diverse recommendation domains.

Several trends are worth highlighting. First, the benefit of out-of-batch sampling is already evident with GOOBS alone: random OOB samples yield +4.2%–+34.2% HR@50 gains over the in-batch baseline, demonstrating that exposure to a broader item pool during training improves generalization even without hard-negative selection. Second, replacing random OOB samples with cluster-based hard negatives (Cluster GOOBS) consistently outperforms all baselines, including the geometrically hard ANCE method, which requires a global ANN index refresh. This is particularly pronounced on the sparser Amazon datasets: on Amazon-Electronics, Cluster GOOBS achieves +55.6% HR@50 vs. +28.6% for ANCE, and on Amazon-Home, +47.3% HR@50 vs. +30.0% for ANCE. Third, the gains are more pronounced on HR@50 than HR@100 across all datasets, suggesting that cluster-based negatives particularly

sharpen the model's ability to rank the most relevant items at the top of the retrieval list — the regime most critical for downstream ranking stages.

Overall, these results validate that semantic cluster membership provides a more effective hard-negative signal than recency (CBNS), dynamic score-based selection (DNS), or approximate nearest-neighbor geometry (ANCE), while requiring no global index and remaining compatible with real-time serving.

5.2 Industry Datasets

To evaluate the effectiveness of Cluster GOOBS in a real-world application, it is deployed to the entry content-discovery surfaces of a large-scale social media application, serving live production traffic, and evaluated via online A/B testing.

Unlike public datasets, real-world datasets present greater complexity and challenges, often exhibiting popularity bias. The application serves a user base on the order of one hundred million monthly active users (MAU), with an eligible item corpus on the order of tens of millions of posts and models trained on billions of user–item interactions. Despite this scale, the top 100 items represent roughly 50% of the total impressions. This indicates that the existing recommendation system does not have sufficient exploration on long tail items, causing users to interact with a limited set of items and creating a strong system feedback loop.

5.2.1 Setup. For both control and test group, 3% of users are randomly selected respectively. In the control group, the users are served by a two-tower model architecture with GOOBS. In the test group, the users are served by the same two-tower model architecture with Cluster GOOBS. The experiment ran for 14 consecutive days on live production traffic. Full production details—including the complete model architecture, feature set, and training hyperparameters—are omitted due to proprietary constraints.

5.2.2 Cluster Selection. As discussed in Section 3.4, in the real-data application, an LLM-based content-understanding model is leveraged to learn media representations and derive clusters. We compared several in-house LLM-based content features and selected the one that yielded the best cluster distribution balance and semantic clarity; this component is modular and can be replaced by any open-source LLM-based content embedding. In an offline sweep of the cluster count from 100 to 10,000, we found that 250–500 clusters gave the best balance between negative hardness and false-negative risk; accordingly, the deployment uses in total 300 clusters, 98% of which have $\geq 10k$ items. Figure 5 summarizes the cluster size across different clusters.

5.2.3 Results. The models are optimizing user engagements such as clicks. Therefore, CTR (click-through-rate) is used to measure online performance. Our system comprises multiple retrieval sources. The model improvement targets one of the largest sources in production. As shown in Table 2, Cluster GOOBS improves the CTR of that source by 53% ($p < 0.01$). Because this gain lands on the highest-traffic source in the system, it also translates into a +6.5% improvement in overall system-level CTR. Meanwhile, it only introduces minor training QPS (Queries Per Second) regression of −1.4% (with no regression for inference QPS), demonstrating the strong scalability and efficiency of the algorithm and making it easily adoptable in production.

The contents distribution is further analyzed to measure the popularity debias from Cluster GOOBS. Items having ≥ 1K impressions in the past 1 day are used as targeted item cohorts. In the test group, the number of items in this ≥1K-impression cohort increased by around 50% as shown in Table 3. The impression contribution from the top 100 items dropped from 50% to 32%, indicating a significant improvement in popularity bias.

Table 2: Online CTR comparison between different sampling techniques with the same two-tower architecture, measured over the 14-day A/B test. The numbers here are relative improvements.

Model	Source CTR	Overall CTR	Training QPS
GOOBS (control)	0%	0%	0%
Cluster GOOBS (test)	+53%	+6.5%	−1.4%

Table 3: Popularity debias comparison between different sampling techniques with the same two-tower architecture. The numbers here are relative improvements.

Model	Imp. buckets ≥1k	Top 100 items' imp. contrib.
GOOBS (control)	0%	50%
Cluster GOOBS (test)	+50%	32%

6 Conclusions

In this paper, we presented a cluster-based hard negative sampling technique together with a practical, real-time framework that realizes it in production for two-tower retrieval. The technique draws hard negatives from semantic clusters derived from LLM-based content representations, addressing the limitations of traditional negative sampling methods, which often produce easy negatives that fail to sufficiently challenge the model.

The system utilizes an item pool to efficiently manage and update OOB samples, ensuring minimal computational complexity while handling billions of training data points. The integration of a customized hashing function within the update engine and a targeted sampling engine allows for precise and effective negative sampling, maintaining the freshness and relevance of the item pool throughout the training process.

Overall, the proposed system provides a robust and scalable solution for improving the performance of recommendation models, offering a seamless integration path into production environments. Future work may explore further optimizations and extensions, including weighted cluster-based negative sampling and user query based negative sampling.

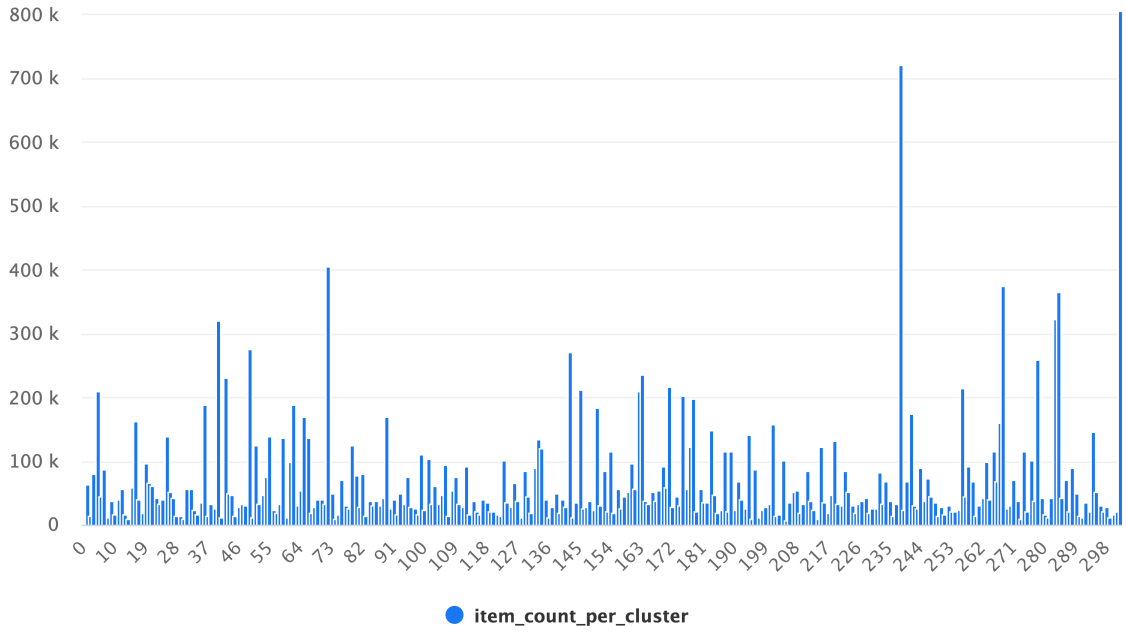

Figure 5: Cluster size distribution

Acknowledgments

We'd like to thank Pavitra Rengarajan, Guanfeng Liang, Claire Zhang, Kien Pham, Chenglin Lu, Kevin Zhao, Shoya Yoshida, Chris Smith, Srinath Sridhar, Tomer Bar, Mustafa Lokhandwala, Brion Spensieri, Hsiao-Ping Tseng, Qin Huang, Ziyi Zhao, Xinchen Hu, Bokai Cao, Miao Yu, Evie Feng, Ziyang Xie, Jinqiang Liu, Xianjie Chen, Wei Zheng, Aashu Singh, Michael Ge, Yuan Zeng, Sung Ha Hwang, Jianyu Wang, Shilin Ding, Xinyao Hu, Hong Yan for overall support on this project.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 42–46.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv preprint arXiv:1901.07555* (2019).

[3] Yoshua Bengio and Jean-Sébastien Senécal. 2003. Quick training of probabilistic neural nets by importance sampling. In *International Workshop on Artificial Intelligence and Statistics*. PMLR, 17–24.

[4] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19, 4 (2008), 713–722.

[5] Rocío Cañamares and Pablo Castells. 2018. Should I follow the crowd? A probabilistic analysis of the effectiveness of popularity in recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 415–424.

[6] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and debias in recommender system: a survey and future directions (2020). *arXiv preprint arXiv:2010.03240* (2020).

[7] Jin Chen, Defu Lian, Binbin Jin, Kai Zheng, and Enhong Chen. 2022. Learning recommenders for implicit feedback with importance resampling. In *Proceedings of the ACM Web Conference 2022*. 1997–2005.

[8] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J Shane Culpepper. 2017. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 445–454.

[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[10] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *Advances in Neural Information Processing Systems*, Vol. 33. 1094–1105.

[11] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506* (2019).

[12] B Hidasi. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).

[13] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[14] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[16] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757.

[17] Chi Liu, Jiangxia Cao, Rui Huang, Kai Zheng, Qiang Luo, Kun Gai, and Guorui Zhou. 2024. KuaiFormer: Transformer-Based Retrieval at Kuaishou. *arXiv preprint arXiv:2411.10057* (2024).

[18] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1557–1565.

[19] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. 2020. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 429–438.

[20] Harrie Oosterhuis. 2021. Computationally efficient optimization of plackett-luce ranking models for relevance and fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1023–1032.

[21] Harald Steck. 2011. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*. 125–132.

[22] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 515–524.

[23] Jinpeng Wang, Jieming Zhu, and Xiuqiang He. 2021. Cross-batch negative sampling for training two-tower recommenders. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 1632–1636.

[24] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1791–1800.

[25] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Estimation for Dense Text Retrieval. In *International Conference on Learning Representations*.

[26] Jing Yan, Liu Jiang, Jianfei Cui, Zhichen Zhao, Xingyan Bin, Feng Zhang, and Zuotao Liu. 2024. Trinity: Syncretizing Multi-/Long-Tail/Long-Term Interests All in One. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6095–6104.

[27] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020*. 441–447.

[28] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM conference on recommender systems*. 269–277.

[29] Jiaqi Zhai, Zhaojie Gong, Yueming Wang, Xiao Sun, Zheng Yan, Fu Li, and Xing Liu. 2023. Revisiting Neural Retrieval on Accelerators. *arXiv preprint arXiv:2306.04039* (2023).

[30] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. 2024. Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. *arXiv preprint arXiv:2402.17152v3* (2024).

[31] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 785–788.

[32] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-opportunity bias in collaborative filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 85–93.