SELF-SUPERVISED LLM-BASED HARD NEGATIVE SAMPLING FOR TWO-TOWER MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

The two-tower model has been widely used for large-scale recommendation systems, particularly in the retrieval stage. Industry standards for training two-tower models typically involve in-batch and/or out-of-batch negative sampling. However, these methods often produce easy negatives that models can quickly learn, failing to sufficiently challenge the model. To address this issue, we propose a novel self-supervised hard negative sampling technique that leverages a large language model (LLM) to generate hard negatives from the same cluster during model training. By utilizing the LLM to learn media representations, our approach ensures that the generated negatives are more challenging and informative. This real-time sampling framework is designed for seamless integration into production models, capable of handling billions of training data points with minimal computational complexity. Experiments on public datasets, along with deployment to a vast number of online users, demonstrate that our negative sampling technique outperforms widely used industry methods. Furthermore, our analysis in industrial applications reveals that this sampling method can help break inherent feedback loops in recommendations and significantly reduce popularity bias.

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 Covington et al. (2016); Liu et al. (2017); Chen et al. (2017). 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 Covington et al. (2016); Huang et al. (2020).

Retrieval model training is often formulated as an extreme classification problem where negative samples are playing a critical role. Various sampling-based techniques have being proposed to enhance training efficiency Bengio & Senécal (2008; 2003); Covington et al. (2016). The widely used sampling techniques are in-batch negative sampling Hidasi (2015); Gillick et al. (2019) and out-of-batch (OOB) negative sampling and also mixed negative sampling Yang et al. (2020); Hidasi & Karatzoglou (2018). However, in-batch negatives are constrained by the mini-bath size, which can result in models experiencing recommendation bias and limited exposure to diverse corpus during training. And OOB negative samples are too easy for model to learn especially when the OOB item pool is very large and diverse.

Besides, retrieval models typically use user engagement such as 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 our recommendation Chen et al. (2020); Cañamares & Castells (2018); Morik et al. (2020); Oosterhuis (2021).

To solve of the problems of negatives being too easy and also popularity bias, we propose a self-supervised hard negative sampling technique leveraging item cluster for the two-tower model and

design a real-time serving framework that can serve at the industry scale. This novel sampling design generates hard negatives on the fly during model training. On public data-sets, we demonstrate that our negative sampling technique yields significant performance improvements, particularly on large-scale data-sets such as Amazon Reviews, which feature a vast number of unique items. We also seen significant $+53\%$ CTR lift in industrial applications and also show that this technique can help with polarity bias mitigation.

The contributions of this paper are summarized below.

- We proposed a novel self-supervised hard negative sampling technique leveraging item cluster to introduce hard negatives for the two-tower model.
- We propose an efficient end-to-end model training/serving system to generate the negatives on the fly that can scale at industry level.
- We demonstrate that this negative sampling technique can significantly outperform the widely used in-batch or out-of-batch negative sampling on public data-sets and industrial data-sets
- We show that it can help with polarity debias in the industrial applications.

2 Related Work

2.1 Multi-stage System

For large scale recommendation system, we have millions of candidates eligible for selection. Given the latency constraint, we cannot 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 Covington et al. (2016); Liu et al. (2017); Chen et al. (2017).

2.2 Two-Tower Model

The Two-Tower model, a variant of the neural network architecture, has been as a powerful tool in recommendation systems since Huang et al. (2013) introduced it. It is widely used in the retrieval stage of industry applications Covington et al. (2016); Huang et al. (2020). 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 we do not need to do the online inference in realtime.

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 Bengio & Senécal (2008; 2003); Covington et al. (2016).

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 Hidasi (2015); Gillick et al. (2019). However, in-batch negatives are inherently limited by the size of mini-batches. This may lead to model suffer from recommendation bias and cannot expose models to a diverse candidates for training. To mitigate this, there are several variants being proposed. For example, Wang et al. (2021) 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 Yi et al. (2019) Yang et al. (2020), ByteDance Yan et al. (2024), Kuaishou Liu et al. (2024). 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, we need to retrieve relevant items from the whole corpus instead of just items from the mini match. 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 model to distinguish. To overcome this obstacle, many different hard negative sampling methods have been proposed in research, such as Dynamic Negative Sampling Zhang et al. (2013) and Adaptive Sampling Chen et al. (2022); Wang et al. (2017). But they are not widely adopted for large scale industrial applications due to the computational cost.

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 Yang et al. (2020); Hidasi & Karatzoglou (2018).

2.4 POPULARITY BIAS

The ideal state of recommendation system is help find most relevant items to users in a personalized way. However, in the real application, we often see popularity bias in the recommendation. Popularity bias means that the recommendation system tends to recommend rather popular items to user 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 Chen et al. (2020); Cañamares & Castells (2018); Morik et al. (2020); Oosterhuis (2021).

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's 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 Abdollahpouri et al. (2017); Steck (2011); Wei et al. (2021), while others try to correct in a post-processing strategy via adjusting the predictions of the model with a bias factor Abdollahpouri et al. (2019); Zhu et al. (2021) . Another typical approach is to balance the popular and unpopular items in the exposure data via by assigning weights that are inversely proportional to item popularity in the loss function Steck (2011).

3 METHODOLOGY

3.1 PROBLEM FORMULATION

Let's 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 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 millions scale.

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

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

3.2 SELF-SUPERVISED CLUSTER-BASED NEGATIVE SAMPLING

To enhance the training efficiency, we rely on a self-supervised clusted-based negative sampling technique to sample hard negatives from the item pool $\mathcal{I}$. First, we do clustering on the item pool $\mathcal{I}$ and assign each y_j with 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, we will sample $y_{i1}^{-n}, y_{i2}^{-n}, ..., y_{iK}^{-n}$ negatives 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.

$$\mathcal{L}(\{x_i, y_i, r_i\}, \{x_i, y_{ik}^{-n}, 0\}_{k=1}^K) = -r_i * log(\frac{e^{s(x_i,y_i|\theta)}}{\sum_{j\in\{y_i,\{y_{i1}^{-n}\}_{k=1}^K\}} e^{s(x_i,y_j|\theta)}})$$

3.3 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, in our real data application, we leverage an Interest Foundation Model to learn media representations and derive clusters from it, offering significant advantages over traditional clustering methods. The Interest Foundation Model is constructed as a set of fine-tuned content understanding models built on top of LLaMA (Large Language Model Meta AI), designed to enhance content understanding. Figure 1 illustrates the workflow, which includes the following steps:

- *Pre-training*: We utilize the pre-trained LLaMA to achieve a 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, 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.

By leveraging the advanced capabilities of LLaMA, our approach to clustering not only enhances the accuracy of media representation but also improves the quality of negative sampling. This results in more challenging and informative negatives, ultimately leading to better model performance. The LLM-based clustering method provides a more dynamic and flexible framework, capable of adapting to the evolving nature of user interests and media content, thereby outperforming traditional clustering techniques in both precision and scalability.

The granularity of the clusters plays a crucial role in performance. For effective negative sampling, it is important to select clusters that are not overly granular, as this helps minimize the risk of choosing false negatives.

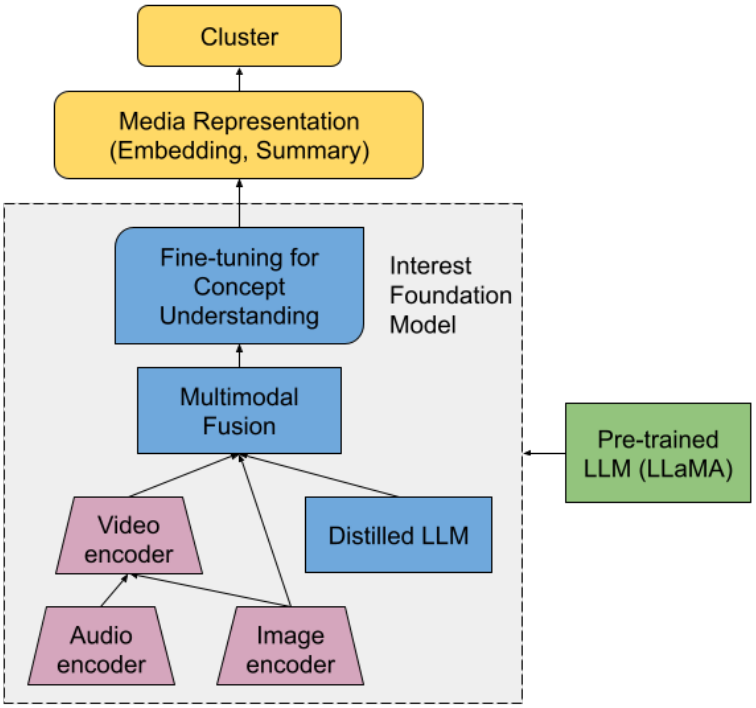

Figure 1: LLM-based cluster generation

4 Real-time sampling framework

We developed a real-time cluster-based out-of-batch (OOB) negative sampling framework, designed to be integrated into production models that handle billions of training data with minimal computational complexity. As illustrated in Figure 2, our approach utilizes 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.

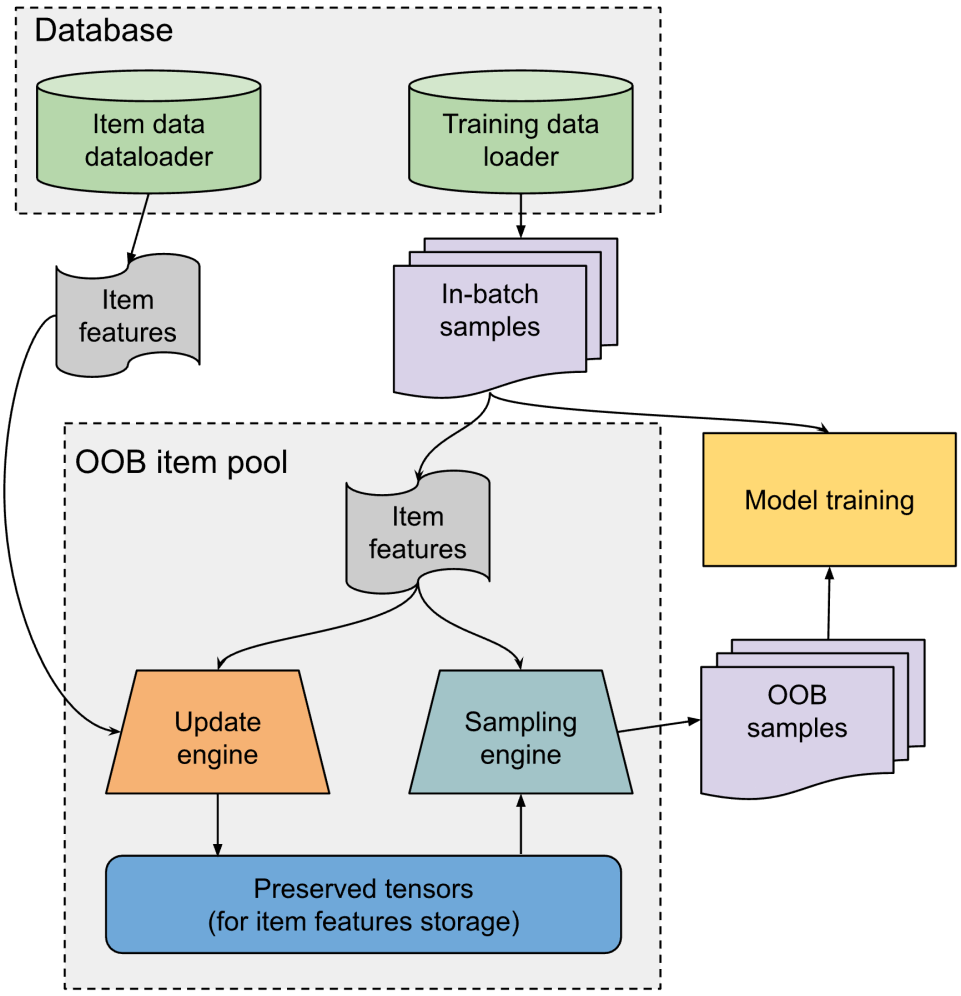

Figure 2: Real-time cluster-based OOB negative sampling framework

To ensure a high sampling hit rate during the early stages of training, we pre-load the OOB item pool 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 our cluster-based negative sampling framework is embedded in the update engine and the sampling engine. As showed 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.

Algorithm 1 Update engine

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

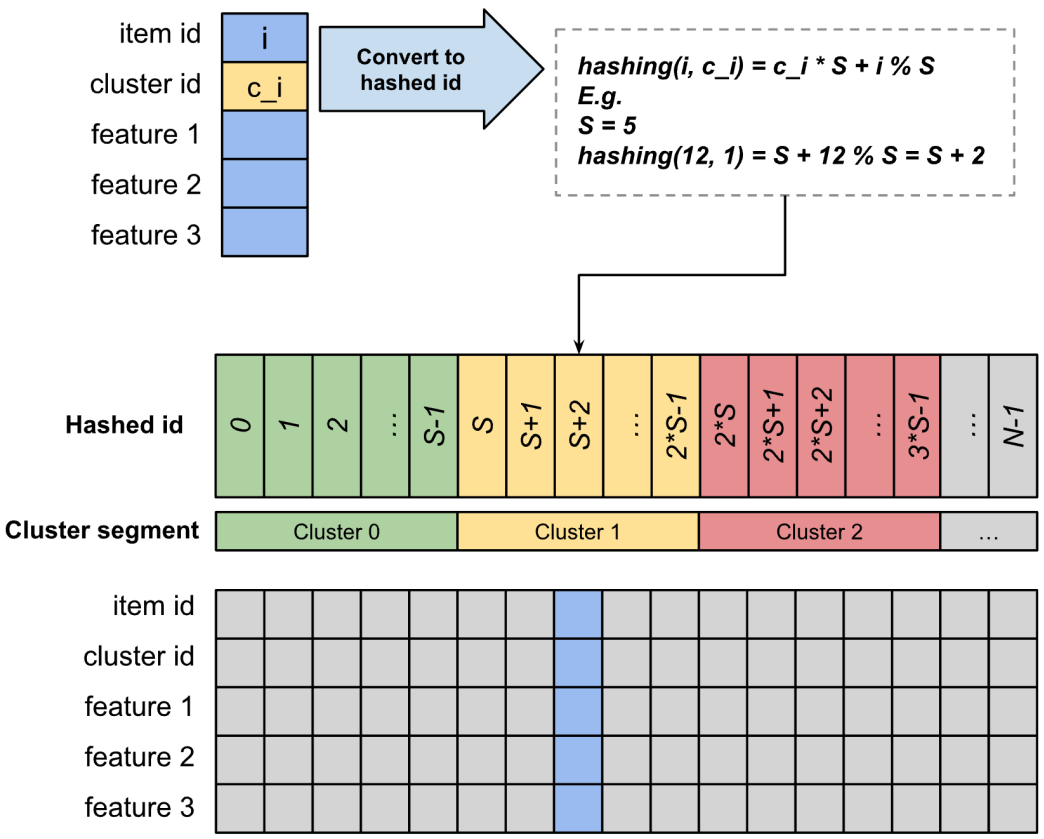

Figure 3: Cluster-based OOB pool update

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 $x_i = [x_{i1}, x_{i2}, ..., xi_B]$, size B, cluster size S
for $j = 1$ **to** B **do**
1. Get cluster ids of the in batch samples: c_{ij}
2. Random sample $r_{ij} \in$ rand $(0, S)$
3. Sample item N_{ij} in index $c_{ij} * S + r_{ij}$ from the item pool

end for

5 Experiments

5.1 Public Data-sets

5.1.1 Data-sets

We first evaluated the performance of the cluster based negative sampling on the public datasets MovieLens-1M and Amazon Reviews (subsets Grocery, Electronics, Home) with full-shuffle similar to previous work on recommendations Jiaqi Zhai & Liu (2023); Jiaqi Zhai (2024). For preprocessing, we used the timestamp-based splitting, where we ordered the data by the timestamps and took the first 80% of the data for training 20% for evaluation. We also converted the rating label to a binary label where rating 1-2 are negatives and 3-5 are positives. For features, we used user

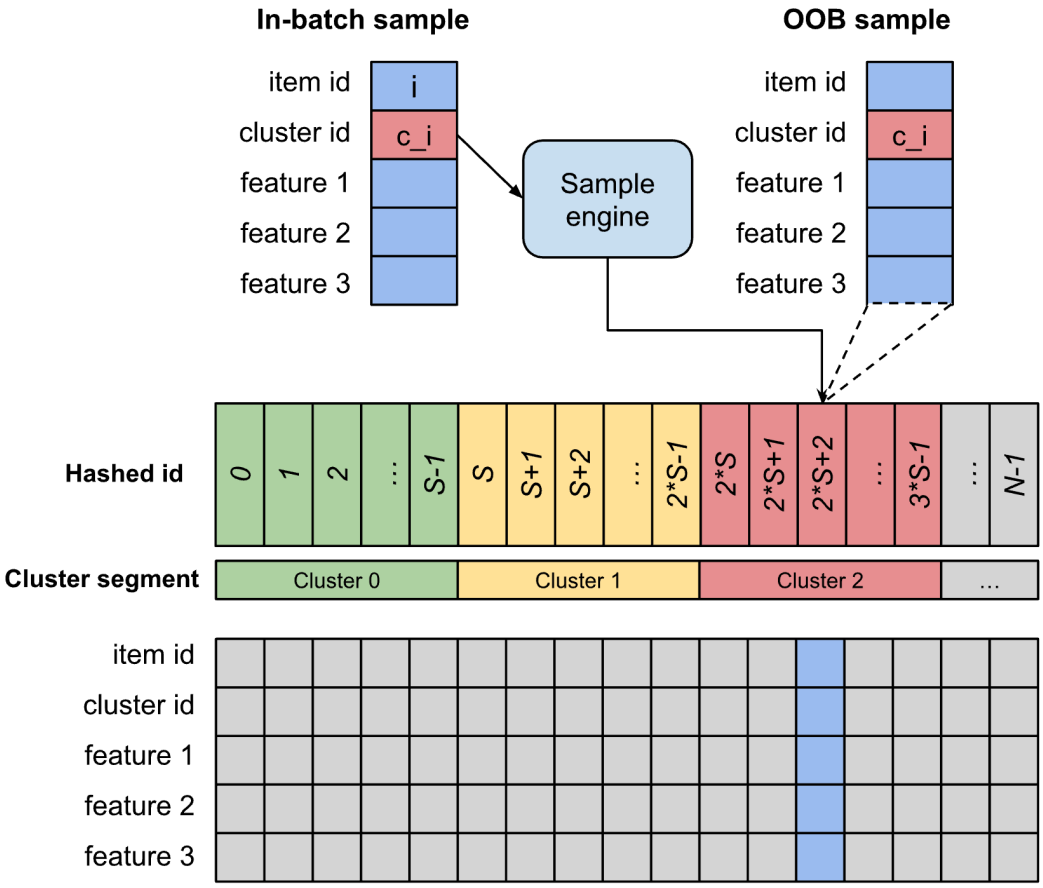

Figure 4: Cluster-based OOB sampling

Dataset	no. items	metrics	Baseline	Baseline w/ OOB	Baseline w/ Cluster OOB
Movielens-1M	3.7k	hr@50	.2253	.2346 (+4.2%)	**.2415 (+7.2%)**
		hr@100	.3588	.3611 (+0.7%)	**.3682 (+2.7%)**
Amazon-Grocery	266k	hr@50	.0254	.0279 (+10.1%)	**.0301 (+18.5%)**
		hr@100	.0406	.0440 (+8.3%)	**.0470 (+15.7%)**
Amazon-Electronics	700k	hr@50	.0084	.0110 (+30.9%)	**.0131 (+55.6%)**
		hr@100	.0154	.0190 (+23.5%)	**.0201 (+30.2%)**
Amazon-Home	1.19M	hr@50	.0050	.0067 (+34.2%)	**.0074 (+47.3%)**
		hr@100	.0082	.0102 (+23.9%)	**.0118 (+42.3%)**

Table 1: Evaluations of different sampling methods on public datasets

id, item id, cluster id (e.g. genre id, category id) that are directly available from the datasets. To enhance the features, we extracted each user's historical engaged item ids and cluster ids to better present user's interests. The historical engagement features is strictly extracted before the timestamp of each training sample to ensure no data leakage. We did not do any data filtering on the low presences users or items in the training data to make as close to the production as possible.

5.1.2 MODELS AND NEGATIVE SAMPLING STRATEGIES

We used a typical two-tower model as a baseline and implemented different negative sampling strategies on top of it:

- **Baseline** (in-batch negative sampling w/ LogQ correction): we used a baseline negative sampling approach that is well adopted in the industry: in-batch negative sampling with LogQ correction. LogQ correction reduced the over penalty of the frequent items by adjusting the logits during training by subtracting the log-probability of an item appearing in the training data. We also implemented the invalid sampled item mask to mask out the logit of false negative samples.
- **Baseline w/ OOB**: in addition to the baseline above, we added out-of-batch (OOB) samples that randomly sampled from the OOB item pool in the real-time. We also implemented the pre-training item pool loading to ensure the OOB sampling hit rate at the beginning of the training.
- **Baseline w/ Cluster OOB**: instead of all using random OOB samples, we added the cluster-based OOB negative samplings where those samples are come from the the same cluster of the user positive engaged item. There will be a ratio between the random OOB negative samples and cluster OOB negative samples. In this evaluation, we used 1:15 for Movielens-1M datasets and 1:31 for Amazon Reviews datasets.

5.1.3 RESULTS

We calculated the HitRate@50 metric across those datasets to evaluate the effectiveness of cluster-based out-of-batch (OOB) negative sampling. As shown in Table 1, our results indicate that cluster-based OOB negative sampling yields significant performance improvements, particularly on large-scale datasets such as Amazon Reviews, which feature a vast number of unique items. Notably, our findings suggest that Random OOB negative samples can improve model performance compared to the baseline (enhanced in-batch negative sampling). Moreover, incorporating cluster-based negative samples further enhances model performance, leading to even more substantial gains.

5.2 INDUSTRY DATA-SETS

To evaluate the effectiveness of cluster-based OOB negative sampling in a real-world application, we applied this to industrial recommendation systems that serve millions or billions of users and conducted online experiments in an A/B testing framework.

Unlike public datasets, real-world datasets present greater complexity and challenges, often exhibiting popularity bias. For instance, in this industrial dataset, we have around 18 million items eligible for impression, however the top 100 items represent 50% of the total impressions. This indicates that the existing recommendation system for this real-world application does not have enough exploration on the long tail items and make users interact with limited items, which will cause 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 arch with random OOB negative sampling. In the test group, the users are served by the same two-tower model arch with cluster-based OOB negative sampling.

5.2.2 CLUSTER SELECTION

As discussed in Section 3.3, in the real-data application, we leveraged the LLM based Interest Foundation Model to learn media representations and derive clusters from it. We have in total 300 clusters and 98% of them have $>= 10k$ items. Figure 5 summarizes the cluster size across different clusters.

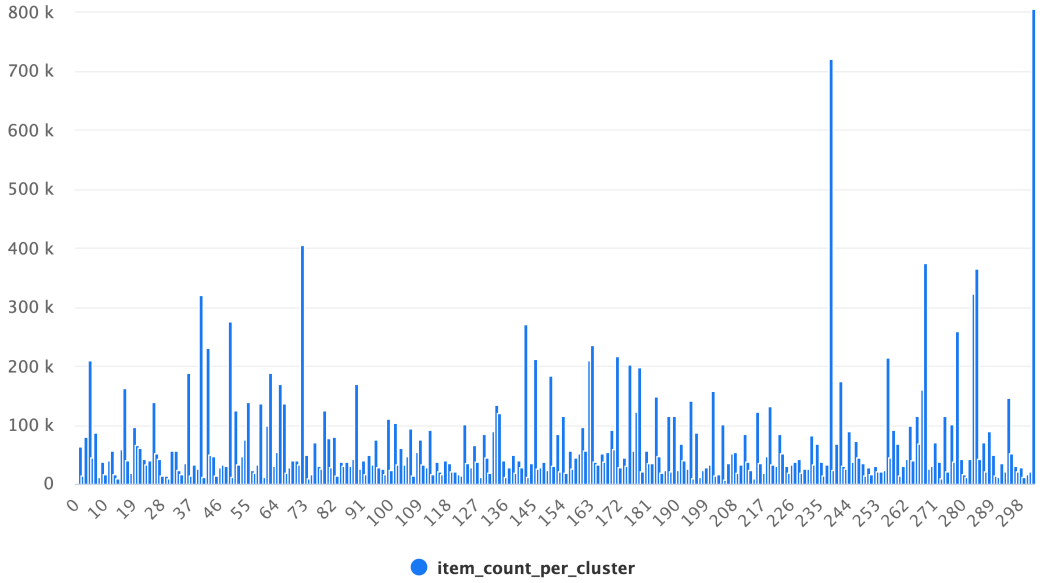

Figure 5: Cluster size distribution

5.2.3 RESULTS

The models are optimizing user engagements such as clicks. Therefore, we use CTR (click-through-rate) to measure online performance. As shown in Table 2, the cluster-based OOB negative sampling improves the CTR by 53%. Meanwhile, it only introduces minor training QPS (Queries Per Second) regression -1.4% (while no regression for inference QPS), demonstrating the strong scalability and efficiency of the algorithm and making it can be easily adopted in production.

We further analyze the contents distribution to measure the popularity debias from cluster-based oob sampling. We use items having $>=$ 1K impressions in past 1 days as targeted item cohorts. In the test group, we see that the percentage has increased by around 50% as shown in Table 3. Also the impression contribution from the top 100 items dropped from 50% to 32%, indicating we have significantly improved the popularity bias with this approach.

Table 2: Online CTR comparison between different sampling techniques with a similar model architecture. The numbers here are relative improvements.

Model	CTR	Training QPS
Random oob (control)	0%	0%
Cluster-based oob (test)	+53%	-1.4%

Table 3: Popularity debias comparison between different sampling techniques with a similar model architecture. The numbers here are relative improvements.

Model	imp buckets $>=$ 1k	Top 100 items' impression contribution
Random oob (control)	0%	50%
Cluster-based oob (test)	+50%	32%

6 CONCLUSIONS

In this paper, we have introduced a novel real-time cluster-based out-of-batch (OOB) negative sampling framework designed to enhance the training of two-tower models in large-scale recommendation systems. By leveraging a large language model (LLM) to generate hard negatives from the same cluster, our approach addresses the limitations of traditional negative sampling methods, which often produce easy negatives that fail to sufficiently challenge the model.

Our implementation 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, our framework 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 of this framework, including weighted cluster-based negative sampling and user query based negative sampling.

REFERENCES

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

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

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

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

Rocío Cañamares and Pablo Castells. 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*, pp. 415–424, 2018.

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

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

Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J Shane Culpepper. 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*, pp. 445–454, 2017.

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

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

B Hidasi. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

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

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

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

Xing Liu Yueming Wang Rui Li Xuan Cao Leon Gao Zhaojie Gong Fangda Gu Michael He Yinghai Lu Yu Shi Jiaqi Zhai, Lucy Liao. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. 2024.

Yueming Wang Xiao Sun Zheng Yan Fu Li Jiaqi Zhai, Zhaojie Gong and Xing Liu. Revisiting neural retrieval on accelerators. 2023.

Chi Liu, Jiangxia Cao, Rui Huang, Kai Zheng, Qiang Luo, Kun Gai, and Guorui Zhou. Kuaiformer: Transformer-based retrieval at kuaishou. *arXiv preprint arXiv:2411.10057*, 2024.

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

Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. 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*, pp. 429–438, 2020.

Harrie Oosterhuis. 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*, pp. 1023–1032, 2021.

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

Jinpeng Wang, Jieming Zhu, and Xiuqiang He. 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*, pp. 1632–1636, 2021.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 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*, pp. 515–524, 2017.

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

Jing Yan, Liu Jiang, Jianfei Cui, Zhichen Zhao, Xingyan Bin, Feng Zhang, and Zuotao Liu. 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*, pp. 6095–6104, 2024.

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

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

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 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*, pp. 785–788, 2013.

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