Trinity: Syncretizing Multi-/Long-Tail/Long-Term Interests All in One

Jing Yan
ByteDance Inc.
Shanghai, China
yanjing.rec@bytedance.com

Liu Jiang
ByteDance Inc.
Beijing, China
jiangliu.reco@bytedance.com

Jianfei Cui
ByteDance Inc.
Beijing, China
cuijianfei.ies@bytedance.com

Zhichen Zhao
ByteDance Inc.
Beijing, China
hanshujin@bytedance.com

Xingyan Bin
ByteDance Inc.
Beijing, China
binxingyan@bytedance.com

Feng Zhang
ByteDance Inc.
Shanghai, China
feng.zhang@bytedance.com

Zuotao Liu
ByteDance Inc.
Shanghai, China
michael.liu@bytedance.com

ABSTRACT

Interest modeling in recommender system has been a constant topic for improving user experience, and typical interest modeling tasks (e.g. multi-interest, long-tail interest and long-term interest) have been investigated in many existing works. However, most of them only consider one interest in isolation, while neglecting their interrelationships. In this paper, we argue that these tasks suffer from a common "interest amnesia" problem, and a solution exists to mitigate it simultaneously. We propose a novel and unified framework in the retrieval stage, "Trinity", to solve interest amnesia problem and improve multiple interest modeling tasks. We construct a real-time clustering system that enables us to project items into enumerable clusters, and calculate statistical interest histograms over these clusters. Based on these histograms, Trinity recognizes underdelivered themes and remains stable when facing emerging hot topics. Its derived retrievers have been deployed on the recommender system of Douyin, significantly improving user experience and retention. We believe that such practical experience can be well generalized to other scenarios.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; *Clustering*; *Nearest-neighbor search.*

KEYWORDS

interest modeling, interest amnesia, statistical method

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671651

ACM Reference Format:
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 (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3637528.3671651

1 INTRODUCTION

Large-scale industrial Recommender Systems (RS) confront numerous requests daily, and are supposed to capture each user's interest. With the growth of daily active users (DAU) as well as their time spending on the platform, interest modeling becomes more and more challenging and diversified. The emerging problems can be categorized as (1) multi-interest, which means users may be interested in many themes and topics (In Douyin, most users have more than 10 interested topics based on our proposed clustering system). (2) long-tail interest, referring to unpopular themes that only attract niche audience. (3) long-term interest, which mainly describes themes that users are willing to watch consistently. For example, one user watched many game and talk show videos about six months ago. In the last month, she was obsessed with dancing and fitness themes. Nowadays she watches some research progress, while she has been following economic news for years. Here all topics mentioned above can be defined as multi-interest. Games/talk shows/dancing/fitness are long-term interest topics that may occasionally be "forgotten" by our system, but still attract her if delivered. Research progress is a typical long-tail theme that is hardly appealing for everyone, but can be enjoyed by considerable users.

The key idea of this paper is that we believe the multi-/long-tail/long-term interests are mutually dependent and reinforcing. Therefore, we aim to improve these interest modeling tasks simultaneously. As illustrated in Fig.1, we have some insightful observations. First, a user's recent behavior sequence can be easily dominated by several hot topics, so we need long-term cues to identify the rest topics of multi-interest. Then, in a completed industry system, popular themes are already well-delivered and multi-interest

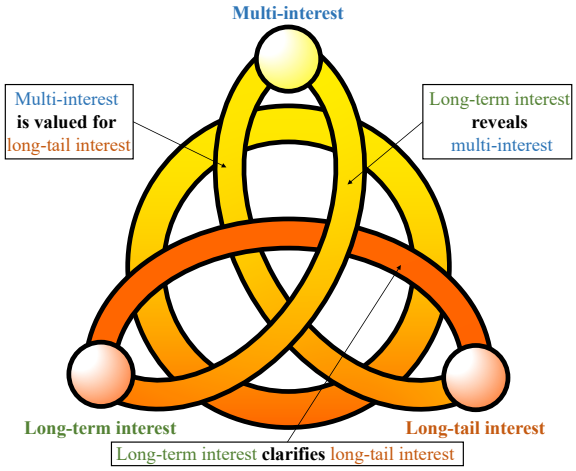

Figure 1: The interest reciprocity relationship and inspiration of Trinity. See text for detailed examples.

we are considering essentially refers to global long-tail themes. For example, in Douyin most users accept news and snapshots while it is more difficult to match calligraphy and its audience. Finally long-tail interest also relies on long-term cues since whether one is truly interested in a certain topic can only be confirmed by long observations. Inspired by these observations, we consider improving multiple interest modeling tasks simultaneously.

However, Modeling these interests is critical but quite hard: in current system we employ online learning models to rank candidates, which are trained by streaming paradigm and naturally tend to newcoming samples. So they may "forget" some topics of interest due to occasional misses of the corresponding training samples, leading to the "interest amnesia" problem, and worsening the delivery. Existing methods focus on modifying online learning framework, and introduce additional predictive heads and features that record users' past behavior. However, such methods pay excessive computational overheads and they only consider one of interest modeling tasks while neglecting their mutual relationship.

In this paper, we figure that the interest amnesia problem is essentially caused by online learning framework itself, and modifying it may be ineffective. On the contrary, we propose a statistic-based framework named "Trinity". Trinity comprehends long-term cues first and summarizes user interest into statistical histograms, by which we can easily distinguish multi-/long-tail/long-term topics. Then, the underdelivered topics are selected by customized strategies to improve user experience. Since Trinity is modeled based on long-term statistical information, it can represent multiple interests while withstanding interest amnesia problem. Trinity is composed of three individual retrievers: Trinity-M for multi-interest, Trinity-LT for long-tail interest and Trinity-L for long-term interest, as demonstrated in Sec.4. To our best knowledge, Trinity is the first work that syncretizes multiple interest modeling into one unified framework and innovatively introduces long sequence cues (length ≥ 1000) into retrieval stage.

Compared with existing methods which focus on one of interest modeling tasks, Trinity has some remarkable advantages: (1) Based on statistical histogram, each interested cluster is clearly evaluated and will not be forgotten or neglected. (2) Since items are assigned to indices exclusively, the computational overheads with interested topic growth are linearly increased, which outperforms existing methods. (3) Based on its clustering system, the whole delivery procedure is more explainable and manageable. We could work out more advanced strategies to reach better user experience. Trinity excellently benefits industrial recommender system, and we have launched the mentioned retrievers in Douyin and Douyin Lite.

2 RELATED WORK

In this section we summarize existing works into multi-interest, long-tail interest and long-term interest fields, and discuss how our work is different from them.

Recently, one of the popular methods considering multi-interest modeling is MIND [6]. It proposes multiple user representations (in this paper they are also denoted as "heads") to retrieve candidates from multiple fields, and has been the mainstream approach in industry applications. ComiRec [1] further considers diversity for improving system performance. MVKE [15] uses sub-expert-networks called "virtual kernels" to capture user preference on various tasks and topics, and combine them to output comprehensive results. Some works consider user interest from different aspects, e.g. Jiang [4] et al. explore multi-scale time effects on user interests. [11, 12] introduce similar ideas into sequence recommendation scenarios. Xie et al. [14] improve negative mining and routing algorithms to address training issues in multi-interest learning. Since these methods train multiple user representations to cover various user preference, we summarize them as "Multi-U" class methods. All the methods above focus on modifying existing online learning framework, while our proposed work relies on long-term statistics.

Long-tail items lack of user feedbacks or collaborative data so there are many methods focusing on multi-modal or graph methods [5, 7, 17] to improve the representations and prediction. Huang and Wu [3] extract product profiles and similar products in a topic model. Kumar et al. [10] realize long-tail recommendation using few shot learning, instead. Note in this paper, we are not focusing exactly on long-tail items. Long-tail interest mainly refers to niche topics but their items are unnecessary to be obscure.

In the area of long-term interest, Pi et al. [8] directly improve long sequence modeling by decoupling the most resource-consuming part from the entire model to a separate module. They update user representations by trigger events rather than requests and support thousands-length sequence. The most well-known method of capturing long-term interest is Search-based Interest Model (SIM [9]). It searches past topic-relevant behavior sequence to supplementarily measure user preference, but is mostly adopted in ranking models. Our approach avoids directing involving long sequence in models, which may be not well scalable. Instead, we exploit clusters and general interest distributions to mine long-term cues.

Our work employs VQ-VAE [13] in the training procedure to assign items to indices, and construct a hierarchical clustering system meanwhile. It allocates cluster embeddings and permits items to find their nearest neighbors. By that we can project untold items into enumerable clusters, and cope with user behavior sequence. Its varieties [2, 18] have been widely used in information retrieval.

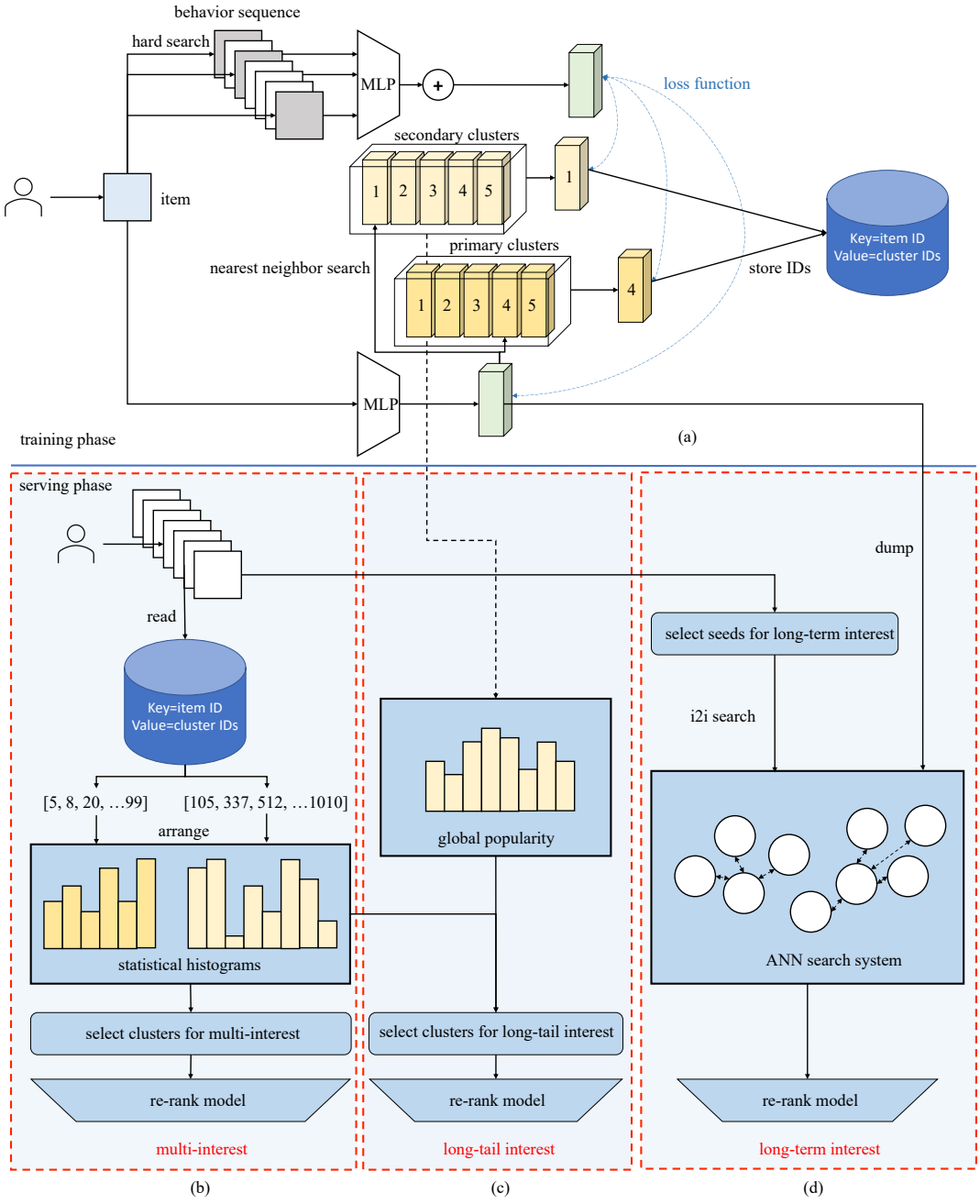

Figure 2: The framework of the proposed Trinity: (a) The training procedure. (b), (c) and (d) demonstrate serving methods for multi-/long-tail/long-term interests.

3 THE BASIC TRINITY FRAMEWORK

As illustrated in Fig.1, we consider multiple interests simultaneously. Different from existing work, we argue that modeling interests are not isolated tasks, on the contrary, these interests are mutually dependent and reinforcing. We have three perspectives:

1) Long-term interest reveals multi-interest. Users' short-term interest can be dominated by emerging trends and unpredictable hot topics. By involving longer behavior sequences, we can figure out which topics form multi-interest and obtain more comprehensive user preference.

2) Multi-interest is valued for long-tail interest. Multi-interest offers a wide collection of content themes, in which hot topics and niche themes can be both captured. However, items of hot topics (e.g. news, games) have been well estimated and delivered by

existing retrievals and ranking stages in a completed recommender system. When we are thinking of multi-interest, we are actually focusing on some uncommon themes, such as science and poems.

3) Long-term interest clarifies long-tail interest. Long-tail themes attract only a narrow audience, while many plays are just made by casual interest. To find out the real audience who consistently consume these themes we need to backtrack long-term behavior records.

For now, the interrelationships of interests are coherent, and inspire us to model them simultaneously. Online learning models are forced to fit current samples so they inevitably tend to new-coming samples and popular items (occupy the most impressions). If samples of some topics become infrequent, the corresponding interests will be gradually forgotten. We conclude this phenomenon as "interest amnesia problem", which needs enormous labour to be mitigated.

Instead of modifying the online learning framework, in this paper, we try to employ another instrument: **statistics**. Statistical cues extracted on long-term behavior can clearly reveal every interested topics, and remains stable when facing emerging trends. So it effectively mitigates interest amnesia problem and distinguishes multi-/long-tail/long-term interests. Specifically, we setup an interest clustering system. Then, users' past behaviors are projected into the system, resulting in long-term behavior histograms. Finally, we can implement customized delivery strategies from the histograms.

Considering the implementation, we need the following instruments: a clustering system which can be constructed by embeddings and evenly measures long-term and recent behaviors, a storage system by which we can map behaviors into clusters and thus form users' histograms in real time, and a series of delivery strategies.

In Fig.2(a) we illustrate the basic framework of Trinity. The training phase is mainly composed of a SIM (Search-based Interest Model [9]) head and a VQ-VAE [13] structure. For each item, we firstly search its SIM sequence (length=N_b), and extract embeddings for both (denoted as green cubes) as $\mathbf{x}$ and $[\mathbf{b}_1, \mathbf{b}_2, \dots \mathbf{b}_i]$, for $i \leq N_b$. Behavior sequence embeddings are mean-pooled to form user-side representation as $\mathbf{b} = \sum_i \mathbf{b}_i / N_b$. Meanwhile we prepare primary and secondary learnable clusters c_j^1, c_k^2 where j in $1 \dots J$, k in $1 \dots K$ (denoted as yellow cubes), and allocate their embeddings $\mathbf{e}_j^1$ and $\mathbf{e}_k^2$. J and K are set to be 128 and 1024. We search the top-1 nearest neighbor clusters for the current item (e.g. ID=4 and 1 in Fig.2) as $\mathbf{c}_{\hat{j}}^1$ and $\mathbf{c}_{\hat{k}}^2$ and apply Binary Cross Entropy (BCE) loss functions for $(\mathbf{b}, \mathbf{x})$, $(\mathbf{b}, \mathbf{e}_{\hat{j}}^1)$ and $(\mathbf{b}, \mathbf{e}_{\hat{k}}^2)$ pairs:

$$L = \sum_p \sum_{\mathbf{A}=\mathbf{x},\mathbf{e}_{\hat{j}}^1,\mathbf{e}_{\hat{k}}^2} y_p \log(\sigma(\mathbf{b}_p^T \mathbf{A}_p)) + (1 - y_p)\log(1 - \sigma(\mathbf{b}_p^T \mathbf{A}_p)), \tag{1}$$

where subscript p denotes sample index in a batch and σ refers to Sigmoid function. Here $y = 1$ if the user finishes the video, or interacts (upvote/share/follow/comment) with it, or watches it for more than 10 seconds. Moreover, we add some random negative samples to obtain more discriminative embeddings. Note the assignment relationship between items and clusters are determined by training progress in real time by nearest neighbor search, and we dump such information into a key-value storage (key=item ID and value=[primary cluster ID, secondary cluster ID]). Our implementation follows the same design in VQ-VAE where gradients for clusters are directly cast on items while cluster embeddings are updated by moving average of item embeddings belonging them. Here we use a hierarchical clustering system of two levels. The fine-grained secondary level is designed to select items while the coarse-grained primary level avoids repetitive topics and acts as dispersion (see Sec.4.1 for detailed example).

There are some details and considerations about clustering method we use: we can divide clustering methods into two-stage/streaming clustering. Two-stage clustering means employing clustering on off-the-shelf embeddings such as K-Means, while streaming clustering does not actually act exactly clustering. In streaming clustering methods (e.g. VQ-VAE) items are assigned to indices with the training procedure and it acts like clustering. Since we have billions of candidates, two-stage clustering is infeasible because of memory limitations. Besides, it has to be implemented offline, which may hurt recommendation performance. So we turn to streaming clustering, and hierarchical VQ-VAE is an appropriate choice. One alternative way is to replace SIM embeddings by multi-modal embeddings. However, it has two drawbacks: (1) There exists a gap between multi-modal embeddings and user interest because they are not collaborative; (2) They suffer from popularity bias, which, however, have been well solved by a learnable two-tower architecture [16]. In this paper we follow the two-tower architecture and implement debias method in [16], thus produce well-balanced indices. We find balanced indices are critical to improve retrieval model effectiveness and adopt Trinity strategies.

4 INTEREST MODELING IN LARGE-SCALE RECOMMENDER SYSTEM

In the section above, we have demonstrated the training procedure. In this section, we introduce specific and practical strategies to solve interest amnesia problem and improve multi-/long-tail/long-term interest modeling respectively. Note in this paper, we only focus on the retrieval stage. The reason is two-fold: on one hand, retrievers determine candidates for ranking models. If we lose topics of interest at this stage, ranking models can do nothing to improve interest modeling. On the other hand, retrievers always employ Approximate Neighbor Nearest (ANN) search system for computational efficiency. In accurate calculation scenarios where ranking models evaluate each candidate exhaustively, we can boost some topics by dispersion to "remember" them. However, we can hardly employ the same idea because approximate calculation discards some candidates at the very beginning.

4.1 Multi-interest

The key idea of improving multi-interest modeling is selecting clusters with moderate but significant intensity in user's statistical histograms, which have already been consumed by users but are easy to be forgotten by models. Given behavior counts on each cluster, well-delivered, underdelivered and uninterested topics are clearly revealed.

As illustrated in Fig.2(b), in the serving stage, for each request we first extract user's long-term behavior sequence (length=2500). Note the sequence only stores item IDs that $playtime \geq 10s$ or have

been finished or interacted (upvote/follow/share/comment). Then, the corresponding cluster IDs are read from the key-value storage mentioned in Sec.3 in real time, and are organized into primary and secondary histograms $\mathbf{h}^1 = [h_1^1, h_2^1, \dots h_J^1]$ and $\mathbf{h}^2 = [h_1^2, h_2^2, \dots h_K^1]$ where each bar refers to behavior count of the j/k-th cluster.

By simply sorting the histograms descendingly, user's interest distribution is clearly revealed. For example, if a sorted histogram is $[50, 20, 20, 4, 2, 0, 0, 0]$ and the corresponding indices are $[10, 33, 100, 91, 62, 21, 5, 83]$, cluster whose ID=10 is the user's major interest (we define "major interest" clusters which own the strongest response). However, there also exists significant response at index=33 and 100, which constitutes "multi-interest" we concentrate in this paper (major interests are also part of multi-interest, but they always have been well-delivered by existing retrievers). Clusters of 91 and 62 are treated as exploring interests that may become user's multi-interest or global long-tail interest. Other clusters are not interested.

Inspired by such observation, we introduce a strategy-based retriever to improve multi-interest modeling. First, we read user's past behavior cluster IDs: $(c_i^1, c_i^2)_{i=1}^L$ where $L \leq 2500$. In each pair, secondary cluster ID is attached as primary cluster's child node. It results in structural data like $c_j^1[h_j^1] \Rightarrow c_{jk}^2[h_{jk}^2]$ where "$\Rightarrow$" denotes one-to-many mapping relationship. Note in the training stage items individually search their nearest clusters so primary and secondary clusters are only statistically related. There may exist same secondary clusters under different primary clusters and we need dual subscripts in h_{jk}^2. In the following procedures, primary clusters are used as dispersion method, and we only leverage secondary clusters to deliver items.

First, we select primary clusters where h_j^1 is greater than $T_p = 30$ and each of whose secondary cluster has h_{jk}^2 greater than $T_s = 10$. Then we randomly choose one secondary cluster for each primary cluster. If there are not enough clusters, we select the secondary cluster from primary clusters above with the greatest h_{jk}^2. If there is still not enough clusters, we collect secondary clusters one-by-one from largest to smallest. In each step above, we remove duplicated clusters. The algorithm is summarized in Alg.1, which forms an additional retriever named "Trinity-M" by adding a re-rank model to evaluate the candidates of selected clusters.

Some middle clusters can be dominated by major interested topics since model bias (model believes most items of cluster A outperforms ones of cluster B), so in an unitary two-tower retriever, they cannot obtain enough impressions. However, in Trinity-M we only select one secondary cluster in each primary cluster. This acts like dispersion and we can collect complementary cluster set, which are not forgotten. By "awakening" these topics, users get part of their interest back, and feel more satisfied with our system. In Sec.5.3 we show the complementariness and comprehensiveness of Trinity-M, which leads to significantly improved user experience as well as more diversified system. Note we involve long sequence cues in Trinity framework by projecting behaviors into clusters, while avoiding straight employing in online learning models (retrievers have to manage billions of candidates, we can hardly use SIM [9] like ranking models). To our best knowledge, Trinity is the first work that successfully employs long sequence cues (≥ 1000) in the retrieval stage.

Algorithm 1: Trinity retriever for multi-interest

Data: Structural data $c_j^1[h_j^1] \Rightarrow c_{jk}^2[h_{jk}^2]$, threshold $T_p = 30$, $T_s = 10$, $N_M = 10$
Result: Secondary cluster set R_M
$R_M \leftarrow \emptyset$;
$p \leftarrow 0$;
foreach c_j^1 **do**
  if $h_j^1 \geq T_p$ *and* $\forall c_{jk}^2, h_{jk}^2 \geq T_s$ **then**
    randomly choose one secondary cluster $c_{j\hat{k}}^2$;
    $R_M \leftarrow R_M \cup \{c_{j\hat{k}}^2\}$;
  end
end
if $|R_M| \geq N_M$ **then**
  $R_M \leftarrow \{randomchoose(R_M, N_M)\}$;
  return R_M;
end
foreach c_j^1 **do**
  if $h_j^1 \geq T_p$ **then**
    $c_{j\hat{k}}^2 \leftarrow argmax(h_{jk}^2)$, where $c_{jk}^2 \notin R_M$;
    $R_M \leftarrow R_M \cup \{c_{j\hat{k}}^2\}$;
  end
end
if $|R_M| \geq N_M$ **then**
  $R_M \leftarrow \{randomchoose(R_M, N_M)\}$;
  return R_M;
end
sort all c^2 by h^2 from largest to smallest;
while $|R_M| < N_M$ **do**
  if $c_p^2 \notin R_M$ **then**
    $R_M \leftarrow R_M \cup \{c_p^2\}$;
    $p \leftarrow p + 1$;
  end
end
return R_M;

We compare Trinity-M with the "Multi-U" class methods in Table.1. Here we use "Multi-U" to summarize most existing works that train several user representations (heads) and search multiple candidates results for capturing multi-interest. The most well-known Multi-U methods are MIND [6], ComiRec [1] and MVKE [15].

Table 1: Comparison between Multi-U class methods and Trinity-M.

Aspect	Multi-U	Trinity-M
Efficiency	deteriorates with large head count	consistently keeps linear increasing
Explainability	semantically ambiguous	semantically meaningful
Extensibility	has not been validated	has been achieved in this paper
Fungibility	feasible	only complementary

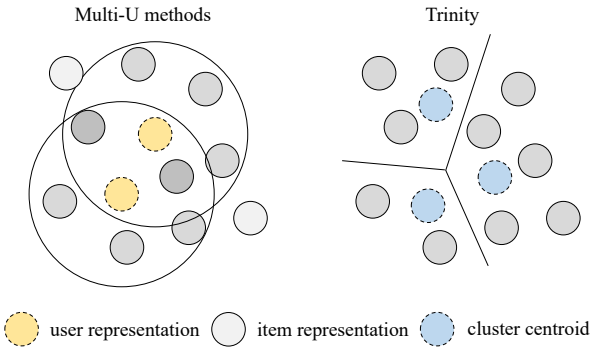

Figure 3: In Multi-U methods, each user representation (yellow dashed circle) searches candidates individually, so some items may be retrieved (solid hollow circles) more than once (deeper gray refers to more retrievals). However, in Trinity items are assigned exclusively, each item can be retrieved by at most once.

(1) **Efficiency**. The efficiency describes how serving cost increases with more considered topics. For Multi-U methods, when we introduce an additional head, we execute once more u2i search. However, as illustrated in Fig.3, items can be redundantly searched by multiple heads. So the more heads we have, the less unique candidates we can retrieve by adding heads. On the contrary, in Trinity-M items are assigned to clusters deterministically and exclusively. When we add additional clusters, they only bring supplementary candidates. Finding more interested clusters consistently costs linearly additional overheads.
(2) **Explainability**. The explainability describes semantic meaning of specific heads/clusters. In Multi-U methods heads are restrained to learn different samples, however, we can hardly control the training procedure or appoint specific topics to be optimized. In trinity-M, clusters themselves are semantically meaningful (see Sec.5.1), and we know what topics we are considering.
(3) **Extensibility**. The extensibility measures whether the method can be extended to other scenarios. Multi-U methods propose seldom solution for long-tail or long-term interest modeling tasks. While Trinity simultaneously improves these interest modeling problems, as demonstrated below.
(4) **Fungibility**. The fungibility refers to whether we can use the proposed method to replace existing retrievers. For this aspect, Multi-U methods are alternative because their multiple heads easily cover candidates of a single u2i model. However, since Trinity is modeled based on long-term statistic, emerging items/topics are hard to be prominent in short time. So we use Trinity-M as an additional retriever and only consider its supplementary contribution.

4.2 Long-tail Interest

For long-tail interest, our idea is to compare personal interest distribution against global interest distribution based on clusters. First, we need to automatically recognize/define global long-tail clusters. Then, we check if there exists significant response in user's statistical histogram on these clusters. Clusters meeting this criteria are collected to form an additional retriever.

Inspired by [16], we employ the similar streaming frequency estimation, but apply it on clusters rather than items. Specifically, cluster c_k (for long-tail interest we only consider secondary clusters) is mapped by a hash function $\mathcal{H}$, and when it occurs in the training streaming, its occurrence interval is calculated by

$$B[\mathcal{H}(c_k)] \leftarrow (1-\alpha) \cdot B[\mathcal{H}(c_k)] + \alpha \cdot (t - A[\mathcal{H}(c_k)]), \quad (2)$$

where $A[\mathcal{H}(c_k)]$ records the last occurrence timestamp of the cluster. $t - A[\mathcal{H}(c_k)]$ denotes the latest occurrence interval and $B[\mathcal{H}(c)]$ is its global moving average. This is illustrated in Fig.2(c).

In general, we can recognize global long-tail themes from clusters of large occurrence intervals. However, large occurrence interval clusters are not only composed by long-tail themes. There are also some "juxtapose" clusters which represent hot topics, but seize few items because of training occasionality. We remove clusters that have less than $T_i = 3$ items and effectively avoid juxtapose clusters. In our implementation, we define top 600 clusters (cover about 22% impressions) ranked by occurrence intervals as long-tail clusters without juxtapose clusters.

After defining long-tail clusters, we check user's statistical histogram. If a cluster has more than $T_l = 3$ behavior count, it is regarded as "convinced" response on long-tail interest, and is involved. We introduce a sampler to select $N_{LT} = 20$ clusters where each cluster has probability of

$$Pr(c_k) = \frac{(\beta + h_k)^\alpha}{\sum_{q=1}^{K} (\beta + h_q)^\alpha} \quad (3)$$

to be sampled. The hyper parameter α adjusts the distribution (for example, $\alpha = 0$ forms uniform distribution) and we use β as a smooth term. In our implementation we set $\alpha = 0.75$ and $\beta = 0.1$. The designed sampler outperforms a uniformly random sampler on user experience as well as diversity since items belonging to top clusters are more probable to be consumed (see Sec.5.2). Note duplicated clusters with Trinity-M have been removed at first.

The algorithm above is summarized in Alg.2. After collecting secondary clusters, we feed their items into a re-rank model to obtain retrieval results. This retriever is denoted as Trinity-LT.

4.3 Long-term Interest

As illustrated in Fig.2(d), we train a pre-rank model to select seeds from user's past behavior sequence. To capture comprehensive interest, any $playtime \geq 10s$, finish or interaction feedback is treated as positive label. The model follows conventional two-tower architecture, and involves random negative samples.

After ranking user's behavior items (length ≤ 2500), we project them into Trinity clustering system, and disperse by limiting that candidates of the same cluster can not exceed T_c. Then we randomly select N_L from top N_s ranked items as seeds, and search items with similarity measured by Trinity embeddings. Similar with Trinity-M

Algorithm 2: Trinity retriever for long-tail interest

Data: $c_k^2[h_k^2]$, global occurrence intervals $B[\mathcal{H}(c)]$, threshold $T_i = 3$, $T_l = 3$, $N_C = 600$, $N_{LT} = 20$
Result: Secondary cluster set R_{LT}
$R_{LT} \leftarrow \emptyset$;
$C \leftarrow \emptyset$;
foreach c_k **do**
  if c_k *has equal or more than* T_i *items* **then**
    $C \leftarrow C \cup \{c_k\}$;
  end
end
sort clusters $c_k \in C$ by $B[\mathcal{H}(c_k)]$, from largest to smallest;
choose top N_C clusters as C;
foreach c_k^2 **do**
  if $c_k^2 \in C$ *and* $h_k^2 \geq T_l$ *and* $c_k^2 \notin R_{LT}$ **then**
    $R_{LT} \leftarrow R_{LT} \cup \{c_k^2\}$;
  end
end
$R_{LT} \leftarrow \{sampler(R_{LT}, N_{LT})\}$;
return R_{LT};

and Trinity-LT, searched items are fed into the re-rank model, and form the retriever of Trinity-L. Note long-term cues have already been leveraged by Trinity-M and Trinity-LT, and we introduce Trinity-L as a lightweighted and supplementary extension.

4.4 Implement Details

The proposed three retrievers select clusters and aggregate their items, while remaining 10K-100K candidates, which is still too large for the following ranking stages. So we employ re-rank models to shrink candidates to about 1000. The re-rank models follow two-tower architecture, but precisely predict ranks without ANN in the serving phase.

The target of re-rank models is to predict video play time. Specifically, samples that are watched within 2 seconds are negative. For the rest positive samples, we use their play time (clipped to be at most 5 minutes) to weight the in-batch Softmax loss [16]. In fact such a model has been deployed as a major retriever called "stay-time retriever", and we just reuse it as re-rank models for additional retrievers.

5 EXPERIMENTS

In this section, we present the performance of Trinity. First, we show visualized analysis on its clustering system, demonstrate some interesting properties of the proposed model. Then, we show online performance of the additional retrievers on our large-scale industry scenarios. We also analyze them in detail to figure how they improve interest modeling tasks as well as user experience.

5.1 Clustering System of Trinity

In Fig.4 we visualize some items belonging to Trinity clusters. Items of the same row share the same secondary cluster ID, and we use (cluster ID, item ID) to refer them. The most important difference between Trinity clustering system and an intuitive clustering system from human perspective (denoted as label system), is its clustering manners. For example, in the first row, all items are about parent-child relationship rather than parenting, which is uncommon from human perspective. These items come from multiple tags such as news and laws, even the last example (A,c) talks about imaginary parent-child relationship that actually doesn't happen. The theme of a cluster focuses on an unconventional topic, and covers multiple tag aspects. The second row shows a much common taxonomy example where three items are all about education, and each of them explains knowledge of a sub-topic (career, economics and parenting). The most surprising example comes from the last row. The items show scenic spots and foods in three places, while these places are in the same province. So we can conclude another property of Trinity clustering: its taxonomy is varying with general topics. In the education topic, it categorizes items by subjects, while in the travel topic, it divides items into geographical bins.

We have tried multiple clustering embeddings to verify their capacity of describing user interests: multi-modal/online-learning model/SIM head (used in Trinity). Only SIM head based embeddings provide significant gains when used in Trinity-L (also see online experiments). The reason is that online-learning models are dominated by recent samples and cannot evenly measure interests long before. Multi-modal embeddings are not collaborative, and lack of discrimination on popular items.

5.2 Online Experiments

We conduct online A/B experiments and show the results of large-scale industrial recommender system on Douyin and Douyin Lite. As mentioned above, we simply launch three additional retrievers: Trinity-M, Trinity-LT and Trinity-L onto the whole system, and report the performance.

Metrics. The ultimate goal of recommendation algorithm is improving user experience, which is widely measured by Daily Active Users (DAU). However, DAU can hardly be observed in A/B experiments so we need alternative metrics. In this paper, we use Average Active Days (AAD) as a surrogate metric to describe user experience. When a user first arrives the platform, he/she is randomly grouped into control or experimental group. Then we count his/her active days during the experimental period, and calculate the average difference between the two groups. For most applications, DAU is the most critical metric. However, in Douyin, there are many users who log in almost everyday, so AAD can not describe their experience. To tackle this problem, we propose an additional metric Average Active Hours (AAH) which calculates users' active hours similarly with AAD. For multi-interest and long-tail interest modeling, diversity is expected to be improved. So we also involve Average Tags (AT) which counts how many tags are consumed over all items. In addition, we also employ Watch Time as an auxiliary metric.

In Table.2 we show the online performance, where only statistically significant metrics are listed. We have exhaustedly verified Multi-U methods, none of them obtains significant improvement within acceptable computational overheads. We report the results of MIND [6] here with 6 serving heads and omit others for concise. Such cost far exceeds our budget and the method is infeasible[1].

[1]Because of numerous candidates, one two-tower model needs about 20,000 CPU cores for one-pass u2i retrieval. So for 6-head-MIND we need 6x20,000 CPU cores to obtain

Figure 4: The visualization of items belonging to Trinity clusters. Items of the same row share the same secondary cluster ID.

Table 2: Results of online A/B experiments, measured by Watch Time, Average Active Days (AAD), Average Active Hours (AAH) and Average Tags (AT).

Retrievers	Douyin				Douyin Lite			
	Watch Time	AAD	AAH	AT	Watch Time	AAD	AAH	AT
MIND [6] (6 heads*)	+0.093%	+0.009%	+0.033%	-0.472%	+0.176%	-	+0.051%	-0.270%
Trinity-M	+0.118%	+0.008%	+0.046%	+0.153%	+0.178%	+0.018%	+0.078%	+0.038%
Trinity-LT	+0.069%	-	+0.019%	+0.546%	+0.060%	-	-	-
Trinity-LT (uniform sampling)	-	-	-	+0.154%	-	-	-	-
Trinity-L	+0.051%	+0.009%	+0.020%	-0.080%	+0.069%	+0.014%	+0.029%	-0.081%

One of unexpected observations is that MIND diminishes diversity, which implies all these heads repetitively retrieve some hot topics. On the contrary, Trinity-M significantly improve AAD and AAH on Douyin and Douyin Lite under limited overheads, and is successfully deployed. We provide an ablation study on Trinity-LT where uniform sampling means using a uniformly distributed sampler to select clusters. The group of uniform sampler has no positive impact on user experience and less diversity improvement, which verifies that clusters with larger response of a user are more likely to be consumed. Trinity-L also significantly improves AAD and AAH on Douyin and Douyin Lite. One interesting observation is that Trinity-L contributes much less AAH but reaches competitive AAD improvement compared with Trinity-M. This implies that multi-interest modeling mainly benefits high-active users (active more than 20 days in the past month). It consists with our expectation since high-active users are more likely to discover new interested topics.

5.3 The Comprehensiveness and Complementariness of Trinity-M

In Fig.5 we visualize topics that can be retrieved by stay-time retriever, MIND and Trinity-M for 2 users. Here we show top-ranked results since only these candidates can be fed into the following ranking stages. Note MIND and Trinity-M can retrieve their left topics, but we omit them for concise visualization.

For the first user, stay-time retriever only provides popular topics such as sports and games. MIND retrieves several additional topics like TV drama and travel. Here calligraphy is a pretty positive case since it is a long-tail topic. Trinity-M provides more topics (history, astronomy and agriculture) in addition. Trinity-M's retrieved clusters are not only more comprehensive, but also tend to long-tail topics.

significant AAD gains. On the contrary, Trinity selects clusters to shrink candidates, and feeds 1/1000 of them to re-rank model. Trinity-M only costs 4,000 CPU cores.

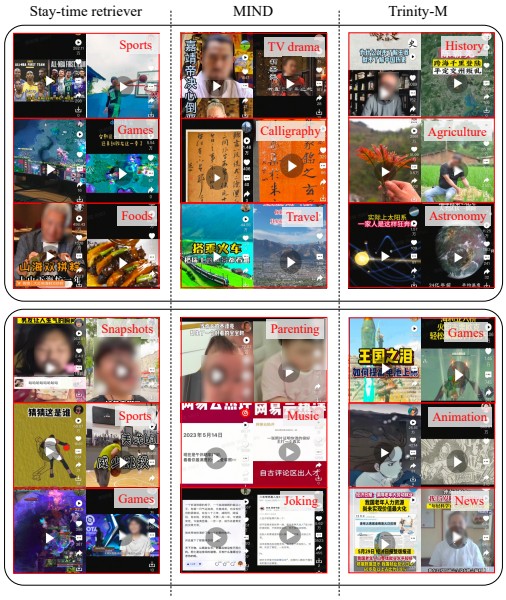

Figure 5: Top retrieved results of multiple retrievers. Two users are divided by boxes, and we only show supplementary topics of MIND and Trinity-M.

The example of the second user is much different, however. This user's interest mainly comes from popular topics such as games, parenting and news. Surprisingly, even hot topics can not be entirely retrieved by the unitary stay-time retriever. And Trinity-M provides complementary results of news and animation. Note the "Games" in the last column is different from the one in the first column, since it is a sub-category describing console games. In Douyin, most users have more than 10 interested secondary clusters measured by Trinity's clustering system, so only Trinity-M meets our demand of modeling multi-interest.

Note Trinity-M is insensitive to most of the hyper parameters. However, it is sensitive to the count of clusters we select. If we select too few clusters, we lose some interested topics. If we select too many clusters, some of them are just generated by casual behavior, and bring noise to the results. So the count of clusters we select is pretty important to obtain significant experience gain.

5.4 What contents benefit from Trinity-LT?

Our implementation of Trinity-LT uses taxonomy provided by Trinity itself, here we verify long-tail themes delivery based on label system instead, which mainly comes from human perspective. From our practical experience, when estimation of some items are improved, their long-term impressions will significantly increase, vice versa. So we can simply check impression distribution change caused by Trinity-LT to validate its influence on long-tail interest modeling.

Trinity-LT boosts some niche themes such as finance (+0.904%), laws (+0.478%), photography (+0.389%) and career (+0.230%), while deboosts popular topics like Movie (-0.520%), Games (-0.349%), Variety (-0.349%) and TV drama (-0.322%). Note the results are calculated across the whole market, in Trinity-LT niche/hot themes are boosted/deboosted even more significantly.

5.5 Seed Distribution of Trinity-L

Here we analyze the properties of Trinity-L. Since each item in user's 2500-length past behavior sequence can probably be selected as a seed, we expect that Trinity-L will retrieve earlier seeds. Compared with the existing i2i retriever whose seeds are selected by rules on recent behavior and employs conventional online learning model embeddings, Trinity-L has more seeds of 7-15 days ago (+67%) and 15-30 days ago (+28%). In the experiment Trinity-L delivers the earliest seeds for mid-active users (active more than 10 but less than 20 days in the past 30 days) and obtain significant AAD improvement on this group. It suggests that the refined long-term interest modeling leads to better user experience, and Trinity-L indeed captures long-term cues to improve user experience. Note "refined" here needs Trinity embeddings since without them we cannot obtain significant AAD improvement even on long-term seeds.

6 CONCLUSION

In this paper we focus on multi-/long-tail/long-term interest modeling at retrieval stage. We figure that online learning framework hardly solves the widely existing interest amnesia problem, and propose a unified and statistic-based framework to tackle the problem: Trinity. Based on a collaborative and time-variant clustering system, Trinity derives three individual retrievers (Trinity-M, Trinity-LT and Trinity-L) for the corresponding interest aspects. Trinity-M and Trinity-LT support customized strategies and retrieve targeted topics. While Trinity-L recovers some interested topics that have been forgotten. All of the retrievers have been deployed on Douyin and Douyin Lite, and have significantly improved user experience. We believe long-term statistic directly mitigates interest amnesia, and is tailored for large-scale industry systems.

REFERENCES

[1] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable Multi-Interest Framework for Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2942–2951. https://doi.org/10.1145/3394486.3403344

[2] Rong Huang, Danfeng Zhang, Weixue Lu, Han Li, Meng Wang, Daiting Shi, Jun Fan, Zhicong Cheng, Simiu Gu, and Dawei Yin. 2023. Learning Discrete Document Representations in Web Search. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (, Long Beach, CA, USA,) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 4185–4194. https://doi.org/10.1145/3580305.3599854

[3] Xin Huang and Feng Wu. 2019. A novel topic-based framework for recommending long tail products. *Computers & Industrial Engineering* 137 (2019), 106063. https://doi.org/10.1016/j.cie.2019.106063

[4] Hao Jiang, Wenjie Wang, Yinwei Wei, Zan Gao, Yinglong Wang, and Liqiang Nie. 2020. What Aspect Do You Like: Multi-Scale Time-Aware User Interest Modeling for Micro-Video Recommendation. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 3487–3495. https://doi.org/10.1145/3394171.3413653

[5] Joseph Johnson and Yiu-Kai Ng. 2017. Enhancing Long Tail Item Recommendations Using Tripartite Graphs and Markov Process. In *Proceedings of the*

International Conference on Web Intelligence (Leipzig, Germany) *(WI '17)*. Association for Computing Machinery, New York, NY, USA, 761–768. https://doi.org/10.1145/3106426.3106439

[6] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 2615–2623. https://doi.org/10.1145/3357384.3357814

[7] Andrew Luke, Joseph Johnson, and Yiu-Kai Ng. 2018. Recommending Long-Tail Items Using Extended Tripartite Graphs. In *2018 IEEE International Conference on Big Knowledge (ICBK)*. 123–130. https://doi.org/10.1109/ICBK.2018.00024

[8] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2671–2679. https://doi.org/10.1145/3292500.3330666

[9] Qi Pi, Xiaoqiang Zhu, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. *CoRR* abs/2006.05639 (2020). arXiv:2006.05639 https://arxiv.org/abs/2006.05639

[10] Yaman Singla, Dhruva Sahrawat, Shubham Maheshwari, Debanjan Mahata, Amanda Stent, Yifang Yin, Rajiv Shah, and Roger Zimmermann. 2020. Harnessing GANs for Zero-Shot Learning of New Classes in Visual Speech Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (04 2020), 2645–2652. https://doi.org/10.1609/aaai.v34i03.5649

[11] Qiaoyu Tan, Jianwei Zhang, Ninghao Liu, Xiao Huang, Hongxia Yang, Jingren Zhou, and Xia Hu. 2021. Dynamic Memory based Attention Network for Sequential Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (May 2021), 4384–4392. https://doi.org/10.1609/aaai.v35i5.16564

[12] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-Interest Network for Sequential Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (Virtual Event, Israel) *(WSDM '21)*. Association for Computing Machinery, New York, NY, USA, 598–606. https://doi.org/10.1145/3437963.3441811

[13] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. *CoRR* abs/1711.00937 (2017). arXiv:1711.00937 http://arxiv.org/abs/1711.00937

[14] Yueqi Xie, Jingqi Gao, Peilin Zhou, Qichen Ye, Yining Hua, Jae Boum Kim, Fangzhao Wu, and Sunghun Kim. 2023. Rethinking Multi-Interest Learning for Candidate Matching in Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 283–293. https://doi.org/10.1145/3604915.3608766

[15] Zhenhui Xu, Meng Zhao, Liqun Liu, Lei Xiao, Xiaopeng Zhang, and Bifeng Zhang. 2022. Mixture of Virtual-Kernel Experts for Multi-Objective User Profile Modeling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 4257–4267. https://doi.org/10.1145/3534678.3539062

[16] 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* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 269–277. https://doi.org/10.1145/3298689.3346996

[17] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the Long Tail Recommendation. *Proc. VLDB Endow.* 5, 9 (may 2012), 896–907. https://doi.org/10.14778/2311906.2311916

[18] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 1328–1336. https://doi.org/10.1145/3488560.3498443