CATS: Cluster-Aware Thompson Sampling for Negative Mining in Sequential Recommendation

Giulia Di Teodoro
University of Pisa, Italy
giulia.di.teodoro@ing.unipi.it

Federico Siciliano
National Institute of Geophysics and Volcanology, Italy
federico.siciliano@ingv.it

Nicola Tonellotto
University of Pisa, Italy
nicola.tonellotto@unipi.it

Fabrizio Silvestri
Sapienza University of Rome, Italy
fsilvestri@diag.uniroma1.it

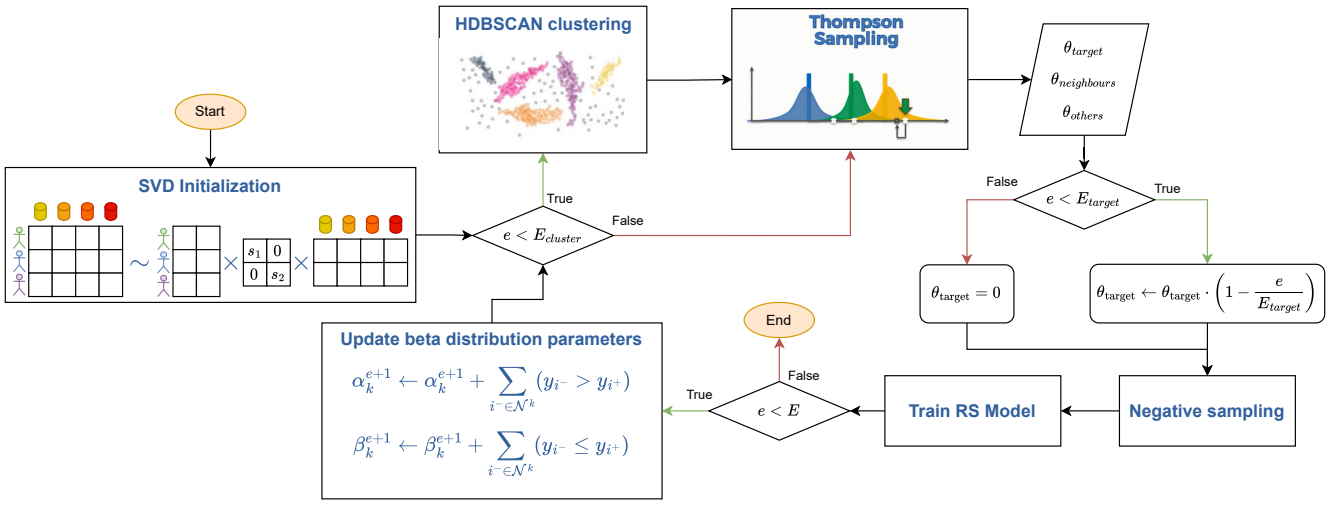

Figure 1: Overview of the CATS algorithm.

Abstract

Modern sequential recommendation systems often rely on negative sampling to efficiently train models over vast item corpora. However, common strategies such as uniform, popularity-based sampling, or hard-negative mining, often yield uninformative negatives, introduce popularity bias, or suffer from false negatives that hinder model learning. While several studies focus on identifying true negatives, none explore latent item representations to mitigate false negatives issue. We propose CATS (**C**luster-**A**ware **T**hompson **S**ampling),an adaptive negative sampling method that balances exploration and exploitation by leveraging unsupervised item clustering and adaptive multi-armed bandit principles. The key insight behind CATS is that false negatives tend to lie in the same cluster as the positive item, whereas true hard negatives are more likely found in nearest clusters. CATS leverages this by adaptively sampling negatives from the positive item's cluster, its neighboring clusters, and the rest of the item space, and decaying the sampling probability from the positive item's cluster over time, reducing the risk of accumulating false negatives during training. Evaluation using state-of-the-art sequential models (SASRec, BERT4Rec, BSARec) and three benchmark datasets (MovieLens-1M, Amazon Beauty and BeerAdvocate) demonstrates that CATS consistently outperforms standard and advanced sampling baselines. Notably, CATS achieves improvements of over 20% in NDCG@10 compared to vanilla sampling methods, and of over 16% compared to more advanced methods. Our findings suggest that incorporating the latent item structure through adaptive sampling strategies like CATS can significantly enhance the performance of sequential recommendation systems. The complete code needed to reproduce the results presented in this paper can be found in our GitHub repository https://github.com/gditeodoro/CATS.

This work is licensed under a Creative Commons Attribution 4.0 International License.

SIGIR '26, Melbourne, VIC, Australia

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3809539

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Negative Mining, Sequential Recommendation, Clustering

ACM Reference Format:
Giulia Di Teodoro, Federico Siciliano, Nicola Tonellotto, and Fabrizio Silvestri. 2026. CATS: Cluster-Aware Thompson Sampling for Negative Mining in Sequential Recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3805712.3809539

1 Introduction

Recommender systems (RSs) have become an indispensable tool for personalizing content in domains such as e-commerce, streaming platforms, and online advertising. Among them, Sequential RSs stand out for their ability to capture dynamic user preferences by considering the order of interactions over time. By learning patterns from behavioural sequences, these models can predict a user's next action more accurately than static preference models. Modern sequential recommenders, such as SASRec and BERT4Rec, have achieved state-of-the-art performance through the adoption of deep architectures originally designed for natural language processing, most notably Transformer-based models [2, 6, 18, 32, 35]. These models treat sequences of user-item interactions in a similar way to what happens in language models for sentence processing. This paradigm shift has enabled substantial performance improvements.

However, despite their effectiveness, these deep models face significant scalability challenges. Unlike language models with vocabularies in the order of tens or hundreds of thousands, RSs often handle millions of unique items. Computing scores across the entire item set during training becomes infeasible. For example, processing a batch of 128 sequences, each containing 200 items, and an item catalogue of one million items would require over 100 GB of memory just for storing logits (assuming 32-bit floating-point values). When accounting for gradients and model weights as well, the total memory usage can easily exceed the capacity of most *standard* GPUs.

Negative sampling and limitations. A widely adopted strategy to reduce computational costs is *negative sampling* [28], where training is limited to *positive items*, i.e., those the users have interacted with, and a small, possibily representative, subset of *negative items*, i.e., those that remain unobserved by the user. Uniform sampling is the simplest approach: negatives are drawn uniformly at random [29]. Though efficient, this approach often results in *easy negatives*, i.e., negatives that are very different from the positive item and provide weak or useless learning signals.

To address this issue, sampling of *hard negatives*, i.e., items that closely resemble positives, have been proposed. These sampling methods select negatives with higher predicted scores than positives [15, 43] or based on popularity [5]. However, these strategies introduce their own problems, like overlooking niche items, reducing catalogue balance, and harming coverage and fairness. Another limitation is the inclusion of *false negatives*, i.e., items the user would like but hasn't encountered yet. For instance, treating *Spider-Man 2* as a negative after a user has watched and liked *Spider-Man 1* introduces misleading signals into the model.

To reduce false negatives, recent works propose several targeted strategies: some filter items with low score variance during training [8] or re-weight samples by importance [3], others adaptively control sampling difficulty [20, 30, 31], while graph-based or semantic similarity methods select meaningful negatives using item transitions or cross-domain signals [9, 38]. However, none of these approaches considers item distribution in the latent space, or explicitly balances exploration and exploitation across item groups focusing on a single-domain scenario.

Motivations. In this work, we argue that the latent structure of the item space, as learned or approximated early during training, can guide more informative sampling. Our intuition is grounded in three key observations:

- Items that share semantic or behavioural similarities tend to lie close together in the embedding space. In light of this, items very close to the positive item in latent space are more likely to be relevant to the user, even if they have not been explicitly interacted with. For instance, a user who reads a science fiction novel is likely to be interested in other works by the same author or from the same series. Leveraging this proximity can help reduce the risk of sampling false negatives.
- Hard negatives reside still near the positive item in the latent space but not as close as potential false negatives. They represent plausible alternative choices, less preferred by the user, and therefore difficult to distinguish from positive items; exactly the type of hard negative that maximises the learning signal.
- The sampling strategy should evolve over time. Early training benefits from exploration: exposing the model to a diverse set of negatives, even easy (less informative) negatives, is essential for training the SR model. This helps to build a solid distinction between what is plausibly interesting and what is irrelevant. As training progresses, the focus should shift towards exploitation: mining more challenging negatives from promising areas, optimizing the quality of the learning signal.

Our Proposal. Based on our key observations, in this work we introduce CATS (***C**luster-**A**ware **T**hompson **S**ampling*), a novel negative sampling framework for sequential recommendation. It integrates unsupervised item clustering with Thompson sampling [1] to balance exploration and exploitation during training. Specifically: (1) item embeddings are initialized via truncated SVD [11] and HDBSCAN [24] is applied to form item clusters and detect outliers; (2) for each training instance, items are grouped into three categories: the target's cluster (the cluster of the positive sample), its neighbouring clusters, and all others; (3) beta priors are maintained over these groups and Thompson sampling is applied to adaptively sampling negative items across these groups based on their observed difficulty. Difficulty is measured by tracking hard negatives across epochs; (4) a decay factor applied to the probability of sampling from the target cluster gradually shifts the focus away from the target cluster in order to reduce the risk of sampling false negatives. Moreover, after a warm-up period, clustering is frozen and sampling continues on the basis of the learned groups.

Contributions. Our main contributions are:

- We propose CATS, the first negative sampling framework to integrate unsupervised item clustering with Thompson sampling in sequential recommendation.

- We conduct extensive experiments that demonstrate CATS improves ranking performance over competitive sampling baselines on three state-of-the-art sequential RS models (SASRec [18], BERT4Rec [33], BSARec [32]) and three benchmark datasets. We compare to standard baselines (uniform [29], popularity [5]) and recent advanced samplers (DNS [43], DNS+ [31], GNNO [9]) using the NDCG@{5,10,20,50} and the HR@{5,10,20,50}, demonstrating consistent improvements. Notably, CATS achieves improvements of over 20% in NDCG@10 compared to vanilla sampling methods, and of over 16% compared to more advanced methods.
- We provide an ablation study and detailed analysis that highlight the role of clustering dynamics, the importance of avoiding oversampling from the target cluster, and the effectiveness of adaptive sampling schedules.

The rest of the paper is organised as follows: Section 2 discusses related work. Section 3 introduces the notation and problem setting. Section 4 presents our proposed CATS framework in detail. Section 5 outlines the experimental setup. Section 6 reports and analyses the results. Section 7 concludes by setting out future directions.

2 Related Work

2.1 Sequential Recommendation

Sequential RSs aim to capture evolving user preferences by modeling item interaction sequences. Early methods like NCF [13] and Caser [34] introduced neural architectures for collaborative filtering. In parallel, models such as GRU4Rec [14] and Neural Attentive Session-based Reccomendation (NARM)[21] adopted recurrent neural networks to learn from sequential patterns. More recently, Transformer-based models [35] have become prominent due to their efficiency and strong performance. SASRec [18] applies unidirectional self-attention, while BERT4Rec [33] and Transformers4Rec [6] leverage bidirectional attention. BSARec [32] improves standard self-attention with frequency-sensitive filtering, integrating both low- and high-frequency signals to better capture user behavior and reduce oversmoothing.

As these deep architectures scale to catalogues with millions of items, computing scores for every candidate item at each training step becomes infeasible. This computational bottleneck makes negative sampling an essential technique to train sequential models efficiently.

2.2 Negative Sampling in Recommendation

A recent survey [41] categorizes negative sampling strategies as either static or dynamic. Classical static methods, such as uniform random sampling [29] and popularity-based sampling [5, 39], are simple but can lead to uninformative gradient updates or reinforce popularity bias. More recent works developed dynamic schemes that aim to choose hard negatives (e.g., highest scoring items among candidates) during model training. Among them, one of the most representative is Dynamic Negative Sampling (DNS) [43] that prioritizes negative items by ranking them and sampling more frequently from the top-ranked ones. Some approaches adopt Generative Adversarial Networks (GANs) to synthesize informative negative samples and enhance model robustness; notable examples include IRGAN [37] and AdvIR [27]. More recent works also exploit auxiliary information [4, 9, 16, 36, 38, 42] to guide the negative sampling process. For instance, MixGCF [16] introduces techniques like positive and hop mixing to craft harder negatives, while other studies [4, 36] leverage social network data for this purpose. Nevertheless, strategies that favor high-scoring items for sampling are prone to the false negative issue, as these items may actually be relevant. To address this, several methods [8, 9, 20, 31] have focused on reducing the impact of false negatives. SRNS [8], for example, uses sampling variance to detect potentially mislabeled negatives, assuming such items exhibit more stability during training. DNS+ [31] adjusts the sampling difficulty by broadening the candidate pool among top-ranked items. GNNO [9], on the other hand, samples from a global item transition graph and discards likely false negatives by assessing neighborhood-level similarity. EXHANS [38] leverages cross-domain user preferences to guide negative sampling and mitigate false negatives, but its design is specific to cross-domain settings and not suitable for single-domain scenarios like ours.

Nevertheless, these dynamic methods do not consider items similarity derived from their embedding distribution in the latent space. It is assumed that item embeddings are arranged in the latent space according to their semantic similarities, which can help guide the selection of negative samples.

3 Notations and problem setting

This section provides an overview of the fundamental principles that underpin our work, including the formulation of sequential recommendations and the distinction between positive and negative training examples.

RSs task. The goal of sequential recommendation is to predict a user's next interaction by modelling their past interactions in chronological order. Formally, each user $u \in U$ is associated with a time-ordered sequence of items $\mathcal{S}_u = [i_1, i_2, \ldots, i_{T_u}]$, where each $i_j \in \mathcal{I}$ represents an item, T_u is the total number of that user's interactions and $\mathcal{I} = \{1, \ldots, N\}$ is the complete set of items in a catalogue. A sequential RS f_θ learns a scoring function $y_j = f_\theta\left(i_j | i_1, \ldots, i_{t-1}\right)$, which gives a score y_j to the likelihood that a candidate item i_j follows the observed context $(i_1, \ldots, i_{t-1})$. In scenarios with implicit feedback, users' interactions (e.g. clicks, purchases and views) serve as positive signals. A positive example i^+ is therefore defined as the event $(i_t | i_{1:t-1}) \in \mathcal{S}_u$, i.e. the true next item in the user's sequence.

At training time, the model parameters θ are optimised to give high scores to the "correct" next items (i_t) and low scores to alternatives ($i_j, j \neq t$).

Negative Sampling. A negative item set for each user u is defined as: $\mathcal{N}_u = \mathcal{I} \setminus \mathcal{S}_u$, i.e. all items with which the user has not interacted. In practice, the universe of all items $\mathcal{I}$ is large, ranging from tens of thousands to millions, so it is infeasible to treat every unobserved item as negative at each training step. Sampling a small subset $\mathcal{N}_u^k \subseteq \mathcal{N}_u$ of size k reduces both memory and computing requirements: only k negative scores must be computed per positive score, rather than $|\mathcal{N}_u|$.

Training Objective. Given a sequence $(i_1, \ldots, i_{t-1})$, one positive item i_t (i^+) and k sampled negative items $\mathcal{N}_u^k$, a common training loss is the cross-entropy (CE) in the sampled form:

$$\mathcal{L}_{u,t} = -log\ \frac{\exp(y_t)}{\exp(y_t) + \sum_{i \in \mathcal{N}_u^k} \exp(y_i)} \tag{1}$$

where σ is the sigmoid function. This loss approaches the ranking task as a multi-class classification problem, modelling the probability distribution over the entire set of sampled items, and has proven to have benefits for learning to rank task [7, 40].

Efficient and effective negative sampling is therefore critical: (i) too few negatives (small k) can provide the model with insufficient contrasting examples; (ii) poorly chosen negatives (e.g. items that are far from the positives in latent space) result in weak gradients; (iii) aggressive hard-negative mining may introduce false negatives, which can harm learning.

4 Methodology

In this section we present a detailed description of 😼CATS😼 for negative sampling.

4.1 Item Embedding Initialization

To initialize the item representations, we apply *Truncated Singular Value Decomposition* (SVD) [11] to the user-item interaction matrix $\mathbf{R}$ obtained from the training set. Standard SVD factorizes $\mathbf{R}$ as $\mathbf{R} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{U \times U}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ contain the left and right singular vectors associated with users and items, respectively, and Σ is a diagonal matrix of singular values indicating the importance of each latent factor. However, in real-world RSs, $\mathbf{R}$ is typically sparse and high-dimensional, making full-rank SVD computationally impractical. Truncated SVD addresses this by retaining only the top d singular values and their corresponding vectors, yielding a low-rank approximation $\mathbf{R}_d \approx \mathbf{U}_d\Sigma_d\mathbf{V}_d^T$, where $\mathbf{U}_d \in \mathbb{R}^{U \times d}$, $\Sigma_d \in \mathbb{R}^{d \times d}$, and $\mathbf{V}_d \in \mathbb{R}^{N \times d}$. The rows of $\mathbf{V}_d$ provide dense item embeddings that reflect co-occurrence patterns in the interaction data.

This initialization is critical for the subsequent clustering step. Randomly initialized embeddings are uninformative since they lack a meaningful geometric structure. The clustering algorithm will therefore produce clusters of poor quality, thus reducing the effectiveness of downstream components. In contrast, SVD, despite being one of the simplest baseline models for recommendation, provides interaction-informed embeddings that are accurately reflect user behavior. This ensures that items with similar profiles are embedded close together, enabling the clustering process to uncover semantically coherent groups of items, which are essential for the group-based sampling and adaptive bandit updates that follow. An experiment demonstrating this can be found in Section 6.3.

4.2 Clustering the Item Space

Let $E_{\text{cluster}} \in \mathbb{N}$ denote a predefined cut-off epoch. Before each training epoch $e \leq E_{\text{cluster}}$, we perform clustering over the current set of item embeddings $\{\mathbf{h}_i \in \mathbb{R}^d\}_{i=1}^N$, where N is the number of items and d is the embedding dimension. We employ HDBSCAN [24], a hierarchical density-based clustering algorithm, which automatically identifies clusters of varying shape and density without requiring the number of clusters to be specified a priori. Applied to the item embeddings, HDBSCAN assigns each item i a cluster label $c_i \in \mathbb{N} \cup \{\text{noise}\}$, marking a small fraction of items as noise due to their low density. Each resulting cluster groups items that share similar latent properties, often capturing semantic or behavioral patterns.

For each training instance $\mathcal{S}_u$ with a positive item i^+, we use its cluster label c_{i^+} to define three disjoint item groups used for structured negative sampling:

- the *target group*, consisting of all items i_j such that $c_{i_j} = c_{i^+}$;
- the *neighbours group*, consisting of items belonging to the two clusters whose centroids are closest (in Euclidean distance) to the centroid of c_{i^+};
- the *others group*, composed of all remaining items not included in the previous two groups.

This partitioning allows us to control the semantic similarity of negative items relative to the positive item. The clustering is recomputed before each epoch $e \leq E_{\text{cluster}}$ using the updated embeddings, and then frozen for all subsequent epochs to ensure consistency and computational efficiency.

4.3 Cluster-Based Thompson Sampling

To dynamically decide how likely to sample from each item group, we leverage the principles of *Thompson sampling* [1], a Bayesian algorithm widely used in online decision-making and multi-armed bandit problems. Thompson sampling is applied in scenarios where actions are taken sequentially in a manner that must balance between *exploration*, sampling from less-certain sources to gather more information and *exploitation*, favouring sources known to be effective. In our setting, this translates to learning, over time, which item groups are most likely to provide informative, i.e., hard negative examples.

Beta Distributions. To implement Thompson sampling, we associate each item group k (*target*, *neighbours*, and *others*) with a Beta distribution $\text{Beta}(\alpha_k, \beta_k)$, which encodes the belief about the group's propensity to yield hard negatives. Initially, we set $\alpha_k = \beta_k = 1$ for all groups, representing a uniform prior with maximum uncertainty.

At the beginning of each training epoch e, we draw a sample $\theta_k \sim \text{Beta}(\alpha_k, \beta_k)$ for each group k. These sampled values are then normalized to form a probability distribution over the groups, which governs the proportion of negative examples to be drawn from each group for the current training epoch.

The Decay Factor. To prevent over-sampling from the target cluster, which could lead to an accumulation of false negatives over time, we introduce a decay mechanism that progressively reduces the over-sampling influence. Specifically, after each epoch e, we multiply the sampled value θ_{target} by a decay factor $\left(1 - \frac{e}{E_{\text{target}}}\right)$, where E_{target} is a hyperparameter that defines the maximum number of epochs during which sampling from the target cluster is allowed. This decay encourages a gradual shift in focus toward neighbours and others groups, promoting broader exploration.

Updating Group Beliefs. After the model processes a epoch, we assess the difficulty of the sampled negatives. A negative item i_j sampled for user u is classified as *hard* if its predicted score y_{u,i_j} exceeds the positive ground truth score y_{u,i^+}. Otherwise, the negative is considered *easy*. Hard negatives are those that the model currently misclassifies or finds ambiguous, and thus are most useful

for learning. We use the ratio of hard to easy negatives to update the parameters α_k and β_k of the Beta distribution for each group. Let N_k^e be the total number of negatives sampled from group k in a given epoch e. Let h_k^e denote the number of negatives classified as hard, while $s_k^e = N_k^e - h_k^e$ corresponds to the remaining easy negatives. Then, $\forall k \in \{target, neighbours, others\}$, we apply the following updates:

$$\alpha_k^{e+1} \leftarrow \alpha_k^e + h_k^e, \qquad \beta_k^{e+1} \leftarrow \beta_k^e + s_k^e.$$

These updates reflect standard Bayesian posterior updating for Bernoulli trials using the Beta distribution as a conjugate prior. Over time, groups that consistently produce hard negatives will have a higher expected θ_k, increasing the probability of sampling negatives from those groups. In contrast, negatives will be sampled less frequently from groups that produce easy negatives, avoiding wasting the training signal.

This adaptive sampling strategy allows the model to dynamically balance exploration and exploitation throughout training. By updating its belief about each group's likelihood of producing hard negatives, the model gradually increases the sampling probability of the most informative groups. As a result, training is focused on the regions of the item space that provide the strongest learning signals, improving both efficiency and effectiveness in capturing user preferences.

4.4 Training Procedure

The overall CATS algorithm is summarized in Algorithm 1. Each component (SVD-based embedding initialization, periodic HDBSCAN clustering, Beta–Bernoulli Thompson sampling with decay, and hard/easy negative updates) has been detailed in Sections 4.1 to 4.3. In brief, the algorithm starts with the initialization of item embeddings. Then, at each epoch, CATS (1) updates item clusters (if $e \leq E_{cluster}$), (2) draws sampling weights θ_k from the current Beta priors and applies a decay to the target group, (3) allocates and draws negatives from the target, neighbour, and other groups in proportion to θ_k, (4) trains the sequential model using these negative samples using the sampled cross-entropy loss defined in 1, and (5) revises the beta parameters based on the number of hard/easy negatives.

Algorithm 1 Training procedure with CATS

Require: R, N, E_{cluster}, E_{target}, d, E

Initialize $\alpha_k^0 \leftarrow 1, \beta_k^0 \leftarrow 1$ for $k \in \{\text{target, neighbours, others}\}$
// Initialize item embeddings
$[U, \Sigma, V^T] \leftarrow \text{TruncatedSVD}(R, d)$
for each item i **do**
  $h_i \leftarrow V[i]$
end for
for $e = 1$ **to** E **do**
  if $e \leq E_{\text{cluster}}$ **then**
    // Compute item clusters
    clusters $\leftarrow$ HDBSCAN($\{h_i\}$)
    centroids $\leftarrow$ ComputeCentroids(clusters, $\{h_i\}$)
    Assign cluster c_i for each i
  end if
  // Update sampling probabilities of the 3 items groups
  for $k \in \{\text{target, neighbours, others}\}$ **do**
    $\theta_k^e \sim \text{Beta}(\alpha_k^e, \beta_k^e)$
  end for
  if $e < E_{\text{target}}$ **then**
    // Apply decay factor
    $\theta_{\text{target}} \leftarrow \theta_{\text{target}} \cdot (1 - \frac{e}{E_{\text{target}}})$
  else
    $\theta_{\text{target}} \leftarrow 0$
  end if
  total $\leftarrow \sum_k \theta_k$
  for k **do**
    $p_k \leftarrow \theta_k / \text{total}$
  end for
  // Negative sampling
  for each training instance (u, i^+) **do**
    $c_{\text{target}} \leftarrow$ cluster of i^+
    $c_{\text{neighbours}} \leftarrow$ two nearest clusters to c_{target}
    Partition items into groups: target, neighbours, others
    $N_k \leftarrow \lfloor N \times p_k \rfloor$
    for k **do**
      Sample N_k negatives uniformly from group k
    end for
    Train model and predict scores $\hat{y}_{u,i^+}$ and $\hat{y}_{u,i_j}, \forall j \in \mathcal{N}_u^k, \forall u \in U$
  end for
  // Update Beta distributions parameters
  for k **do**
    $h_k^e \leftarrow$ count hard negatives in k ($y_{u,i^-} > y_{u,i^+}$)
    $s_k^e \leftarrow$ count easy negatives in k ($y_{u,i^-} \leq y_{u,i^+}$)
    $\alpha_k^{e+1} \leftarrow \alpha_k^e + h_k^e$
    $\beta_k^{e+1} \leftarrow \beta_k^e + s_k^e$
  end for
end for

5 Experiments

This work is guided by the following research questions:

(1) **RQ1: Effectiveness** — How does CATS perform in comparison to standard and advanced negative sampling strategies?
(2) **RQ2: Generalizability** — Does CATS improve performance consistently across a variety of widely-used sequential RSs?
(3) **RQ3: Ablation Study** — Are all the components of CATS necessary to achieve its performance gains, or are some design choices more critical than others?

5.1 Experimental Setup

5.1.1 Datasets. Experiments are conducted on three benchmark datasets: MovieLens-1M[12], Amazon Beauty reviews[26], and BeerAdvocate reviews[23]. We remove any users or items that do not have associated positive feedback. In offline evaluation, prior work commonly employs a leave-one-out strategy, typically using the final interaction of each user as the test instance. However, this can introduce data leakage and compromise evaluation reliability [17, 25].

To address this, we adopt a temporal split by selecting a global timestamp at the 95th percentile of all interaction times. Interactions occurring before this point are allocated to training, while those after are reserved for testing, ensuring that test users are not included in the training set. For evaluation, we use each test user's most recent interaction, while their second-to-last interaction is used for validation. This setup avoids training on future data and allows for reliable model selection and early stopping. The training dataset statistics after preprocessing are reported in Table 1. Our computational resource limitations precluded experiments on larger datasets. Nonetheless, the datasets used in our study are standard in the literature and provide a reliable basis for comparative evaluation.

5.1.2 Models. We evaluate CATS using three sequential RSs: SASRec [18] (self-attention), BERT4Rec [33] (bidirectional transformer), and BSARec [32] (self-attention with frequency-aware filtering).

Table 1: The datasets statistics after preprocessing

Dataset	#Users	#Items	#Actions	Sparsity
MovieLens-1M	5,227	3,411	724,964	95.93%
All Beauty	1,184	1,027	6,593	99.46%
Beer Advocate	10,700	20,588	618,003	99.72%

5.1.3 Baselines and Metrics. We compare with several negative sampling strategies:

- **Uniform** [29]: samples negative items uniformly at random.
- **Popularity-based** [5]: samples negatives in proportion to item popularity.
- **Dynamic Negative Sampling (DNS)** [43]: selects the highest-scoring item among a set of randomly sampled candidates as the negative example.
- **DNS+** [31]: improves upon DNS by dynamically adjusting the hardness of negative samples during training, thereby mitigating the false negative problem.
- **GNNO** [9]: constructs a weighted item transition graph and samples negatives based on neighborhood overlap. Items with high neighborhood similarity are excluded to avoid false negatives.

We report standard top-k ranking metrics: Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG) at $k = 5, 10, 20, 50$. The ranking is performed across the entire set of items, as the sampled metrics have been shown to be inconsistent with their exact version [19].

5.1.4 Implementation details. We set the maximum sequence length to 200 for the ML-1M and BeerAdvocate datasets, and to 20 for Amazon Beauty. The item embedding dimension is fixed at 64. Models are trained for 600 epochs.
To automatically tune hyperparameters, we adopt Hyperopt within the RayTune framework [22]. Hyperopt leverages Bayesian optimization to efficiently explore the search space, with a trial budget of 200. The search space is defined as follows: batch size ∈ {8, 16, 32, 64, 256}; dropout rate ∈[0.1, 0.5]; learning rate for the Adam optimizer ∈ $[10^{-4}, 10^{-2}]$; number of negative samples ∈ {8, 16, 32, 64, 100, 150, 256, 300}; number of attention blocks ∈ {1, 2, 3, 4}; number of attention heads ∈ {1, 2, 4}. For BSARec, we follow the original settings: $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $c \in \{1, 3, 5, 7, 9\}$.
For DNS+, the number of top-scoring negatives considered is chosen from {2, 3, 4, 5, 8, 10, 15, 20, 30, 50}. For GNNO, the parameter c is set to 0 (as per the original paper), while λ_{max} and c are sampled uniformly from [0, 1]. For CATS the E_{target} and $E_{cluster}$ parameters are both sampled from {2,5,10,15,20,30,50}.

The full set of best hyperparameters configurations and the code are available in our GitHub repository for reproducibility: https://github.com/gditeodoro/CATS.

5.1.5 Software and Hardware. We implemented our experiments in Python 3.11, primarily using PyTorch 2.6.0 for model development and training. Clustering operations were performed using scikit-learn 1.6.1 and data manipulation relied on Pandas 2.2.3 and NumPy 2.1.3 Evaluation metrics and logging were handled via TensorBoard and custom utilities. All experiments were run in a reproducible environment with fixed random seeds and deterministic settings where applicable. The complete list of requirements and instructions to execute the code can be found in our GitHub repository: https://anonymous.4open.science/r/CATS-1949/README.md.

Experiments were conducted on a workstation equipped with an AMD Ryzen 9 7950X 16-core (32-thread) processor, an NVIDIA GeForce RTX 4090 GPU, 128GB of RAM, and a 2TB NVMe SSD. The system ran Ubuntu 24.04 with kernel version 6.11.0-25-generic.

6 Results

6.1 RQ1: Effectiveness

Table 2 summarises the HR@{ 5, 10, 20, 50} and NDCG@{5, 10, 20, 50} values for each sampling strategy, across all three datasets and SASRec and BERT4Rec models. CATS achieves the top performance in 44 out of 48 cases, trailing the leader by at most 5% in the other 4 cases. Uniform and popularity sampling serve as strong, albeit suboptimal, baselines, while advanced sampling methods sometimes underperforms uniform sampling, highlighting the risk of overly aggressive hard-negative mining.

With SASRec on MovieLens, CATS yields a relative improvement in NDCG@10 of 21% over uniform sampling and 93% over popularity-based sampling. Compared to DNS+ and GNNO, gains of 23% and 31% are observed, respectively. When using BERT4Rec, our method matches GNNO's performance (difference <3%) while surpassing all other baselines by 10–82% in both HR@10 and NDCG@10.

The Amazon Beauty dataset is relatively small and dense, causing most samplers to quickly converge to high scores. While popularity sampling achieves competitive performance, advanced methods such as DNS+ either tie with or slightly surpass it in a few metrics, especially when using BERT4Rec. Nevertheless, CATS remains the top method in 15 out of 16 metrics, with the exception of HR@20 with SASRec.

On the Beer Advocate dataset, uniform sampling performs surprisingly well, outperforming most dynamic methods. However, our method outperforms uniform by 10% in NDCG@10 with SASRec and by 6% with BERT4Rec. Advanced samplers (DNS, DNS+ and GNNO) often perform worse than uniform sampling on this dataset, suggesting that an excessive emphasis on hard-negative mining can reduce performance when user tastes are niche and sparse. CATS avoids this pitfall by maintaining a calibrated balance

Table 2: Comparison of negative sampling strategies on MovieLens, Amazon Beauty, and Beer Advocate datasets using SASRec and BERT4Rec. The table reports Hit Rate (HR) and NDCG at cutoffs {5, 10, 20, 50}. Superscripts indicate a significant Wilcoxon signed-rank test at level 0.01***, 0.05** or 0.1*. CATS consistently achieves the best, in bold, or second-best results, underlined, results.

Dataset	Model	Sampling	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20	HR@50	NDCG@50
ML-1M	SASRec	Uniform	0.0653	0.0457	0.1040	0.0580	0.1547	0.0707	0.2387	0.0874
		Popularity	0.0440	0.0293	0.0653	0.0362	0.1080	0.0471	0.1813	0.0617
		DNS	0.0520	0.0361	0.0773	0.0444	0.1227	0.0556	0.2333	0.0775
		DNS+	0.0640	0.0454	0.1000	0.0569	0.1547	0.0707	0.2387	0.0874
		GNNO	0.0640	0.0438	0.0947	0.0536	0.1467	0.0667	0.2453	0.0862
		CATS (ours)	**0.0800****	**0.0583****	**0.1173***	**0.0701****	**0.1773*****	**0.0853*****	**0.2627***	**0.1020*****
	BERT4Rec	Uniform	0.0573	0.0381	0.0787	0.0451	0.1067	0.0521	0.1907	0.0687
		Popularity	0.0333	0.0217	0.0587	0.0298	0.0880	0.0371	0.1733	0.0538
		DNS	0.0533	0.0393	0.0827	0.0486	0.1240	0.0592	0.2200	0.0779
		DNS+	0.0533	0.0384	0.0813	0.0477	0.1293	0.0598	0.2027	0.0743
		GNNO	0.0613	**0.0458**	0.0920	**0.0557**	0.1253	0.0641	0.2107	0.0808
		CATS (ours)	**0.0667**	0.0438	**0.0987**	0.0543	**0.1413****	**0.0650**	**0.2373**	**0.0841***
Beauty	SASRec	Uniform	0.4237	0.4133	0.4576	0.4231	**0.5763**	0.4519	0.5763	0.4519
		Popularity	0.4915	0.4831	0.4915	0.4831	0.4915	0.4831	0.5763	0.5007
		DNS	0.4746	0.4621	0.4746	0.4621	0.4746	0.4621	0.4746	0.4621
		DNS+	0.4746	0.4683	0.4746	0.4683	0.4746	0.4683	**0.5932**	0.4928
		GNNO	0.4407	0.3720	0.4407	0.3720	0.4407	0.3720	0.5424	0.3942
		CATS (ours)	**0.5593***	**0.5207***	**0.5593***	**0.5207***	0.5593	**0.5207****	**0.5932**	**0.5277**
	BERT4Rec	Uniform	0.5763	0.5763	0.5763	0.5763	**0.5932**	0.5802	**0.5932**	0.5802
		Popularity	0.5763	0.5763	0.5763	0.5763	0.5932	0.5810	**0.5932**	0.5810
		DNS	**0.5932**	0.5870	**0.5932**	0.5870	**0.5932**	0.5870	**0.5932**	0.5870
		DNS+	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**
		GNNO	0.5593	0.5247	0.5763	0.5307	0.5763	0.5307	**0.5932**	0.5339
		CATS (ours)	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**	**0.5932**
Beer	SASRec	Uniform	0.0250	0.0168	0.0412	0.0220	0.0622	0.0272	**0.1094**	0.0364
		Popularity	0.0169	0.0115	0.0228	0.0133	0.0353	0.0165	0.0641	0.0221
		DNS	0.0166	0.0117	0.0298	0.0160	0.0571	0.0229	0.0972	0.0308
		DNS+	0.0203	0.0131	0.0339	0.0175	0.0534	0.0223	0.0957	0.0306
		GNNO	0.0232	0.0160	0.0379	0.0207	0.0604	0.0263	0.1035	0.0349
		CATS (ours)	**0.0287***	**0.0193***	**0.0442**	**0.0241**	**0.0648**	**0.0293**	0.1086	**0.0379**
	BERT4Rec	Uniform	0.0239	0.0157	0.0357	0.0195	0.0596	0.0255	0.1035	0.0341
		Popularity	0.0085	0.0052	0.0155	0.0075	0.0247	0.0097	0.0604	0.0167
		DNS	0.0195	0.0138	0.0342	0.0185	0.0563	0.0241	0.1079	0.0341
		DNS+	0.0155	0.0101	0.0309	0.0150	0.0508	0.0200	0.0972	0.0293
		GNNO	0.0151	0.0101	0.0302	0.0148	0.0460	0.0187	0.0987	0.0291
		CATS (ours)	**0.0250**	**0.0159***	**0.0398***	**0.0206**	**0.0622**	**0.0262**	**0.1090**	**0.0354**

between hard negatives (from the target and neighbouring clusters) and broader exploration.

6.2 RQ2: Generalizability

To assess whether the benefits of CATS extend beyond those of SASRec and BERT4Rec, we evaluate it with the recently proposed BSARec model. Due to space constraints, we report results only on the ML-1M and BeerAdvocate datasets, omitting Amazon Beauty, which is the smallest among them. Table 3 reports the HR@{5,10,20,50} and NDCG@{5,10,20,50} scores for BSARec using uniform, popularity-based, DNS, DNS+, GNNO, and CATS sampling methods.

On MovieLens-1M, CATS once again achieves the highest scores, with a relative gain of 5.1% in HR@10 over uniform sampling and 3.9% over popularity-based sampling. It also outperforms DNS+ by 3.0%, GNNO by 2.5%, and DNS by 1.8% in terms of NDCG@10.

Among the baselines on the Beer Advocate dataset, GNNO stands out as the most competitive, achieving the highest performance in NDCG@5 and HR@5. However, CATS outperforms GNNO by up

to 7% across all remaining metrics, highlighting its superior overall effectiveness on this dataset.

These results confirm that the cluster-aware Thompson sampling framework is model-agnostic and consistently improves ranking quality across different sequential architectures.

6.3 RQ3: Ablation Study

To assess the contribution of each component of CATS, we perform an ablation study on the MovieLens-1M and Beer Advocate datasets using SASRec (see Table 4). These two dataset-model combinations were selected due to computational efficiency and space constraints, but they still offer a representative picture of the method's behaviour.

The default configuration of CATS consistently achieves the best or near-best performance consistently across all metrics, thus validating the effectiveness of the full method.

Clustering Dynamics. The hyperparameter $E_{cluster}$ determines for how many epochs item clusters are recomputed based on the updated item embeddings from the previous training epoch. This ensures that the items groups used by CATS remain meaningful and reflect the model's evolving representation space before negative sampling and training resume in the next epoch.
The *Full Dynamic Clustering* variant, in which item clusters are recomputed at every epoch until the end of the training, does not provide substantial performance gains compared to our default approach (which stops clustering after a certain number of epochs $E_{cluster}$). While it marginally improves HR@50, it is computationally expensive and not clearly superior overall. Conversely, *No Dynamic Clustering*, where the initial clustering is done on the item embedding obtained by SVD and never updated, performs always worst, with the exception of HR@50 on MovieLens. This generally confirms that keeping the clustering fixed, as embeddings evolve via gradient descent, quickly makes the initial SVD-based grouping obsolete.

Importantly, the best-performing models across datasets always select moderate values of E_{cluster} (typically not exceeding 30), indicating that frequent reclustering is unnecessary. In fact, after a certain number of epochs, item representations stabilize, leading to negligible shifts in cluster assignments. We empirically verify this by computing a transition matrix between cluster assignments across consecutive epochs on ML-1M. The transition matrix is plotted in Fig. 2. In this matrix, rows correspond to clusters at epoch $e+1$ and columns to clusters at epoch e, with each cell indicating the proportion of items transitioning between clusters. The dominance of high values along the diagonal demonstrates that most items remain in the same or similar clusters over time. This stability further justifies limiting the frequency of reclustering, as later iterations yield diminishing returns while incurring computational overhead.

Initialization. Disabling the SVD initialization *(NO SVD Initialization)* for item embeddings and clustering on item embeddings obtained by random Xavier initialization [10], results in consistent performance drops. This suggests that a good initial representation is necessary for meaningful clustering in early training stages.

Sampling Strategy. We test multiple variations of the hard-negative sampling mechanism. The hyperparameter E_{target} determine form

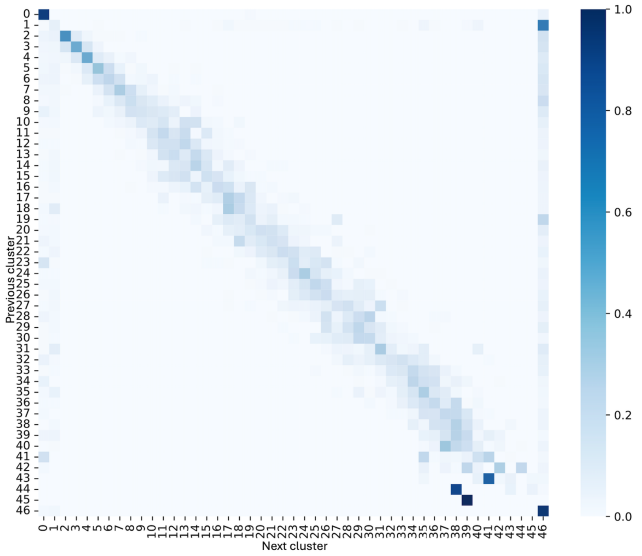

Figure 2: Transition matrix of items among clusters.

how many epoch CATS is allowed to sample negative from the target cluster. Instead, the decay factor of line 16 of Algorithm 1 is introduced to avoid to over-sample from the target cluster, which could contain false negatives. The *No Decay* variant samples from the target cluster using Thompson Sampling without adjusting the sampling probability θ_{target} via the decay factor. *Full Potential False Negatives* refers to the variant where $E_{\text{target}} = E$, meaning that negative samples are drawn from the target group throughout all training epochs. Conversely, *No Potential False Negatives* disable sampling from the target cluster entirely.
Our results show a consistent performance degradation in these two extreme cases, either never or always sampling from the target group, with the exception of NDCG@10 on ML-1M. This suggests that the target cluster likely includes items that are relatively similar to the ground-truth item but not too close, thus still functioning as valid hard negatives rather than false negatives. Therefore, sampling from this cluster can be beneficial in the early stages of training. However, if such sampling persists for too many epochs, the risk of including false negatives increases, reducing the model's ability to generalize. These findings support the need for a controlled and time-limited exploitation of the target cluster, as implemented by our decaying mechanism and the E_{target} hyperparameter.

Exploration Mechanism. *No Thompson*, which samples uniformly across the three clusters while retaining the decay schedule, performs reasonably well but never surpasses the full method. This confirms that balancing exploration via Thompson sampling is more effective than uniform sampling, even under a decaying schedule.

Overall, the ablation study confirms that each component (dynamic clustering, SVD initialization, Thompson-based exploration, and controlled hard-negative sampling) contributes to the performance of CATS.

Table 3: Comparison of negative sampling strategies on MovieLens, and Beer Advocate datasets using BSARec. The table reports Hit Rate (HR) and NDCG at cutoffs {5, 10, 20, 50}. Superscripts indicate a significant Wilcoxon signed-rank test at level 0.01***, 0.05** or 0.1*. CATS consistently achieves the best, in bold, or second-best results, underlined, results.

Dataset	Sampling	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20	HR@50	NDCG@50
ML-1M	Uniform	0.0720	0.0509	0.1067	0.0620	0.1640	0.0763	0.2507	0.0934
	Popularity	0.0573	0.0373	0.0773	0.0438	0.1213	0.0549	0.1933	0.0694
	DNS	0.0507	0.0341	0.0800	0.0439	0.1173	0.0534	0.2040	0.0707
	DNS+	**0.0747**	**0.0530**	0.1107	0.0642	0.1627	0.0774	0.2533	0.0956
	GNNO	0.0733	0.0495	0.1027	0.0589	0.1520	0.0715	0.2373	0.0884
	CATS (ours)	**0.0747**	0.0514	**0.1187***	**0.0656***	**0.1693***	**0.0784***	**0.2600****	**0.0963****
Beer	Uniform	0.0295	0.0199	0.0427	0.0242	0.0655	0.0299	0.1116	0.0389
	Popularity	0.0221	0.0144	0.0324	0.0176	0.046	0.021	0.0692	0.0256
	DNS	0.0247	0.0156	0.0416	0.0211	0.0626	0.0263	0.1219	0.0379
	DNS+	0.032	0.0189	0.0438	0.0227	0.0707	0.0295	0.1274	0.0408
	GNNO	**0.0339**	**0.0217**	0.0471	0.0259	0.0729	0.0323	0.1296	0.0434
	CATS (ours)	0.0313	0.021	**0.0493**	**0.0268**	**0.0781**	**0.034**	**0.1325**	**0.0447**

Table 4: Ablation study on CATS using SASRec over MovieLens-1M and Beer datasets. The best result is highlighted in bold. We analyze the impact of each component: dynamic clustering frequency, SVD initialization, sampling decay, and Thompson-based exploration. The default CATS configuration achieves the best overall performance, while removing or altering key components leads to dataset-specific drops in HR and NDCG, confirming the importance of all design choices.

Dataset	Model	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20	HR@50	NDCG@50
Beer	Default	**0.0287**	**0.0193**	**0.0442**	**0.0241**	**0.0648**	**0.0293**	0.1086	**0.0379**
	Full Dynamic Clustering	0.0258	0.0173	0.0401	0.0219	0.0637	0.0278	**0.1105**	0.0371
	Full Potential False Negatives	0.0133	0.0087	0.0243	0.0123	0.0357	0.0152	0.0784	0.0236
	No SVD Initialization	0.0184	0.0120	0.0335	0.0169	0.0593	0.0233	0.1082	0.0330
	No Decay	0.0265	0.0181	0.0387	0.0220	0.0574	0.0267	0.1009	0.0352
	No Potential False Negatives	0.0261	0.0172	0.0412	0.0220	0.0622	0.0272	0.1060	0.0358
	No Dynamic Clustering	0.0085	0.0043	0.0173	0.0072	0.0269	0.0096	0.0563	0.0153
	No Thompson	0.0261	0.0164	0.0357	0.0195	0.0585	0.0253	0.1013	0.0337
ML-1M	Default	**0.0800**	**0.0583**	**0.1173**	**0.0701**	**0.1773**	**0.0853**	0.2627	**0.1020**
	Full Dynamic Clustering	0.0693	0.0541	0.1107	0.0676	0.1573	0.0793	0.2480	0.0972
	Full Potential False Negatives	0.0787	0.0567	0.1140	**0.0701**	0.1680	0.0824	0.2640	0.1015
	No SVD Initialization	0.0720	0.0533	0.1133	0.0669	0.1547	0.0772	0.2387	0.0939
	No Decay	0.0747	0.0543	0.1160	0.0677	0.1613	0.0792	0.2587	0.0985
	No Potential False Negatives	0.0653	0.0448	0.1080	0.0583	0.1627	0.0719	0.2467	0.0885
	No Dynamic Clustering	0.0720	0.0493	0.1160	0.0634	0.1707	0.0771	**0.2667**	0.0958
	No Thompson	0.0693	0.0472	0.1093	0.0602	0.1560	0.0721	0.2560	0.0921

7 Conclusions

In this work, we introduced CATS, a novel negative sampling framework for sequential recommendation that combines hierarchical clustering of items with Thompson sampling to adaptively select informative negatives. By dynamically identifying latent item groups and employing Beta-Bernoulli bandit updates, our approach balances exploitation of structurally similar hard negatives with exploration of diverse candidates. Empirical results on standard benchmarks show that CATS consistently improves ranking performance of SASRec, BERT4Rec and BSARec over uniform, popularity, and other advanced sampling strategies. Our ablation studies confirm that each component of the method contributes to the overall improvement, emphasising the importance of their combined effect. Future research directions include extending this framework to incorporate user-centric, i.e. tailor negative sampling to indivudual behaviours, and content-aware, i.e. leverage item metadata, clustering features. We will also explore non-linear or data-driven decay strategies that adapt to convergence signals. All code and experimental artifacts are publicly released to foster reproducibility and further research in adaptive negative samping for recommendation. We believe CATS offers a versatile blueprint for integrating latent structure and Bayesian exploration in large-scale RSs.

8 GenAI Usage Disclosure

In accordance with ACM's guidelines on the use of generative AI tools, we disclose that generative AI technologies were used solely to assist in code debugging and grammar checking during the preparation of this paper. All research ideas, experiments, analysis, and writing were conducted and critically reviewed by the authors. No part of the scientific content or creative reasoning has been generated or substantially rewritten by generative AI tools.

Acknowledgments

This work was partially supported by the FoReLab and CrossLab projects (Departments of Excellence), the NEREO PRIN project (Research Grant no. 2022AEFHAZ) funded by the Italian Ministry of Education and Research (MUR), and the NVIDIA Academic Grants Program 2026. This work has been partially supported by the AI-Sec Lab initiative between Sapienza University of Rome and the CYBER 4.0 Competence Center.

References

[1] Shipra Agrawal and Navin Goyal. 2012. Analysis of Thompson Sampling for the Multi-armed Bandit Problem. In *Proc. 25th Annual Conference on Learning Theory*. 39.1–39.26.

[2] Filippo Betello, Antonio Purificato, Federico Siciliano, Giovanni Trappolini, Andrea Bacciu, Nicola Tonellotto, and Fabrizio Silvestri. 2025. A Reproducible Analysis of Sequential Recommender Systems. *IEEE Access* 13 (2025), 5762–5772.

[3] 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.

[4] Jiawei Chen, Can Wang, Sheng Zhou, Qihao Shi, Yan Feng, and Chun Chen. 2019. Samwalker: Social recommendation with informative sampling strategy. In *Proc. WWW*. 228–239.

[5] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In *Proc. SIGKDD*. 767–776.

[6] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation. In *Proc. RecSys*. 143–153.

[7] Giulia Di Teodoro, Federico Siciliano, Nicola Tonellotto, and Fabrizio Silvestri. 2025. A Theoretical Analysis of Recommendation Loss Functions under Negative Sampling. In *Proc. IJCNN*. 1–9.

[8] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and robustify negative sampling for implicit collaborative filtering. *Proc. NeurIPS* (2020), 1094–1105.

[9] Lu Fan, Jiashu Pu, Rongsheng Zhang, and Xiao-Ming Wu. 2023. Neighborhood-based hard negative mining for sequential recommendation. In *Proc. SIGIR*. 2042–2046.

[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*. 249–256.

[11] Per Christian Hansen. 1987. The truncated SVD as a method for regularization. *BIT Numerical Mathematics* 27 (1987), 534–553.

[12] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM TIIS* 5, 4 (2015), 1–19.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proc. WWW*. 173–182.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proc. ICLR*.

[15] 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 *Proc. SIGKDD*. 2553–2561.

[16] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In *Proc. SIGKDD*. 665–674.

[17] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A critical study on data leakage in recommender system offline evaluation. *ACM TOIS* 41, 3 (2023), 1–27.

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proc. ICDM*. 197–206.

[19] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proc. SIGKDD*. 1748–1757.

[20] Riwei Lai, Rui Chen, Qilong Han, Chi Zhang, and Li Chen. 2024. Adaptive hardness negative sampling for collaborative filtering. In *Proc. AAAI*, Vol. 38. 8645–8652.

[21] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proc. CIKM*. 1419–1428.

[22] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. Tune: A Research Platform for Distributed Model Selection and Training. In *Proc. ICML AutoML Workshop*.

[23] Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proc. ICDM*. 1020–1025.

[24] Leland McInnes, John Healy, Steve Astels, et al. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software* 2, 11 (2017), 205.

[25] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring Data Splitting Strategies for the Evaluation of Recommendation Models. In *Proc. RecSys*. 681–686.

[26] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proc. EMNLP*. 188–197.

[27] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *Proc. WWW*. 1443–1453.

[28] Steffen Rendle. 2021. Item recommendation from implicit feedback. In *Recommender Systems Handbook*. Springer, 143–171.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. UAI*. 452–461.

[30] Jinseok Seol and Jaesik Choi. 2025. Context-aware Negative Sampling for Sequential Recommendation. *IEEE Access* (2025).

[31] Wentao Shi, Jiawei Chen, Fuli Feng, Jizhi Zhang, Junkang Wu, Chongming Gao, and Xiangnan He. 2023. On the theories behind hard negative sampling for recommendation. In *Proc. WWW*. 812–822.

[32] Yehjin Shin, Jeongwhan Choi, Hyowon Wi, and Noseong Park. 2024. An attentive inductive bias for sequential recommendation beyond the self-attention. In *Proc. AAAI*, Vol. 38. 8984–8992.

[33] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proc. CIKM*. 1441–1450.

[34] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proc. WSDM*. 565–573.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proc. NeurIPS* 30 (2017).

[36] Can Wang, Jiawei Chen, Sheng Zhou, Qihao Shi, Yan Feng, and Chun Chen. 2023. SamWalker++: Recommendation With Informative Sampling Strategy . *IEEE TKDE* 35, 02 (2023), 2004–2018.

[37] 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 *Proc. SIGIR*. 515–524.

[38] Yidan Wang, Xuri Ge, Xin Chen, Ruobing Xie, Su Yan, Xu Zhang, Zhumin Chen, Jun Ma, and Xin Xin. 2025. Exploration and Exploitation of Hard Negative Samples for Cross-Domain Sequential Recommendation. In *Proc. WSDM*. 669–677.

[39] Ga Wu, Maksims Volkovs, Chee Loong Soon, Scott Sanner, and Himanshu Rai. 2019. Noise contrastive estimation for one-class collaborative filtering. In *Proc. SIGIR*. 135–144.

[40] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, and Tianyu Qiu. 2024. On the effectiveness of sampled softmax loss for item recommendation. *ACM TOIS* 42, 4 (2024), 1–26.

[41] Lanling Xu, Jianxun Lian, Wayne Xin Zhao, Ming Gong, Linjun Shou, Daxin Jiang, Xing Xie, and Ji-Rong Wen. 2022. Negative sampling for contrastive representation learning: A review. *arXiv preprint arXiv:2206.00212* (2022).

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proc. SIGKDD*. 974–983.

[43] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proc. SIGIR*. 785–788.