
Exploring Alignment Techniques in Large Language Models for Long-Tail Item Recommendation

Wei Bin Au Yeong
waueyong@andrew.cmu.edu

Allyne Zhang
allynez@andrew.cmu.edu

Xun Yi Lim
xunyi1@andrew.cmu.edu

Yogesh Adhi Narayan
yadhinar@andrew.cmu.edu

Abstract

Large Language Models (LLMs) are increasingly explored in recommender systems (RecSys), but their advantages over traditional models remain uncertain. A key area where LLMs could add value is long-tail item recommendation, where less popular items form the majority of options. This project explores the effectiveness of LLMs for long-tail recommendation by first evaluating collaborative filtering models and LLM-based approaches in isolation. We then implement an alignment-based architecture, adapted from recent work, that integrates collaborative signals with LLM-derived semantic representations to combine personalization with semantic generalization. Our goal is to assess whether such alignment improves recommendation quality for long-tail items without sacrificing accuracy or efficiency in warm-start settings.

1 Overview and Context

1.1 Motivation

Large Language Models (LLMs) have shown success across various domains and are increasingly explored for adoption in recommender systems (RecSys). However, their real-world adoption remains limited due to uncertainty about their advantages over traditional collaborative filtering methods.

One key area where LLMs may offer distinct value is **long-tail item recommendation**—the task of recommending less popular items that collectively make up the majority of a catalog. Improving long-tail recommendations can benefit platforms such as e-commerce, streaming services, and social media by enhancing both content diversity and user personalization. For instance, better long-tail coverage in e-commerce promotes niche products, while in streaming and social platforms it helps users discover non-mainstream content.

While recent studies suggest that LLMs’ pretrained knowledge and reasoning capabilities may improve generalization (Wu et al., 2024; Liu et al., 2024; Luo et al., 2024), others highlight challenges such as semantic bias and limited use of behavioral signals (Lichtenberg et al., 2024). Given these mixed findings, this project aims to assess their potential for improving long-tail recommendations while balancing effectiveness and feasibility of deployment. A clearer understanding of when and how LLMs add value in RecSys is essential to guiding future research and practical system design.

1.2 Objective

This project aims to systematically assess the effectiveness of LLMs for long-tail recommendation. Specifically, we explore one practical approach that combines the semantic reasoning abilities of pretrained LLMs with the personalization strengths of collaborative filtering. By aligning representa-

tions from both sources, we aim to leverage the benefits of LLMs while addressing their limitations in terms of scalability and reliance on semantic data alone.

Our evaluation focuses on three key areas: (1) increasing long-tail item exposure by surfacing diverse and less popular items in the top ranks, (2) maintaining ranking accuracy to ensure that relevant items are still prioritized for the user, and (3) achieving reasonable computational efficiency to assess the feasibility of real-world deployment.

2 Related Work and Background

2.1 Long-Tail Item Recommendation in Collaborative Filtering

Deep learning-based collaborative filtering (CF) models such as Wide&Deep (Cheng et al., 2016) and DeepFM (Guo et al., 2017) are widely adopted due to their ability to model complex user-item interactions and deliver strong accuracy on head items. However, these models often underperform on long-tail items (Park and Tuzhilin, 2008; Carnovalini et al., 2025). Because they are typically optimized to maximize overall ranking accuracy, such models tend to favor popular content, unintentionally reinforcing popularity bias (Kowald, 2025). This skew leads to a lack of exposure for niche items, undermining personalization, fairness, and catalog coverage.

To evaluate performance in long-tail settings, recent work has introduced metrics that go beyond accuracy. **Coverage** measures the proportion of unique items recommended across all users, reflecting catalog utilization (Kaminskas and Bridge, 2016). **Diversity** captures the dissimilarity among items within a user’s recommendation list, often through entropy or embedding-based distance (Vargas and Castells, 2011). Finally, **average popularity bin** and related metrics assess whether recommendations skew toward popular items or effectively include long-tail content (Abdollahpouri et al., 2019; Carnovalini et al., 2025).

These metrics help highlight the limitations of collaborative filtering when applied to long-tail recommendation tasks. Our work adopts them to systematically assess whether language-model-based representations can mitigate these limitations while preserving accuracy and practical deployability.

2.2 Leveraging LLMs for Recommendation

Transformer-based language models like BERT and GPT have recently gained traction in recommendation systems for their ability to extract rich semantic representations from textual metadata. These capabilities are particularly beneficial in cold-start and sparse data settings, where CF models underperform.

CTR-BERT (Muhammed et al., 2021), for example, uses BERT to encode item descriptions and user profiles for click-through prediction, enabling generalization to unseen items. Prompt-based methods (Bao et al., 2023) treat recommendation as a text completion task, supporting zero- and few-shot inference. Other models like Coral (Wu et al., 2024) and LLM-ESR (Liu et al., 2024) integrate LLM-derived semantic features into user-item representations, improving diversity and robustness for long-tail content.

Despite these advantages, LLMs face challenges in warm-start scenarios and impose high inference costs. To address this, recent work explores using LLMs during training for representation learning, then removing them from inference to improve efficiency. Our work follows this direction, using LLMs to enrich item and user embeddings while avoiding runtime complexity.

2.3 Hybrid and Alignment-Based Approaches

To overcome the limitations of both collaborative filtering and language modeling, recent work has focused on hybrid architectures that integrate behavioral signals with semantic representations. These models aim to combine the accuracy of traditional CF in warm settings with the generalization capabilities of language models in sparse or cold-start scenarios.

CTRL (Li et al., 2025) aligns tabular collaborative features with text embeddings through contrastive learning, enabling LLMs to operate over structured data without discarding CF strengths. A-LLMRec (Kim et al., 2024) takes this further by projecting both user and item embeddings into the token

space of a frozen LLM, allowing it to score candidates using a prompt-based interface enriched with collaborative history.

These alignment-based methods avoid inference-time LLM computation by leveraging pretrained models during training or preprocessing. This makes them appealing for real-world deployment, where latency and scalability are concerns. Our work follows this line by adopting a similar architectural strategy with lightweight encoders and simplified projection layers. Rather than propose a novel framework, we evaluate this alignment approach in the context of long-tail recommendation—testing whether semantic-aware alignment improves item coverage and diversity without compromising relevance or practicality.

3 Methodology

3.1 Model Description

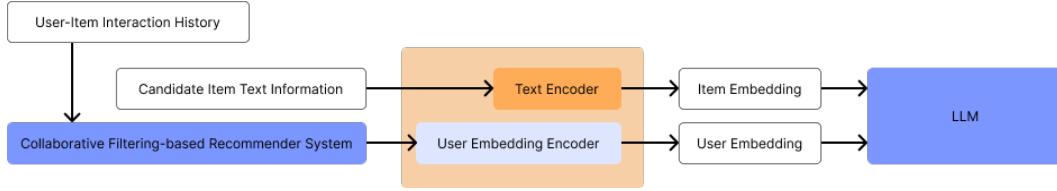


Figure 1: Model Overview

The alignment model consists of 2 main components, (1) the text encoder that generates an embedding from the textual description of the pool of candidate items that could be recommended and (2) the user encoder that transforms the user embedding from an existing collaborative filtering-based recommendation system model to an embedding in the same space as the LLM to be used. The item and user embeddings from the text and user encoders respectively will then be added into a text prompt for the LLM to perform the recommendation task. This allows for the LLM to consider collaborative information from the user’s previous item interactions during the recommendation process.

Following A-LLMRec, we would be using SASRec Kang and McAuley (2018) as our backbone collaborative-based recommender system, but will be using mGTE Zhang et al. (2024) and GPT-2 Radford et al. (2019) for the text encoder and language model respectively. Overall our model used for inference would have 168M parameters, with the full model parameter breakdown in Figure 13.

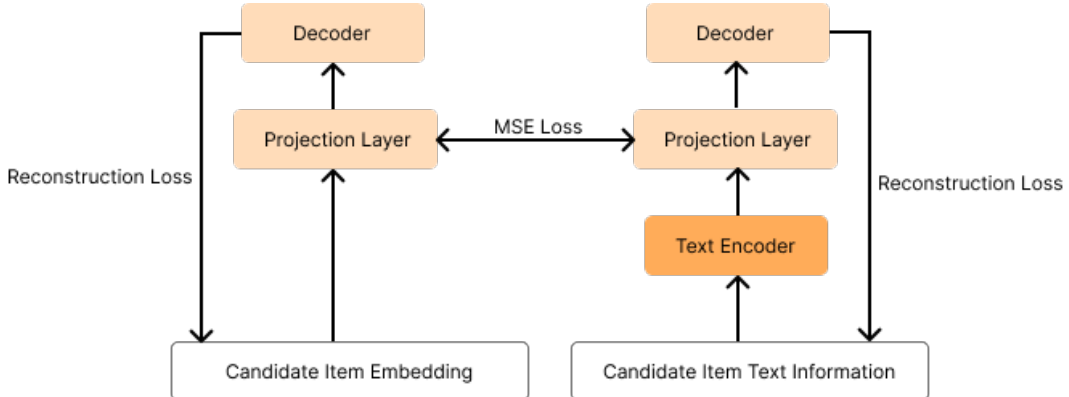


Figure 2: Text Encoder Training Process

1. Text Encoder: The text encoder will be trained in conjunction with an item encoder that takes the original item embedding as input as shown in Figure 2. This enables the text encoder to align itself

to the original item embedding representations. We will be using SBERT Reimers and Gurevych (2019) as our text encoder and during the training process, we will use a single-layer perceptron to project both item and text embeddings into the same space in order to align them. Furthermore, to ensure that the projection layers do not learn a trivial representation solely to maximise the MSE Loss (e.g. have all 0 weights so that both projection layers have the exact same output regardless of the input), we introduce decoders and a reconstruction loss for both item and user inputs to ensure that the projection layer learns a meaningful representation of their respective inputs.

2. **User Encoder:** The user encoder takes in the output from existing collaborative filtering-based recommendation system models and will be trained with the output from the text encoder to align their representations. Items that were previously rated by the user would be passed through the text encoder to generate embeddings. We would do a dot-product with the user embedding and the respective item embeddings and train the user encoder to maximise the similarity of positive items (items that users have rated above 3 out of 5) and minimise the similarity of the negative items (items that users have rated below 3 out of 5).

A similar approach in A-LLMRec (Kim et al., 2024) has been proven to work well, hence we chose a similar but simpler approach to try out. Beyond aligning the collaborative information to the LLM latent space to help LLM-based recommendation systems outperform traditional collaborative-based systems in warm settings, we hope to explore the use of the encoders as a way to shortlist long-tailed items as potential recommendation candidates.

3.2 Dataset

This project utilizes the **2018 Amazon Review dataset** (Ni et al., 2019), which has been widely adopted in recommendation research (Zhou et al., 2018; Li et al., 2025). Following (Kim et al., 2024), we use the **Movies & TV** category, which contains approximately 8.7 million user-item interactions across 204,000 unique items.

To balance computational requirements and to construct a test set that meaningfully evaluates model performance on long-tail items, we curate a filtered 30% subset of the data, resulting in 1.06 million interactions and 74,000 unique items. The dataset includes both **user-item interaction tuples** (`user_id`, `item_id`), where presence of a tuple indicates a positive interaction (i.e., assigned a value of 1), and cleaned **item metadata** (e.g., title, description, category, brand). This metadata is carefully processed from noisy original sources to promote better generalization; it encourages models to move beyond pure historical interaction patterns and recommend based on item characteristics.

To prepare the dataset for training and evaluation, we apply the following preprocessing pipeline:

- **Metadata Cleaning:** Item metadata is standardized and formatted to assign each item a consistent set of categorical and textual features.
- **Filtering Noisy Data:** Following (Kim et al., 2024), we remove users and items with fewer than five interactions to reduce noise and prevent overfitting. Importantly, this filtering does not remove true long-tail items, which typically have more than five interactions.
- **Assigning Popularity Bins:** Each item is labeled with a popularity bin (Q1 to Q4) based on the quartile of its interaction frequency, with Q1 representing the least popular (long-tail) items. This additional labeling supports targeted evaluation and sampling.
- **Train/Val/Test Split:** Following common practice, data is split chronologically based on timestamps. For each user, the last interaction is used for the test set, the second-to-last for validation, and the remaining for training.
- **Negative Sampling (Train):** One negative sample is randomly selected for each positive interaction from items the user has not interacted with.
- **Negative Sampling (Test):** For evaluation, 100 negative candidates are sampled per user. Crucially, sampling probabilities are weighted by item popularity, ensuring greater inclusion of long-tail items. This novel long-tail-aware sampling approach allows us to rigorously test how well models surface underrepresented items—an aspect often overlooked in standard setups, where long-tail items are rarely included among the candidates that models are asked to rank and recommend.
- **Model-Specific Preparation:** Our baseline models operate in a pointwise manner, predicting individual scores for user-item pairs. In contrast, our LLM-based alignment model uses

a listwise approach, ranking a full set of candidate items jointly. As a result, the data must be prepared in different formats to accommodate each model type.

These steps ensure that the dataset is structured to support rigorous training and evaluation of recommender models, while preserving user behavior patterns and enabling robust assessment of both relevance and diversity. Notably, our careful metadata processing and long-tail-aware sampling are designed to highlight the potential of aligned LLM representations in surfacing underexposed content.

3.3 Evaluation Metric

To evaluate model performance, we adopt a ranking-based setup where each model is tasked with ranking 101 candidate items for each user: one ground-truth positive item and 100 sampled negatives (see Section 3.2 for sampling details). The model assigns a score to each candidate, and we evaluate its ability to both prioritize the true positive item and surface diverse, less popular recommendations. Formal definitions for all metrics are provided in Appendix 9.1.

Our evaluation covers three metric categories:

Computation. These metrics assess model efficiency and deployment feasibility:

- **Model Size (M):** Number of trainable parameters (in millions), reflecting memory and storage demands.
- **Inference Time (ms):** Average latency per prediction, important for real-time applications.

Long-Tail Exposure (Primary Objective). These metrics evaluate the model’s ability to recommend less popular (long-tail) items within the top- k results, where $k = 20$:

- **Average Rank Bin:** The mean rank (1 = most popular, 10 = least popular) of the top- k items; lower values indicate the items recommended tend to be more popular.
- **Coverage:** The percentage of unique categories recommended across all users, reflecting catalog breadth.
- **Diversity:** The dissimilarity among recommended items within each user’s list (e.g., via entropy or embedding distance), encouraging non-redundant recommendations. This is simplified to number of unique categories recommended in top- k / k and averaged across all users.

Accuracy. These metrics ensure that promoting long-tail exposure does not compromise ranking relevance:

- **Mean Reciprocal Rank (MRR):** The average reciprocal rank of the true positive item. Taking $1/\text{MRR}$ provides the average position the true positive item is ranked at.
- **Hit@ k :** The proportion of cases where the true item appears among the top k recommendations.

Together, these metrics provide a comprehensive evaluation of our alignment-based approach: measuring its effectiveness at surfacing long-tail items, preserving ranking accuracy, and maintaining computational feasibility for real-world deployment.

3.4 Loss Function

The training is split into two stages. Stage 1 involves contrastive pretraining between text and item embeddings. In this stage, Binary Cross Entropy Loss is used and instead of traditional binary classification, it is adapted to determine whether a given item (positive or negative) is the correct next item for a user sequence. The equations for these are found in A.9 and 1. This encourages higher scores for positive samples and lower scores for negative samples.

$$\mathcal{L}_{\text{BCE}} = \text{BCE}(\hat{y}^+, 1) + \text{BCE}(\hat{y}^-, 0) \quad (1)$$

In addition, a matching loss is used to align the intermediate projection vectors of item embeddings from the recommender model and text embeddings from the text encoder. This encourages the

two domains which are text and recommendation to share a common representational space. The equations are found in 2 and A.10

$$\mathcal{L}_{\text{match}} = \text{MSE}(z_{\text{rec}}^+, z_{\text{text}}^+) + \text{MSE}(z_{\text{rec}}^-, z_{\text{text}}^-) \quad (2)$$

An item embedding reconstruction loss is also computed to ensure that the projected embeddings retain the original item information. The equation for this is found in 3. A similar text embedding reconstruction loss is applied to SBERT outputs, encouraging the network to reconstruct text embeddings accurately after MLP projection. The equation for this is found in 4.

$$\mathcal{L}_{\text{recon}} = \text{MSE}(\hat{z}_{\text{rec}}^+, z_{\text{rec}}^+) + \text{MSE}(\hat{z}_{\text{rec}}^-, z_{\text{rec}}^-) \quad (3)$$

$$\mathcal{L}_{\text{text-recon}} = \text{MSE}(\hat{z}_{\text{text}}^+, z_{\text{text}}^+) + \text{MSE}(\hat{z}_{\text{text}}^-, z_{\text{text}}^-) \quad (4)$$

The total loss for Phase 1 is computed as a weighted combination of these components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{match}} + 0.5 \cdot \mathcal{L}_{\text{recon}} + 0.2 \cdot \mathcal{L}_{\text{text-recon}} \quad (5)$$

In Stage 2, we pretrain the LLM to understand and incorporate user and item embeddings for recommendation. The LLM is trained to generate the correct item title using a causal language modeling loss, which conditions the prediction on the user embedding, interaction history, and a candidate item set. This process helps align the LLM’s language generation capabilities with embedding-based signals from the recommender system.

4 Baseline and Extensions

4.1 Choice of Baseline

To systematically compare modeling strategies, we evaluate two distinct classes of recommendation models: (1) traditional collaborative filtering models, and (2) a pretrained language model. Both classes are tested under two input settings: (1) using only interaction data, and (2) using interaction data combined with item metadata. This comparison allows us to examine how each modeling approach leverages different types of information, and how they perform in terms of long-tail exposure, semantic generalization, and ranking accuracy.

4.1.1 Collaborative Model Baselines

We have selected **DeepFM** (Guo et al., 2017), **DeepFFM** (Zhang et al., 2019), and **XDeepFM** (Lian et al., 2018) as our baseline models. These are widely used traditional recommendation models that leverage factorization machines and deep learning to capture both low-order and high-order feature interactions.

- **DeepFM** integrates a Factorization Machine (FM) with a deep neural network (DNN), allowing it to model both linear and non-linear feature interactions without manual feature engineering.
- **DeepFFM** extends Field-aware Factorization Machines (FFM) by incorporating field-specific embeddings, improving its ability to handle interactions between categorical features.
- **XDeepFM** enhances DeepFM by introducing a Compressed Interaction Network (CIN), which explicitly models higher-order feature interactions in an efficient manner.

These models serve as strong collaborative filtering baselines, performing well on popular (head) items with sufficient interaction history but often struggling to generalize to long-tail items, particularly those with limited collaborative signals. This limitation motivates the inclusion of language models that can leverage semantic information from item metadata to improve generalization and long-tail coverage.

4.1.2 Language Model Baseline

To isolate the semantic reasoning ability of language models, we evaluate a **GPT-2**-based recommendation baseline in a zero-shot setting. This enables us to study how well a pretrained language model can recommend items without relying on collaborative filtering signals.

We use a **few-shot prompting** approach to simulate recommendation: for each user, GPT-2 is provided with the titles of previously interacted items, followed by 101 candidate items (1 positive and 100 negatives). The model is queried pointwise to score each candidate based on its relevance to the user’s history. The prompt template used is shown in Appendix 9.2.

This baseline operates purely on metadata and pretrained language representations. It allows us to evaluate whether semantic matching alone can effectively generalize to novel or long-tail items.

4.2 Baseline Evaluation

4.2.1 Evaluation Strategy

Our baseline evaluation compares collaborative filtering and language-based models in isolation to understand their respective strengths and limitations, motivating our alignment-based approach that integrates both.

Each model is tested under two input settings: (1) interaction-only, using just user and item IDs, and (2) interaction plus metadata, where item-side features such as titles and descriptions are included. These settings allow us to assess how well models perform with behavioral signals alone, and whether semantic information helps improve long-tail coverage and diversity.

We focus on three aspects: (1) behavioral performance from interactions alone, (2) semantic generalization enabled by metadata, and (3) trade-offs between long-tail exposure and ranking accuracy. Together, these comparisons clarify how different modeling strategies behave and why combining them may offer the best of both worlds.

4.2.2 Evaluation Results

We evaluate the performance of three collaborative filtering models (DeepFM, DeepFFM, and XDeepFM) and a language model baseline (GPT-2) under the two input settings and summarize the results in Tables 1 and 2, grouped by computation, long-tail exposure, and accuracy metrics.

Table 1: Results using interaction data only.

Model	Computation		Long-Tail Exposure			Accuracy	
	Size (M)	Inf. Time (ms)	Coverage	Diversity	Avg. Rank Bin	MRR	Hit@10
DeepFM	0.0251	0.2239	1.0000	0.0816	4.0121	0.0582	0.1679
DeepFFM	0.1016	0.2239	0.9902	0.0814	3.9721	0.0675	0.1889
XDeepFM	0.0411	0.1915	0.9902	0.0813	3.9769	0.0678	0.1917
GPT-2	137	89,822	1.0000	0.0832	3.9350	0.0126	0.0500

Table 2: Results using interaction data + item metadata.

Model	Computation		Long-Tail Exposure			Accuracy	
	Size (M)	Inf. Time (ms)	Coverage	Diversity	Avg. Rank. Bin	MRR	Hit@10
DeepFM	0.0251	0.1787	0.9902	0.0776	4.4723	0.0707	0.1159
DeepFFM	0.1016	1.4599	0.9902	0.0694	3.5989	0.0433	0.1268
XDeepFM	0.0412	0.1935	1.0000	0.0761	3.7669	0.0648	0.1654
GPT-2	137	342,390	1.0000	0.0853	3.8433	0.0459	0.0833

The baseline results highlight the complementary strengths and limitations of collaborative filtering and language-based models.

Collaborative Filtering Models DeepFM, DeepFFM, and XDeepFM achieve strong accuracy on MRR and Hit@10 across both settings, reflecting their strength on head items with ample interaction data. However, they struggle with long-tail exposure. Despite near-perfect coverage, their average rank bin remains skewed toward popular items, and recommendation diversity is low, indicating repeated recommendation of similar high-frequency content.

Language Model Baseline (GPT-2) GPT-2 consistently performs poorly on ranking accuracy. While it shows improvement in diversity, it rarely ranks the true item in the top-10. Its inference cost is also prohibitively high, making it impractical for deployment.

Effect of Metadata Adding metadata has limited effect on collaborative models, which are not designed to exploit textual features. GPT-2 sees a small gain in diversity, but its accuracy significantly declines. This confirms that semantic signals alone are insufficient for relevance and must complement, not replace, behavioral data.

Motivation for Alignment These findings support our hypothesis: collaborative models offer precision but limited diversity, while LLMs can enhance novelty but lack relevance and efficiency. This motivates our alignment-based model, which combines the strengths of both to improve long-tail recommendation while remaining practical for real-world use.

4.3 Implemented extensions

To bridge the gap between collaborative filtering and language-based models, we implement an alignment-based recommendation system inspired by A-LLMRec (Kim et al., 2024). Our implementation largely follows the original architecture but adopts a simplified and modular setup for ease of experimentation, as described in Section 3.1.

The extension retains A-LLMRec’s two-stage alignment strategy, enabling us to evaluate whether mapping collaborative embeddings into the LLM’s semantic space improves long-tail generalization without compromising relevance for popular items. In addition, we explore the potential of using these aligned representations to enhance candidate selection—particularly for long-tail items that are often underrepresented in traditional retrieval pipelines.

5 Results and Analysis

5.1 Train and Validation Results

The plots of the train and validation results can be found in Appendix 9.3. The total loss shows a steep decline in the early training steps and plateaus which shows that the model converges successfully. The text reconstruction loss drops sharply and this is expected because the SBERT embeddings are relatively stable and well structured which allows the projection MLP to learn the reconstruction easily.

The item reconstruction loss has a significant downward trend as well which shows that the MLP is able to recover original item embeddings from the latent space reasonably well. However, it is higher than the text reconstruction loss which is expected due to the less semantically structured nature of learned recommender embeddings.

The matching loss and BPR loss has a lot of noise. For the matching loss, one possibility for the noise would be that the alignment objective depends heavily on the diversity of text descriptions and how consistently the two modalities represent similar semantics. There is still a general downward slope. The BPR loss is the noisiest and this is possible due to the pointwise BCE used on a varying composition of the batch. The learning still has a downwards trend. Due to time and resource limits, we were not able to train the model for more epochs and this is left as future work. We were still able to obtain good results with a lower number of epochs.

5.2 Final Evaluation Results

Our final evaluation compares the performance of A-LLMRec against the best-performing collaborative filtering model, XDeepFM, and the GPT-2 language model baseline, using the setting where item metadata is available (Table 3). These results demonstrate the effectiveness of alignment-based modeling in combining the strengths of both paradigms.

Model	Computation		Long-Tail Exposure			Accuracy	
	Size (M)	Inf. Time (ms)	Coverage	Diversity	Avg. Ranked Bin	MRR	Hit@10
XDeepFM	0.0412	0.1935	1.0000	0.0761	3.7669	0.0648	0.1654
GPT-2	137	342,390	1.0000	0.0853	3.8433	0.0459	0.0833
A-LLMRec	168	12.500	0.9902	0.0812	3.9335	0.1787	0.5908

Table 3: Final Results on Amazon Reviews (Movies & TV). We report metrics across computation, long-tail exposure, and accuracy. The inference of our implementation of A-LLMRec was ran on a g5.2xlarge AWS instance with 101 candidate items.

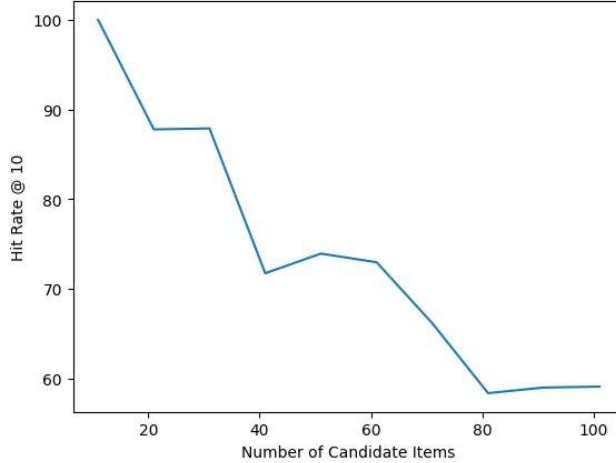


Figure 3: Hit@10 against number of candidate items for our implementation of A-LLMRec

A-LLMRec achieves the best overall balance across accuracy, long-tail exposure, and computational efficiency. It reaches a Hit@10 of 0.5908, more than three times higher than the strongest collaborative baseline, and an MRR of 0.1787. At the same time, it maintains solid diversity (0.0812) and an average ranked bin of 3.93, indicating that it successfully surfaces less popular items without compromising recommendation relevance.

In comparison, **GPT-2 performs well only on diversity** (0.0853) but fails to rank relevant items effectively, with Hit@10 at 0.0833. Its inference time is prohibitively high at over 340,000 milliseconds, making it impractical for real-time deployment. **XDeepFM, on the other hand, maintains decent ranking accuracy** but remains biased toward popular items, with an average rank bin of 3.77. Although its coverage is high, the recommendations tend to be redundant and concentrated on head items.

Overall, the advantage of A-LLMRec lies in its ability to align collaborative and semantic information into a shared space, allowing the model to incorporate behavioral signals and metadata jointly. This results in better personalization and broader catalog exposure while keeping inference time low at 12.5 milliseconds, a significant improvement over GPT-2.

5.3 Further Discussions

Sensitivity to Candidate Pool Size As shown in Figure 3, increasing the number of candidate items leads to a decline in Hit@10, reflecting the growing challenge of accurate ranking in larger candidate pools. Nevertheless, A-LLMRec consistently outperforms all baselines even at 101 candidates, demonstrating strong robustness in practical scenarios. This underscores the promise of alignment-based approaches, which allow us to leverage stronger language models with large context windows to handle richer candidate sets and support more diverse recommendation outputs.

Significance and Risk The success of the alignment approach depends on the quality of the learned mappings between collaborative and semantic spaces. Our evaluation uses only 30% of the dataset,

which may not fully reflect its scalability or robustness. In real-world settings with more data and diverse user behavior, new interactions between components could emerge. Larger-scale experiments are needed to confirm these results.

6 Future Directions

A key limitation of our current approach is the use of long-tail exposure metrics as a proxy for actual recommendation quality. Since positive interactions with long-tail items are inherently sparse, our evaluation focuses on whether these items are surfaced—not whether they align with user intent. In practice, not all users seek novelty; many prefer personalized and familiar content. This raises a critical question: how can models balance personalization with long-tail diversity?

One possible extension is to incorporate user-level diversity preferences. For example, a generalist–specialist score could adapt recommendation breadth: generalists would receive more diverse suggestions, while specialists would see more focused results.

Our framework also has potential for addressing user cold-start scenarios. Similar to how we target sparse-item cases, new users could be clustered based on profile metadata, with recommendations bootstrapped from behaviorally similar users.

Lastly, our alignment architecture is LLM-agnostic. Future work could explore scaling to larger or domain-specific language models, and more systematically evaluate trade-offs between model complexity, performance, and inference efficiency.

7 Conclusion

This project set out to systematically evaluate the potential of Large Language Models (LLMs) for improving long-tail item recommendation. We first benchmarked traditional collaborative filtering models and LLM-based approaches separately, showing that each excels in different areas. Collaborative models perform well on accuracy but favor popular items, while language models can improve diversity but struggle with precision and efficiency.

To address this, we implemented an alignment-based model that integrates collaborative signals with semantic representations. Our version of A-LLMRec outperformed all baselines in both accuracy and long-tail exposure, while maintaining practical inference speed. It successfully surfaced less popular items without compromising ranking relevance or deployability.

These results support the value of combining collaborative and semantic knowledge through alignment techniques. While further work is needed to scale the model and personalize recommendations, our findings show that LLMs can meaningfully enhance recommender systems when carefully integrated.

8 Administrative Details

Breakdown of team roles:

- Allyne: Responsible for data preparation and implementation of baseline models.
- Wei Bin: Responsible for training A-LLMRec and running inference.
- Xun Yi: Responsible for evaluating the long-tail item task.
- Yogesh: Responsible for training A-LLMRec and writing the inference logic.

Our working code is available at <https://github.com/allynee/LLM-LongTailRec>.

References

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-ranking. arXiv:1901.07555 [cs.IR] <https://arxiv.org/abs/1901.07555>

- Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM, 1007–1014. <https://doi.org/10.1145/3604915.3608857>
- Filippo Carnovalini, Antonio Rodà, and Geraint Wiggins. 2025. Popularity Bias in Recommender Systems: The Search for Fairness in the Long Tail. *Information* 16, 2 (19 Feb. 2025), 1–26. <https://doi.org/10.3390/info16020151> Funding Information: This research was partially funded by the Department of Information Engineering of the University of Padova under the project 201CCreative Recommendations to avoid Unfair Bottlenecks201D. Research carried out within the project 201CSISSI201D was funded by Regione Veneto for program POC 201420132020 (application ID 10449164). Publisher Copyright: © 2025 by the authors..
- Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) (*DLRS 2016*). Association for Computing Machinery, New York, NY, USA, 7–10. <https://doi.org/10.1145/2988450.2988454>
- Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. arXiv:1703.04247 [cs.IR] <https://arxiv.org/abs/1703.04247>
- Marius Kaminskis and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (Dec. 2016), 42 pages. <https://doi.org/10.1145/2926720>
- Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. arXiv:1808.09781 [cs.IR] <https://arxiv.org/abs/1808.09781>
- Sein Kim, Hongseok Kang, Seungyeon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) (*KDD '24*). Association for Computing Machinery, New York, NY, USA, 1395–1406. <https://doi.org/10.1145/3637528.3671931>
- Dominik Kowald. 2025. Investigating Popularity Bias Amplification in Recommender Systems Employed in the Entertainment Domain. arXiv:2504.04752 [cs.IR] <https://arxiv.org/abs/2504.04752>
- Xiangyang Li, Bo Chen, Lu Hou, and Ruiming TANG. 2025. CTRL: Connect Collaborative and Language Model for CTR Prediction. *ACM Trans. Recomm. Syst.* (Feb. 2025). <https://doi.org/10.1145/3713080> Just Accepted.
- Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. ACM, 1754–1763. <https://doi.org/10.1145/3219819.3220023>
- Jan Malte Lichtenberg, Alexander Buchholz, and Pola Schwöbel. 2024. Large Language Models as Recommender Systems: A Study of Popularity Bias. arXiv:2406.01285 [cs.IR] <https://arxiv.org/abs/2406.01285>
- Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024. LLM-ESR: Large Language Models Enhancement for Long-tailed Sequential Recommendation. arXiv:2405.20646 [cs.IR] <https://arxiv.org/abs/2405.20646>
- Sichun Luo, Yuxuan Yao, Bowei He, Yinya Huang, Aojun Zhou, Xinyi Zhang, Yuanzhang Xiao, Mingjie Zhan, and Linqi Song. 2024. Integrating Large Language Models into Recommendation via Mutual Augmentation and Adaptive Aggregation. arXiv:2401.13870 [cs.IR] <https://arxiv.org/abs/2401.13870>

- Aashiq Muhamed, Iman Keivanloo, Sujana Perera, James Mrazek, Yi Xu, Qi Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul M. Chilimbi. 2021. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. <https://api.semanticscholar.org/CorpusID:243844336>
- Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 188–197. <https://doi.org/10.18653/v1/D19-1018>
- Yoon-joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. *RecSys’08: Proceedings of the 2008 ACM Conference on Recommender Systems*, 11–18. <https://doi.org/10.1145/1454008.1454012>
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).
- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084 [cs.CL] <https://arxiv.org/abs/1908.10084>
- Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems (Chicago, Illinois, USA) (RecSys ’11)*. Association for Computing Machinery, New York, NY, USA, 109–116. <https://doi.org/10.1145/2043932.2043955>
- Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. 2024. CoRAL: Collaborative Retrieval-Augmented Large Language Models Improve Long-tail Recommendation. arXiv:2403.06447 [cs.IR] <https://arxiv.org/abs/2403.06447>
- Junlin Zhang, Tongwen Huang, and Zhiqi Zhang. 2019. FAT-DeepFFM: Field Attentive Deep Field-aware Factorization Machine. arXiv:1905.06336 [cs.LG] <https://arxiv.org/abs/1905.06336>
- Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. 2024. mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*. 1393–1412.
- Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. arXiv:1706.06978 [stat.ML] <https://arxiv.org/abs/1706.06978>

9 Appendix

9.1 Equations

- **Diversity**

$$\text{Diversity} = \frac{|\bigcup_{u \in U} R_u|}{|I|} \quad (\text{A.6})$$

where U is the set of users, I is the set of categories, and R_u denotes the set of recommended categories for user u .

- **Coverage**

$$\text{Coverage} = \frac{1}{|I|} \left| \bigcup_{u \in U} R(u) \right| \quad (\text{A.7})$$

When the result is 1, it means all categories are recommended at least once.

- **Mean Reciprocal Rank**

$$\text{MRR@k} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \begin{cases} \frac{1}{R_i} & \text{if relevant item appears in top-}k \\ 0 & \text{otherwise} \end{cases} \quad (\text{A.8})$$

When the result is 1, it means the relevant item is recommended first.

- **Binary Cross Entropy Loss**

$$\text{BCE}(p, y) = -[y \cdot \log(p) + (1 - y) \cdot \log(1 - p)] \quad (\text{A.9})$$

- **Mean Square Error**

$$\text{MSE}(a, b) = \frac{1}{d} \sum_{i=1}^d (a_i - b_i)^2 \quad (\text{A.10})$$

9.2 Prompt Template for GPT-2 Baseline

We use a few-shot prompting strategy to simulate the recommendation process. For each user, GPT-2 is provided with a list of previously interacted items, followed by a candidate item. The model is asked to assess whether the candidate is relevant to the user. Two versions of the prompt are used: one using only item titles, and another incorporating additional metadata fields.

Without metadata:

The user previously interacted with these movies:

1. Title: {item_title_1}
2. Title: {item_title_2}
3. Title: {item_title_3}
- ..

Is the following movie relevant to this user? Respond with only 'Yes' or 'No'.

Candidate Movie: Title: {candidate_title}

Answer:

With metadata:

The user previously interacted with these movies:

1. Title: {item_title_1} | Category: {item_category_1} | Description: {item_description_1} | Brand: {item_brand_1}
2. Title: {item_title_2} | Category: {item_category_2} | Description: {item_description_2} | Brand: {item_brand_2}
3. Title: {item_title_3} | Category: {item_category_3} | Description: {item_description_3} | Brand: {item_brand_3}
- ...

Is the following movie relevant to this user? Respond with only 'Yes' or 'No'.

Candidate Movie: Title: {candidate_title} | Category: {candidate_category} |

Description: {candidate_description} | Brand: {candidate_brand}

Answer:

9.3 Train & Validation Plots

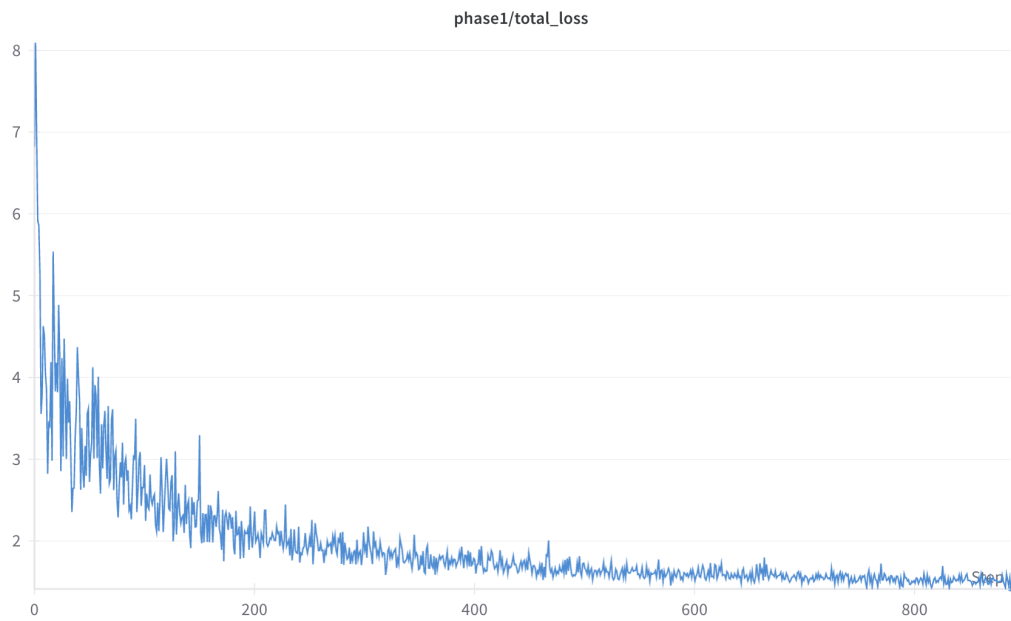


Figure 4: Phase 1 Total Loss

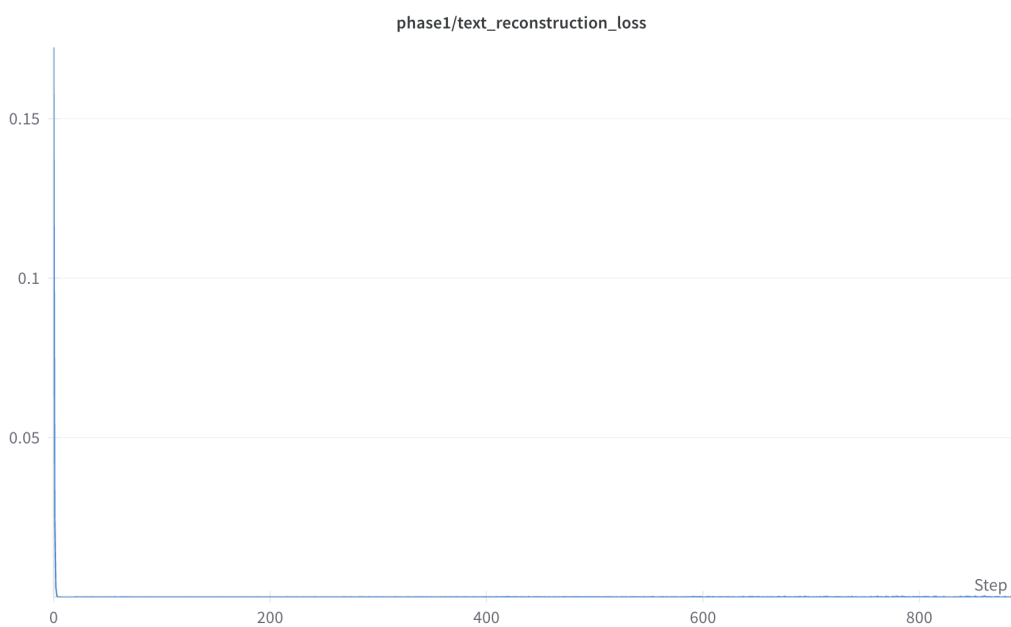


Figure 5: Phase 1 Total Loss

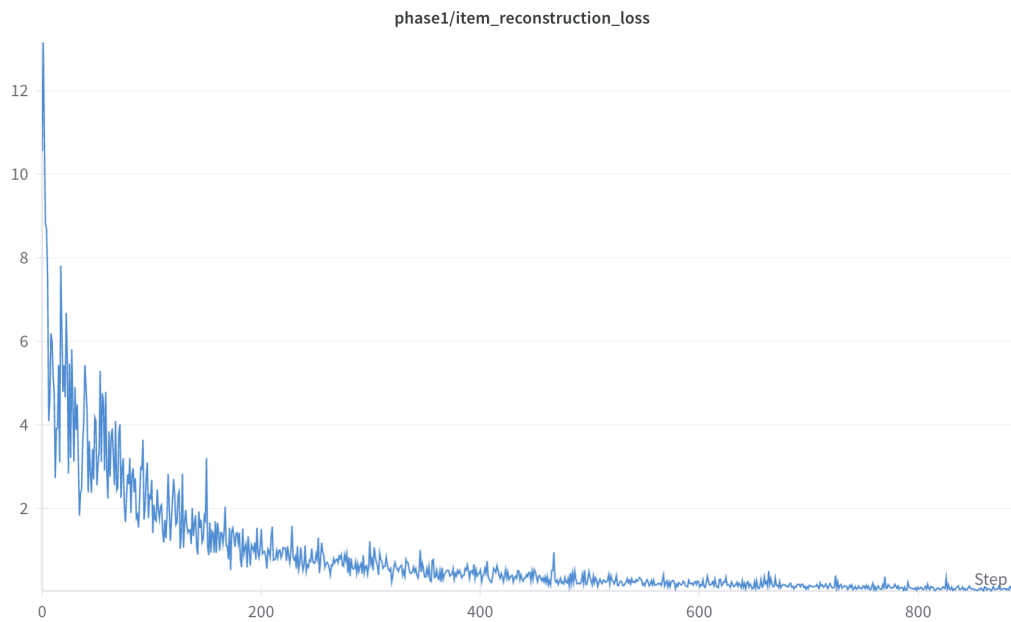


Figure 6: Phase 1 Total Loss

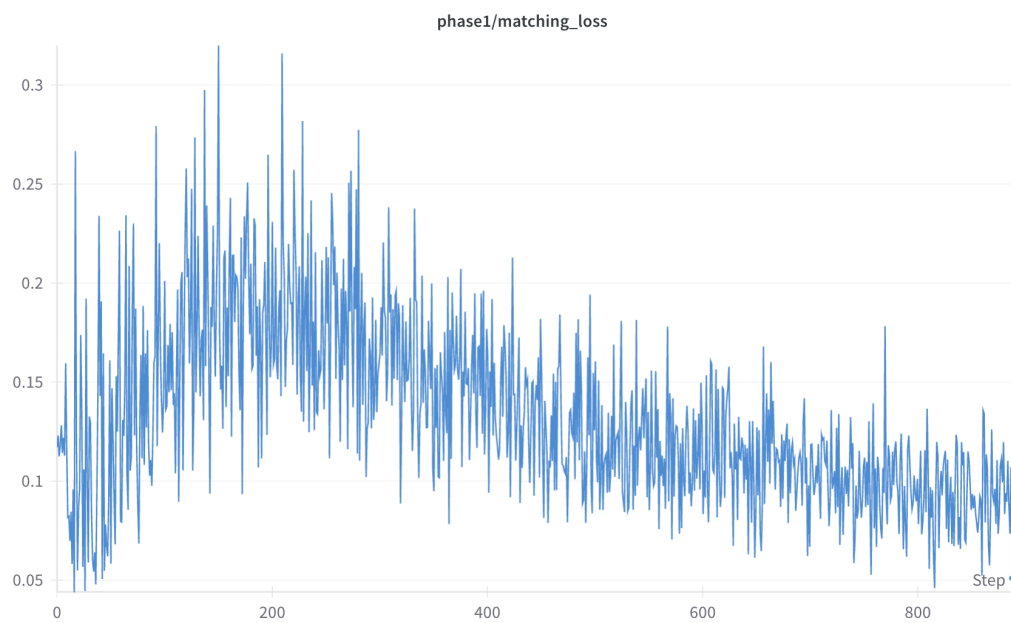


Figure 7: Phase 1 Total Loss

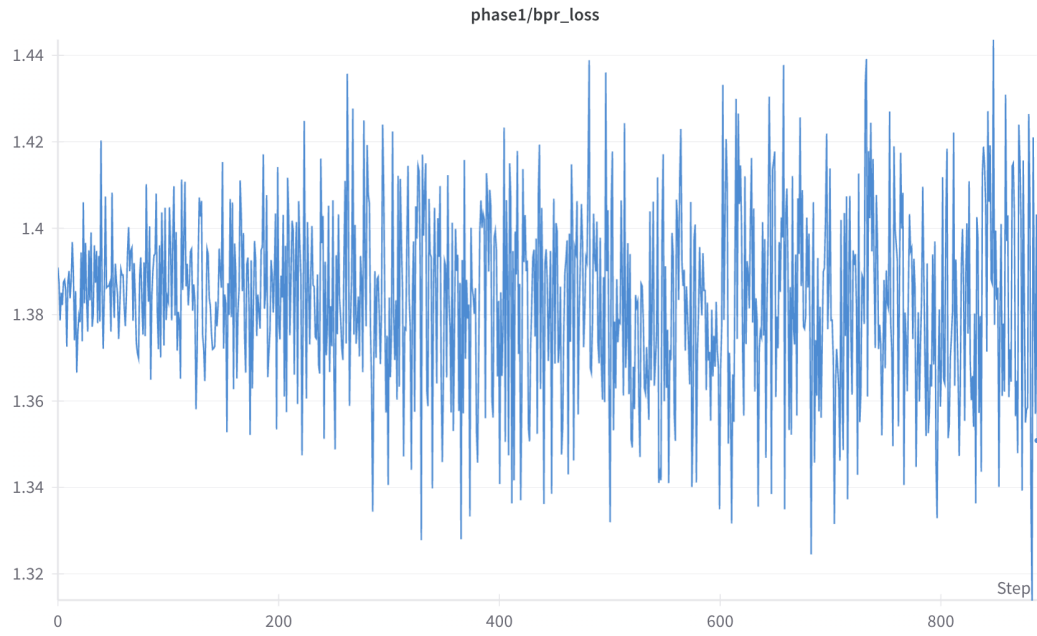


Figure 8: Phase 1 Total Loss

9.4 Varying Candidate Item Size Plots

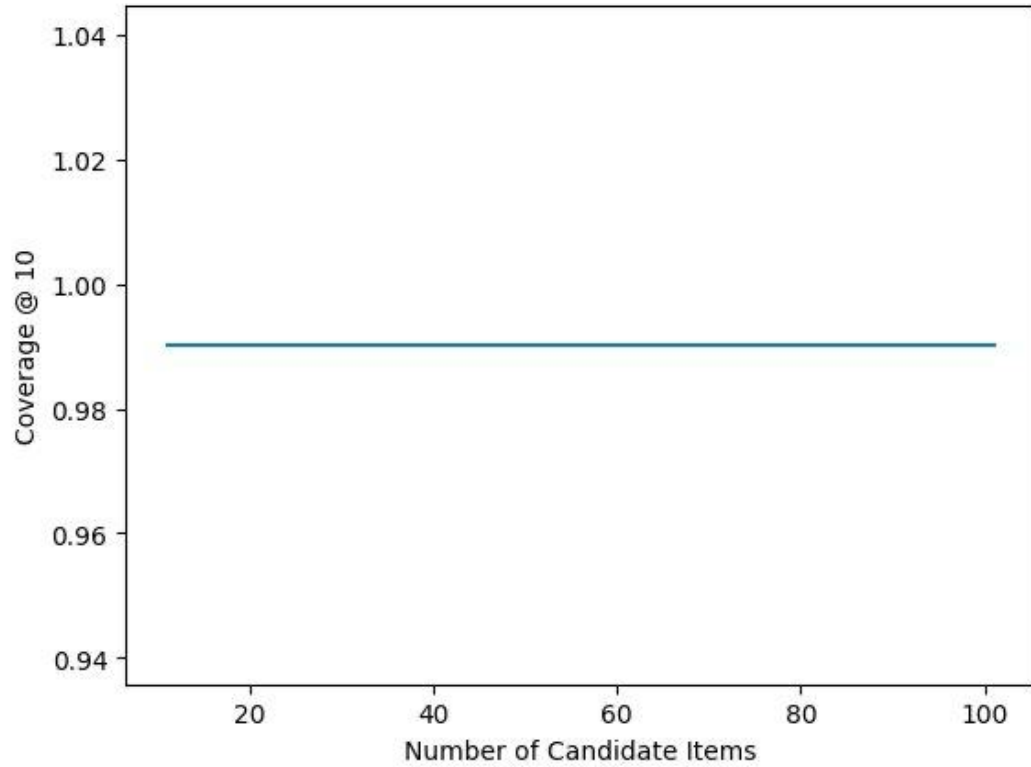


Figure 9: Coverage @ 10

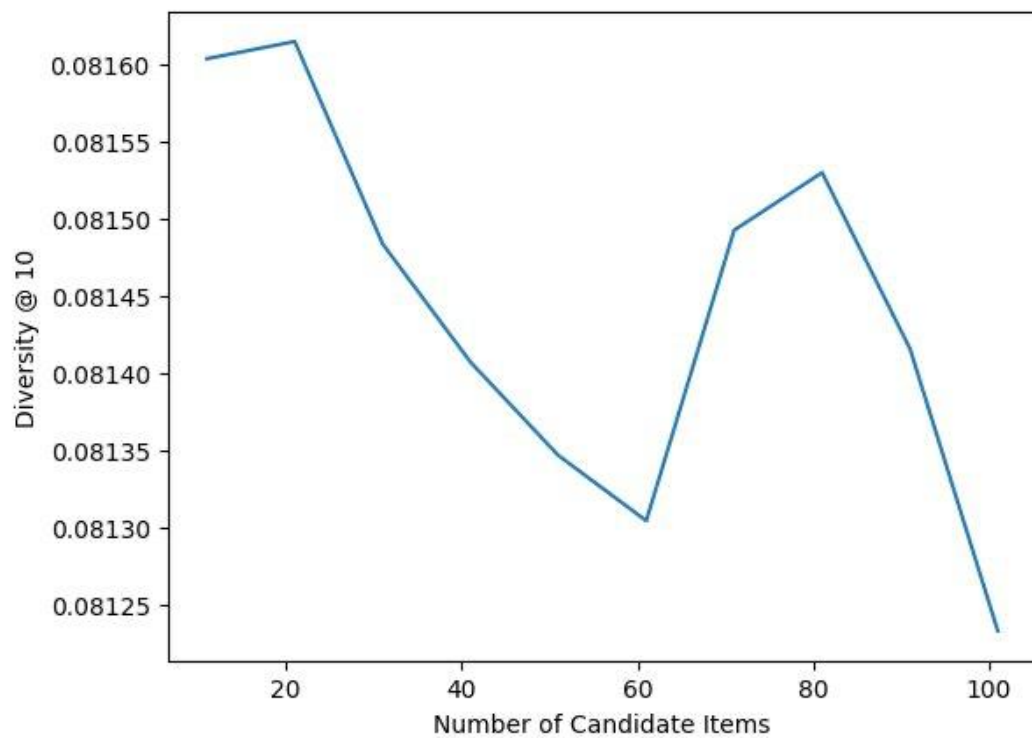


Figure 10: Diversity @ 10

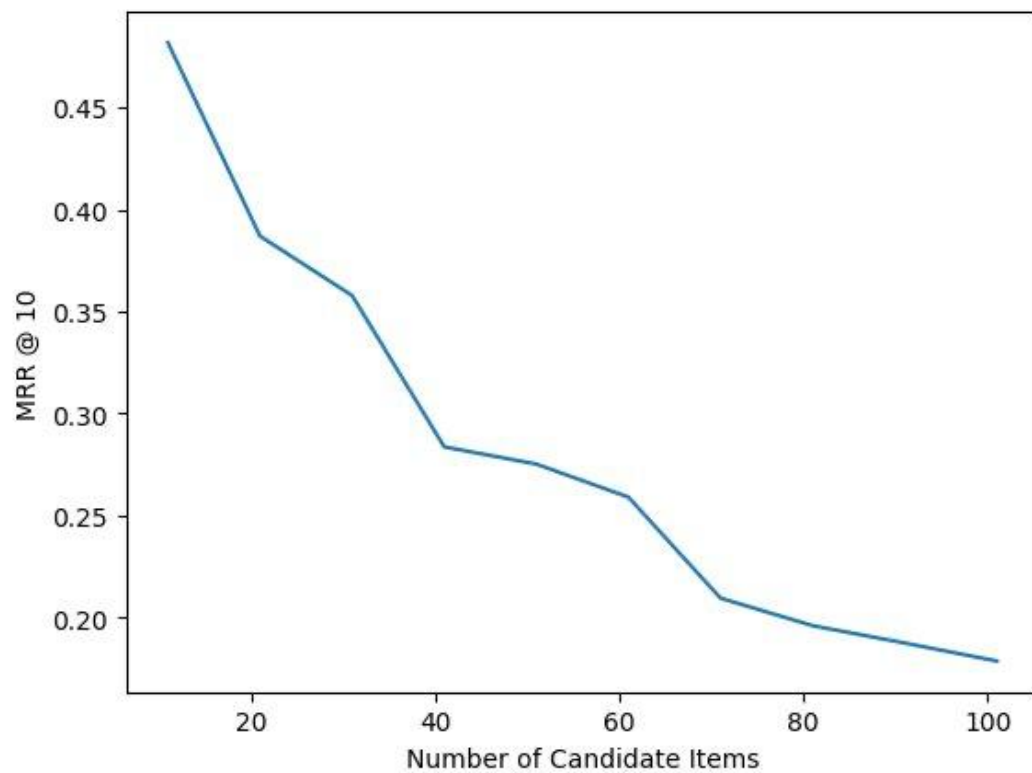


Figure 11: Mean Reciprocal Rank @ 10

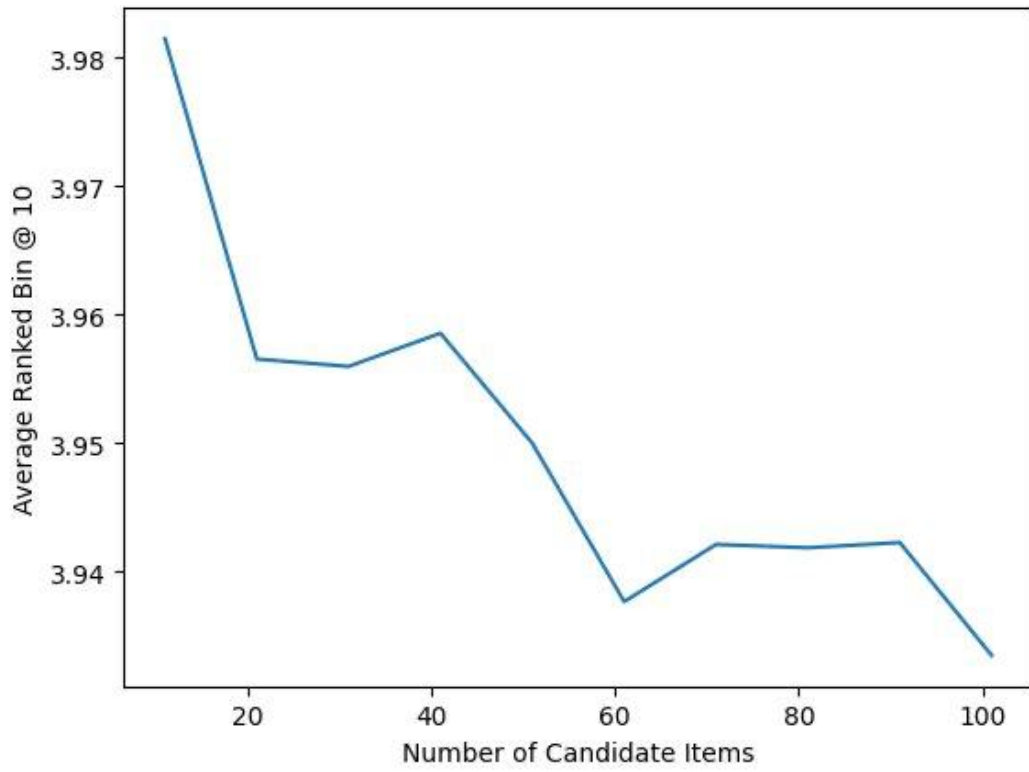


Figure 12: Average Ranked Bin @ 10

9.5 A-LLMRec Parameter Count

Layer (type:depth-idx)	Param #
A_llmrec_model	--
└RecSys: 1-1	--
└SASRec: 2-1	--
└Embedding: 3-1	(4,333,950)
└Embedding: 3-2	(2,500)
└Dropout: 3-3	--
└ModuleList: 3-4	(200)
└ModuleList: 3-5	(20,400)
└ModuleList: 3-6	(200)
└ModuleList: 3-7	(10,200)
└LayerNorm: 3-8	(100)
└two_layer_mlp: 1-2	--
└Linear: 2-2	(6,528)
└Linear: 2-3	(6,450)
└Sigmoid: 2-4	--
└MSELoss: 1-3	--
└BCEWithLogitsLoss: 1-4	--
└llm4rec: 1-5	--
└GPT2LMHeadModel: 2-5	--
└GPT2Model: 3-9	(124,443,648)
└Linear: 3-10	(38,601,216)
└Sequential: 1-6	--
└Linear: 2-6	39,168
└LayerNorm: 2-7	1,536
└LeakyReLU: 2-8	--
└Linear: 2-9	590,592
└Sequential: 1-7	--
└Linear: 2-10	99,072
└LayerNorm: 2-11	1,536
└GELU: 2-12	--
└Linear: 2-13	590,592
Total params: 168,747,888	
Trainable params: 1,322,496	
Non-trainable params: 167,425,392	

Figure 13: A-LLMRec Parameter Count