xLSTM Architecture's Feasibility For Recommendations (Draft Version)

Vivekanand

Abstract

The primary idea of Recommeder system is to present users with the most relevant items such as Music, Movies, Software, Services by learning large-scale interaction datasets. In this research paper, we will investigate the feasibility of extended Long Short-Term Memory (xLSTM) architecture for sequential recommendation tasks.

We benchmark xLSTM against strong existing baselines, including Transformer-based models (such as BERT4Rec, SAS4Rec) and recurrent alternatives, on five public datasets: MovieLens-100K, MovieLens-1M, MovieLens-10M, MUSIK4all, and Amazon Software. We also evaluate and compare models using top-k ranking metrics (Recall@10, MRR@10, NDCG@10)with training time.

Our results show that, xLSTM attains competitive accuracy relative to Transformer baselines, particularly on medium and large-scale datasets, and while offering favorable memory scaling and stable training. On smaller datasets, performance gaps narrow or favor Transformers models, highlighting trade-offs between capacity and data sequence settings.

Overall, our results indicate that xLSTM is a strong sequence model at scale competitive with Transformer baselines on larger datasets like ML-1M/10M, memory-efficient and training-stable, yet less effective on the smallest dataset (ML-100K). Finally, these results and findings will point to a practical path for deploying xLSTM in modern large-scale recommender systems.

1 Motivation

The rapid growth of the user interaction data has created an unprecedented demand to create efficient recommender systems that must act faster, accurate and computationally efficient. Despite there are several advancements in the sequential recommendations research which are primarily based on existing architectures like LSTM and Transformer-based, but still the scalability, long-term dependency and computational problems remain the key challenges. This motivates to the exploration of the latest and enhanced architectures like xLSTM and to address the existing limitations and to check the feasibility of sequential recommendation research for the large scale interaction datasets.

The critical subdomain of recommenders is sequential recommendations, which models user-item interactions over the period of time, capturing temporal patterns and evolving user preferences. Recent advancements in this research area have leveraged deep learning architectures, including Recurrent Neural Networks (RNN) **?**, Long Short Term Memory (LSTM)**?**, and other recent architectures like BERT4Rec Sun et al. [2019] and SASRec Kang and McAuley [2018], which primarily adopt Transformer-based self-attention mechanisms for sequence modeling.

In this thesis, we will investigate xLSTM, primarily mLSTM Beck et al. [2024], an enhanced variant of LSTM that incorporates architectural improvements including bidirectional processing, attention integration, and refined gating mechanisms. We will also benchmark xLSTM against classical and state-of-the-art sequential recommenders using datasets like MovieLens and other temporal interaction datasets. Our final results shows that xLSTM demonstrates notable improvements due to

Preprint. Under review.

its hybrid design like mLSTM and sLSTM that blends recurrent memory and attention mechanisms, which we will leverage further.

1.1 Existing Architecture Overview

In earlier recommendation prediction tasks, Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM) were commonly used to predict the end user preferences and provide us with personalized suggestions. Roy and Dutta [2022], Author [2023a]. Classical methods such as collaborative filtering and content-based filtering will have also evolved to incorporate sequential modeling, capturing dynamic user behavior over the period of time Author [2024a].

Among the deep learning approaches, Long Short-Term Memory (LSTM) network remains a widely used architecture for modeling sequential interactions due to its ability that it can preserve long-term dependencies and manage temporal dynamics effectively utilizing its gating mechanisms.

Despite its success in implementation, LSTM also suffered from several well-known limitations, such as: (1) its inability to revise storage decisions once made - which means once the LSTM decides whether to keep or forget some information, it cannot go back and change that decision later (2) the need to compress all contextual information into scalar cell states - which means that all the memory must be compressed into single vector, a fixed size memory, and (3) limited parallelizability due to its recurrent memory mixing Beck et al. [2024] - it can process one step at a time and hard to perform computations in parallel. These limitations have led to development of enhanced variants, such as xLSTM, which will introduces architectural refinements like attention mechanisms, bidirectional memory, and improved gating functions to better handle large scale sequential recommendation tasks.

1.2 General Classification of Recommender Systems

According to the RecBole framework Author [2020a], recommender systems can be broadly classified into four major categories based on the nature of data used and task formulation:

1. **General Recommendation (GR)**: These models rely solely on user-item interaction data, typically implicit feedback. Implicit feedback will contains the datasets which indirectly indicate user preferences such as clicks, Add-to-cart events, purchase, Time spent, frequency etc. They are evaluated through top-N recommendation tasks and primarily align with collaborative filtering approaches.
2. **Content-Aware Recommendation**: These models incorporate additional side information such as user or item features. They are often applied in click-through rate (CTR) prediction tasks, using explicit feedback and binary classification evaluation. Its feature based, often go beyond raw user-item interactions about users, items or context.
3. **Sequential Recommendation (SR)**: SR focuses on next-item prediction by modeling the temporal ordering of user interactions. These models utilize sequential data to capture evolving user preferences. Session-based recommendations are generally included in this category and this is our primary research focus.
4. **Knowledge-Based Recommendation**: These methods leverage external knowledge graphs to enhance general or sequential recommendations by providing semantic or structural context beyond user-item interactions.

Broadly, recommendation systems can also be categorized into two paradigms: Collaborative Filtering, which models similarities between users or items based on interaction histories, and Content-Based Filtering, which relies on item attributes or metadata to make predictions Author [2024a, 2023a]

1.3 Problem Statement

Despite the advancements brought by deep learning in improving recommendation accuracy, several critical challenges persist in modeling user-item sequences at scale. These challenges include the cold-start problem, data sparsity, and the need for robust long-term dependency tracking across user interactions Noorian et al. [2024], Roy and Dutta [2022].

Recurrent architectures such as LSTM, though effective in capturing temporal dependencies, suffer from limited scalability and are inherently sequential, making them less efficient for large-scale

recommendation environments Yang and Esquivel [2024], Ahmadian Yazdi et al. [2024]. In contrast, Transformer-based models like BERT4Rec and SASRec offer greater parallelism and improved performance through self-attention mechanisms but often come with high computational costs and memory requirements Sun et al. [2019], Kang and McAuley [2018].

This work investigates whether the recently proposed xLSTM architecture Beck et al. [2024], which integrates bidirectional memory, attention mechanisms, and enhanced gating structures, can serve as a middle ground—achieving scalability, computational efficiency, and high accuracy in sequential recommendation tasks.

1.4 Research Questions RQ and Objectives

This report aims to answer the following key questions:

• RQ1: How does xLSTM's performance scale with dataset size compared to established architectures like BERT4Rec?

• RQ2: Does increasing sequence length and embedding size lead to measurable performance gains in sequential recommenders?

• RQ3: What trade-offs exist between recommendation accuracy and computational cost as sequence length and model complexity increase?

RQ4: Embedding Saturation and Utilization: Are larger embeddings really helping the model learn better user/item relationships, or are they underutilized?

The primary objective is to evaluate the effectiveness of the xLSTM model across multiple datasets and benchmark it against state-of-the-art baselines using established ranking metrics.

1.5 Research Contributions of This Work

This thesis makes the following key contributions to the field of sequential recommender systems:

Implementation and Evaluation of xLSTM: We present a novel adaptation and empirical evaluation of the xLSTM architecture for sequential recommendation tasks, using benchmark datasets such as MovieLens for reproducibility and comparative analysis.

Comprehensive Model Benchmarking: We conduct a rigorous comparison between xLSTM and several baseline models, including Transformer-based (e.g., BERT4Rec, SASRec) and RNN-based (e.g., standard LSTM) architectures. The evaluation spans both quantitative (e.g., HR@K, NDCG@K) and qualitative performance metrics.

Architectural Integration: Our implementation incorporates architectural innovations such as attention mechanisms, memory-efficient recurrent kernels, and chunkwise sequence modeling, aiming to balance performance with computational efficiency.

Addressing Real-World Challenges: We offer empirical insights into mitigating common real-world limitations in recommender systems, including the cold-start problem, interaction sparsity, and long-sequence dependency modeling.

1.6 Market and Industry Relevance

The global recommender system market driven by personalization demands in retail, media, and fintech. Efficient models like xLSTM could offer industrial-grade scalability while maintaining personalization quality and can be computationally efficient. Adoption of such architectures can improve click-through rates, user retention, and recommendation diversity, directly impacting KPIs across sectors like e-commerce, content streaming, financial services, and smart energy platforms.

2 Existing Model Architectures For Sequential Recommenders

2.1 Singular Value Decomposition (SVD)

SVD (matrix factorization) compresses the user–item matrix into small "latent factors." Users and items get vectors; recommendations come from matching these vectors. It scales well and works great for steady preferences, but it ignores sequence dynamics and has trouble with cold-start users/items.

SVD breaks the large user–item matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ into three smaller pieces and keeps only the most informative parts:

$$\mathbf{R} \;\approx\; \mathbf{U}_k \, \Sigma_k \, \mathbf{V}_k^\top ,$$

so each user and each item get a k-dimensional vector that captures their main tastes and properties **?**. We usually work with matrix-factorization form, scoring user u and item i by

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u^\top \mathbf{q}_i,$$

and learn $\mathbf{p}_u, \mathbf{q}_i$ by minimizing a regularized squared loss

$$\min_{\{\mathbf{p},\mathbf{q},b\}} \sum_{(u,i)\in\mathcal{D}} \left(r_{ui} - \mu - b_u - b_i - \mathbf{p}_u^\top \mathbf{q}_i\right)^2 + \lambda \left(\sum_u \|\mathbf{p}_u\|^2 + \sum_i \|\mathbf{q}_i\|^2 + b_u^2 + b_i^2 \right),$$

which gives robust, low-rank representations and strong top-k recommendations **?**. With implicit logs (clicks, plays) we weight observations via confidences c_{ui} instead of using star ratings directly, which makes the model usable at web scale **?**. On its own, SVD is *static*: it explains long-term preference but ignores the order of events. To make it sequence-aware, one simple step is to let parameters drift over time (timeSVD++), e.g., time-dependent biases $b_u(t), b_i(t)$ and decays for older interactions, so recent behavior counts more **?**. Another route is to factorize *transitions*: model the probability of the next item from the last one and the user. FPMC combines MF with a first-order Markov chain and scores a candidate by user taste *and* compatibility with the last item, yielding strong next-item predictions **?**. In practice we also train on sliding windows or apply exponential decay before factorization to emphasize recency. Because it is fast, stable, and easy to tune, SVD serves as a solid baseline for sequential tasks, a candidate generator for heavier models, and a source of clean embeddings to initialize LSTMs/Transformers.

2.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks read a user's history in order (one step after another) to predict the next item. They capture short-term trends (what the user is into right now), but training can be unstable and slow on long sequences.

Recurrent Neural Networks (RNN). A (vanilla) RNN updates its hidden state by combining the current input with the previous state, then predicts the next item; it is simple and fast but struggles with long-range dependencies due to vanishing gradients **????**.

$\mathrm{h}_t = \phi(W_x x_t \;+\; W_h h_{t-1} \;+\; b_h)\,,$

$\mathrm{s}_{t+1} = E^\top h_t,$
$\hat{y}_{t+1} = \mathrm{softmax}(s_{t+1})\,,$

L = - $\sum_t \log \hat{y}_t[y_t]$ $\quad (cross - entropyoverthetruenextitem\mathrm{y}_t)$,

Where x_t is the current input (item embedding), h_t the hidden state, $\phi(\cdot)$ a pointwise nonlinearity (e.g., $\tanh$ or ReLU), E the shared item-embedding/output matrix, and W_x, W_h, b_h learned parameters. Training uses backpropagation through time (BPTT) **?**.

2.3 Long Short Term Memory (LSTM)

LSTM improves on RNNs with gates that help remember important past interactions for longer. It's a strong sequential baseline used widely before Transformers, offering better long-range memory than vanilla RNNs. However, it can be heavier to train and may still lag behind attention models on very long histories. LSTMs extend RNNs with a gated memory cell that preserves information over long

spans, mitigating vanishing gradients and enabling next-item predictors to remember earlier parts of a user's sequence **?**. LSTMs add a gated memory cell to overcome vanishing gradients and keep long-range context in user sequences **??**.

$$
\begin{aligned}
\mathrm{i}_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i)\,, \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f)\,, \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o)\,, \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c)\,.
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{c}_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\
h_t &= o_t \odot \tanh(c_t).
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{s}_{t+1} &= E^\top h_t, \\
\hat{y}_{t+1} &= \mathrm{softmax}(s_{t+1})\,.
\end{aligned}
$$

Where x_t is the current input (item), h_t the hidden state, c_t the cell state, σ the logistic function, $\odot$ element-wise product, E shared item embeddings, and W_*, U_*, b_* are learned parameters. This head produces next-item probabilities for ranking.

2.4 Gated Recurrent Unit for Recommendation (GRU4Rec)

GRU4Rec Hidasi et al. [2016] is a pioneering model in session-based recommendation, utilizing Gated Recurrent Units (GRUs) to model user behavior over sequences of interactions. GRUs efficiently capture short and mid-range temporal dependencies by maintaining hidden state updates through gated mechanisms, which control information flow and mitigate vanishing gradient issues. Compared to traditional LSTMs, GRUs offer a more compact architecture with fewer parameters, leading to faster training and inference times. It captures temporal dependencies efficiently with fewer parameters than LSTM, making it faster and suitable for shorter sequences. GRU4Rec models session dynamics and adapts well to implicit feedback settings, making it a strong lightweight baseline.

GRU4Rec is particularly effective for modeling sessions where user-item interactions are sparse, sequential, and implicit in nature. The model processes sequences in an autoregressive manner, predicting the next item based on prior interactions. Despite its simplicity, GRU4Rec serves as a strong baseline for session-based recommendation tasks, though it may struggle to capture complex, long-range dependencies compared to more recent attention-based models.

2.5 Self-Attentive Sequential Recommendation (SASRec)

Kang and McAuley [2018] applies Transformer-style self-attention to the recommendation domain. It models user-item sequences without recurrence, allowing for full parallelism and effective long-range dependency capture. The model uses positional encodings and attention weights to identify the most relevant past items when predicting the next interaction, offering high accuracy and scalability.

2.6 Bidirectional Encoder Representations for Recommendation (BERT4Rec)

BERT4Rec adopts the BERT (Bidirectional Encoder Representations from Transformers) Sun et al. [2019] architecture for sequential recommendation. It treats user interaction history as a sequence and uses a masked item prediction task to learn bidirectional dependencies. Unlike SASRec, which processes in a left-to-right fashion, BERT4Rec learns from both past and future items, improving its ability to model complex sequence semantics.

3 Existing Literature Review and Architectures For Recommenders

Review of Sequential Recommender Architectures and Opportunities for xLSTM Integration:

Sequential recommender systems have evolved significantly, with early models relying on Recurrent Neural Networks (RNNs) such as LSTM to capture temporal dependencies in user-item interactions. For example, hybrid architectures combining BERT for semantic encoding and LSTM for sequence modeling have been used in point-of-interest recommendations within the tourism domain, addressing dynamic user behaviors and sparse sequential data [Noorian et al., 2024]. Similarly, BiLSTM with

attention has been explored in educational recommendation contexts, focusing on modeling shifting learning patterns, but struggling with long-sequence scalability [Ahmadian Yazdi et al., 2024]. In the business intelligence domain, hierarchical time-aware LSTM frameworks such as DynaPR integrate product attributes and user behavior but remain limited by sparse data and evolving interests [Yang and Esquivel, 2024].

Session-based recommenders have applied attention-driven LSTMs to better capture recent interactions, yet they face challenges with long-session dependencies and real-time inference [Kumar and Kumar, 2024]. Generative retrieval-based systems like TIGER introduce Transformer architectures but fall short in handling sequential sparsity and long-term dependencies [Author, 2024b]. Likewise, deep Transformer-based models such as BERT and RoBERTa have been leveraged for textual review recommendation but lack explicit temporal modeling [Author, 2023b]. Recent research also examines the use of large language models (LLMs) like GPT and T5 in multi-domain recommendations, although they present difficulties in temporal adaptability and cross-domain scalability [Author, 2024a].

These limitations suggest a strong case for incorporating xLSTM [Beck et al., 2024], which offers enhanced gating mechanisms, bidirectional memory flow, and built-in attention layers. By addressing core challenges such as long-term dependency modeling, data sparsity, and cross-domain adaptability, xLSTM presents a unified architecture capable of improving performance across diverse sequential recommendation scenarios.

3.1 Sequential Neural Recommendation Using BERT and LSTM

Noorian et al. [Noorian et al., 2024] proposed a hybrid sequential recommender system for the tourism industry, leveraging BERT for semantic encoding of contextual and demographic user data, and LSTM for modeling sequential behavior in point-of-interest (POI) recommendations. The model was evaluated on datasets such as Yelp and TripAdvisor, aiming to address challenges related to dynamic user behavior and data sparsity. While LSTM played a key role in capturing temporal dependencies, the approach faced limitations in retaining long-term context and exhibited inefficiencies in real-time adaptability due to memory bottlenecks and sequential processing constraints.

3.2 Sequential Neural Recommendation System Exploiting BERT

In this sequential RS paper, Noorian et al. [Noorian et al., 2024] introduced a hybrid recommendation architecture tailored to the tourism industry, integrating user demographic, contextual, and geo-tagged data for point-of-interest (POI) suggestions. Their system combines BERT for semantic representation with LSTM for sequential modeling, enabling the system to account for both static content and evolving user behaviors. While effective at capturing short-term dependencies, the architecture encounters limitations in modeling long-range temporal patterns and suffers from computational inefficiencies due to the sequential nature of LSTM processing. Evaluations on datasets like Yelp and TripAdvisor highlighted challenges in memory retention and real-time adaptability, particularly in sparse interaction scenarios.

3.3 Dynamic Educational Recommender System Based on Improved LSTM

In this dynamic RS LSTM based paper, Ahmadian Yazdi et al. [Ahmadian Yazdi et al., 2024] proposed an educational recommender system that utilizes a BiLSTM architecture enhanced with an attention mechanism to model evolving student preferences. The system focuses on addressing the cold-start problem and adapting to changing learning patterns in digital education environments. Evaluated using datasets such as the Open University Learning Analytics, the model demonstrates effective handling of short-term user behavior. However, it encounters scalability limitations when dealing with long interaction sequences and struggles to capture complex temporal dependencies across diverse learning sessions.

3.4 Session-Based Recommendations with Attention-Driven LSTM

Kumar and Kumar [Kumar and Kumar, 2024] proposed a session-based recommender system that leverages an attention-enhanced LSTM architecture to model user intents during short-term browsing sessions. This approach emphasizes recent user interactions, using attention mechanisms

to prioritize temporally proximate events, while LSTM retains the sequential dynamics of session-specific behavior. Although effective in capturing immediate preferences, the architecture exhibits limitations in handling long-term session dependencies and incurs computational overhead, making it less suitable for real-time and large-scale deployment scenarios.

3.5 Time-Aware LSTM Neural Networks for Dynamic Personalized Recommendation in Business Intelligence

Yang and Esquivel [Yang and Esquivel, 2024] proposed DynaPR, a time-aware hierarchical LSTM framework designed to model evolving user preferences in business intelligence scenarios. The architecture integrates product attributes with temporal context into a unified embedding space, enabling the capture of both short-term and long-term behavioral patterns. While the hierarchical LSTM layers offer improvements in temporal modeling, the system still encounters challenges in handling sparse user-product interactions, dynamically shifting interests, and insufficient attribute fusion across time.

3.6 Recommender Systems with Generative Retrieval

Recent developments in generative retrieval frameworks, such as TIGER (Transformer-based Generative Retrieval), aim to improve recommendation quality by generating user-item representations using semantic IDs and Transformer-based architectures [Author, 2024b]. These models offer advantages in capturing high-level semantic similarity but often suffer from limitations in encoding sequential dependencies and handling sparse or rare item interactions. While LSTM modules have been explored to mitigate these issues, their performance remains constrained by their limited capacity to model long-term temporal dependencies, especially in low-frequency item scenarios.

3.7 Exploiting Deep Transformer Models in Textual Review-Based Recommender Systems

Transformer models such as BERT and RoBERTa have been widely adopted for review-based recommender systems due to their strong semantic encoding capabilities [Author, 2023b]. These models excel at capturing contextual information from user-generated text, which enhances item profiling and user preference modeling. However, they lack an inherent mechanism for modeling sequential dependencies over time, which is critical in dynamic recommendation settings. To address this gap, LSTM components are often introduced to capture temporal relationships between reviews. Yet, these hybrid models encounter limitations when applied to sparse datasets or cross-domain recommendation scenarios, as traditional LSTM architectures struggle with long-range dependencies and adaptation to heterogeneous data.

3.8 Exploring the Impact of Large Language Models on Recommender Systems

The integration of large language models (LLMs) such as GPT and T5 into recommender systems has gained attention for their strong capabilities in understanding contextual language and generating content-aware recommendations [Author, 2024a]. These models have shown promise in capturing nuanced user preferences across multiple domains. However, they often lack robust mechanisms for modeling long-range sequential dependencies and struggle with temporal personalization, especially in cold-start and sparse interaction scenarios. Standard LSTM components, when used in hybrid architectures, further compound these limitations due to scalability issues and inadequate memory retention across domains.

4 Methodology

This section outlines the experimental design, data preparation procedures, model configurations, and evaluation protocols used to investigate the performance of the xLSTM model in sequential recommendation tasks.

4.1 Datasets

We conducted experiments on four benchmark datasets widely used in the recommender systems community:

- **MovieLens-100K, 1M, and 10M:** Publicly available under the GroupLens license Harper and Konstan [2015], enabling benchmarking under both sparse and dense user–item conditions.
- **MUSIK4all:** Large-scale music interaction dataset (228M events, 119k users) with play counts and timestamps. Released by Johannes Kepler University Linz for music recommendation research Schedl et al. [2022].
- **Amazon Product Data (Software subset):** Publicly available review and rating data capturing millions of user–item interactions. Provides timestamps and metadata for collaborative and content-based recommendation benchmarking. **?**Ni et al. [2019].

Table 1: Experimental datasets.

Dataset	Users	Items	Interactions	Avg. len.	Sparsity
ML-100K	943	1,682	100,000	106.05	0.9369
ML-1M	6,040	3,416	999,611	165.49	0.9515
ML-10M	71,567	10,681	10,000,054	139.74	0.9869
Music For All	4584	51,291	2,533,266	552.6	0.9991
Amazon SW	3,75,147	21,663	4,59,436	-	0.9892

All datasets underwent preprocessing to ensure consistent format and indexing across experiments.

4.2 Model Pipeline

4.3 Model Selection

Three representative models were selected for benchmarking:

- **xLSTM:** Our proposed architecture incorporating chunkwise memory, gating mechanisms, and partial recurrence.
- **BERT4Rec:** A Transformer-based sequential recommender model known for its bidirectional encoding of sequence data.
- **SASRec:** A self-attention-based model using unidirectional encoding, widely adopted for next-item recommendation tasks.

4.4 Data Preparation and Splitting Protocol

To preserve chronological integrity and ensure fair evaluation, the data was split at the user level using the following procedure:

- **Training Set:** All user interactions from timestep $t = 1$ to $t = N - 2$, excluding the last two items. For each user, the training set comprises all interactions from timestep $t = 1$ to $t = N - 2$, where N is the total number of interactions for that user. This configuration ensures the model learns from a substantial portion of the user's historical behavior, allowing it to capture long-term preferences and sequence patterns while preventing exposure to future events (i.e., data leakage).
- **Validation Set:** The second-to-last item (i.e., at $t = N - 1$) used to tune model parameters and trigger early stopping. The interaction at timestep $t = N - 1$ is reserved for validation. This item is not seen during training and is used to monitor the model's ability to generalize to unseen future behavior. Performance on the validation item drives hyperparameter tuning and early stopping criteria. By placing the validation set immediately after the training window, it closely mimics real-world next-item prediction scenarios.

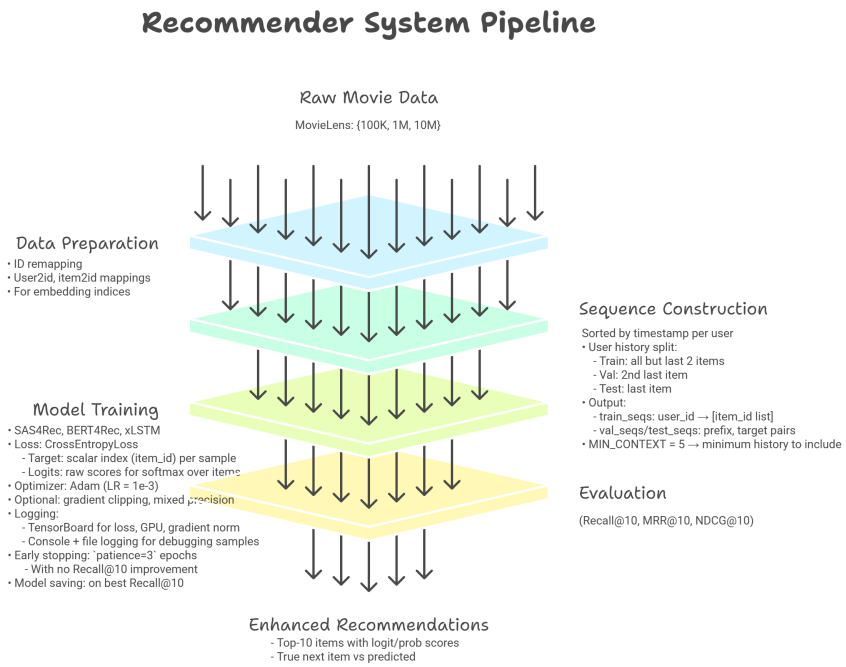

Figure 1: Model Pipeline

- **Test Set:** The final item (i.e., at $t = N$) used for final evaluation of recommendation accuracy. The final interaction at timestep $t = N$ is used as the held-out test point for evaluating model performance. This setup reflects a realistic deployment environment where the goal is to predict the next most relevant item given all historical interactions up to the test moment. The use of a single-item prediction aligns with the sequential recommendation paradigm, ensuring a fair and consistent benchmark across users and models.
- **Sequence Lengths:** The models were trained and evaluated using varying sequence lengths of 32, 64, and 128. These values represent the maximum number of past user interactions considered when predicting the next item. By experimenting with multiple sequence lengths, we assess the model's robustness and adaptability to different user history depths. Short sequences (e.g., 32) simulate scenarios with limited historical data, such as cold-start or session-based use cases. Medium and long sequences (e.g., 64 and 128) represent users with rich interaction histories, where capturing long-term dependencies becomes critical. This variation allows for analyzing how different architectures scale with memory depth and whether performance degrades or improves with longer input contexts.

4.5 Training Configuration

- **ID Remapping:** All user and item identifiers were remapped to continuous integer indices starting from zero. This remapping step ensures a compact and contiguous representation of categorical variables, which is essential for optimizing memory usage and improving the efficiency of embedding lookups during training. In large-scale datasets where raw IDs may be sparse or non-sequential (e.g., hashed strings or UUIDs), direct indexing can lead to large, sparse embedding tables with high memory overhead. Integer remapping also enables tight integration with matrix operations and GPU acceleration, which expect dense index ranges. This preprocessing step is standard in scalable recommender system pipelines and directly influences training stability and computational throughput [Rendle et al., 2020, Zhao et al., 2020].

- **Temporal Ordering:** All user interaction sequences were chronologically ordered to ensure that the model learns realistic user behavior patterns over time. This temporal consistency is crucial for sequential recommenders, which rely on the assumption that user preferences evolve as a function of prior interactions. By preserving the true order of events, the model is better equipped to capture short-term trends, session-based preferences, and long-range temporal dependencies. Chronological ordering also prevents data leakage during training and evaluation, which would otherwise inflate performance metrics and lead to unrealistic generalization [Hidasi et al., 2016, Quadrana et al., 2018].
- **Early Stopping:** To prevent overfitting and reduce unnecessary computation, training was terminated if Recall@10 on the validation set did not improve over three consecutive evaluation checkpoints (patience = 3). This strategy ensures that the model does not overoptimize on the training set, particularly important in sequential recommendation tasks where validation performance is a more reliable indicator of generalization. Early stopping also helps in controlling training time and model complexity, as suggested in empirical best practices for deep learning in recommender systems [Srivastava et al., 2014, Zhang et al., 2021].
- **Reproducibility:** Experiments were conducted using three different random seeds (42, 123, and 2023), which is essential for ensuring statistical robustness in sequential recommendation tasks. Single-run evaluations may suffer from high variance due to randomness in initialization, data ordering, or optimizer behavior. By averaging results across multiple seeds, we ensure that observed performance trends are stable, reliable, and not artifacts of random fluctuations. This practice strengthens the scientific reproducibility of our findings, as recommended in recent reproducibility studies in machine learning [Raff, 2021, Hutson, 2018].

4.6 Evaluation Design

Each model was trained and evaluated across:

- **4 datasets**
- **3 sequence lengths** (32, 64, 128)
- **3 random seeds**

This resulted in a total of **100 experimental runs**. Evaluation metrics included **Recall@10**, **MRR@10**, and **NDCG@10** to capture ranking accuracy and recommendation quality.

5 Model Architectures Compared

This study evaluates and compares the following model architectures:

xLSTM: The proposed architecture that extends traditional LSTM with chunkwise attention, bidirectional memory routing, and GPU-optimized kernels for faster training and inference.

GRU4Rec: A gated recurrent unit (GRU) based model tailored for session-based recommendation tasks. It serves as a lightweight baseline.

SASRec: A Transformer-based model that utilizes self-attention mechanisms to model user sequences without recurrence. Particularly effective for capturing long-range dependencies.

BERT4Rec: A bidirectional Transformer model that uses masked language modeling for sequential recommendation, allowing the system to learn item dependencies in both forward and backward directions.

Table 2 presents a comparative analysis of architectural and operational characteristics across LSTM, xLSTM, BERT4Rec, and SASRec. The key takeaways are:

- **Attention Mechanism:** xLSTM, BERT4Rec, and SASRec employ attention mechanisms to capture contextual relevance in sequences, unlike standard LSTM which lacks this capability Sun et al. [2019], Kang and McAuley [2018], Beck et al. [2024].

Table 2: Model architecture comparison across key dimensions.

Feature	LSTM	xLSTM	BERT4Rec	SASRec
Attention	No	Yes	Yes	Yes
Bi-directional	No	Yes	Yes	Yes
Training Stability	Medium	High	Medium	High
Inference Speed	Fast	Moderate	Slow	Moderate

- **Bidirectionality:** Both xLSTM and Transformer-based models (BERT4Rec and SASRec) support bidirectional processing, enhancing their ability to learn dependencies from both past and future contexts. In contrast, traditional LSTM is unidirectional Beck et al. [2024].
- **Training Stability:** xLSTM offers improved training stability due to its gated memory structure and regularization-friendly design. This leads to more consistent convergence relative to both LSTM and BERT4Rec Deng et al. [2024], Wu et al. [2024].
- **Inference Speed:** LSTM remains the fastest during inference due to its simplicity. xLSTM strikes a middle ground with moderate inference speed, while BERT4Rec is typically slower due to the computational overhead of deep Transformer layers Sun et al. [2019].
- **Overall Trade-off:** xLSTM provides a balance between modeling complexity, interpretability, and runtime performance, making it a compelling alternative for applications where both sequential fidelity and efficiency matter Beck et al. [2024].

5.0.1 xLSTM

Proposed xLSTM Based Approach Recommender in Details

Proposed xLSTM-Based Recommender Architecture To address the limitations of traditional LSTM architectures in sequential recommendation tasks, we adopt the recently proposed xLSTM framework Beck et al. [2024]. xLSTM introduces two principal variants: *sLSTM*, which employs scalar memory updates and memory mixing strategies, and *mLSTN*, which utilizes matrix memory representations with a covariance-based update rule, allowing for greater parallelization and expressiveness.

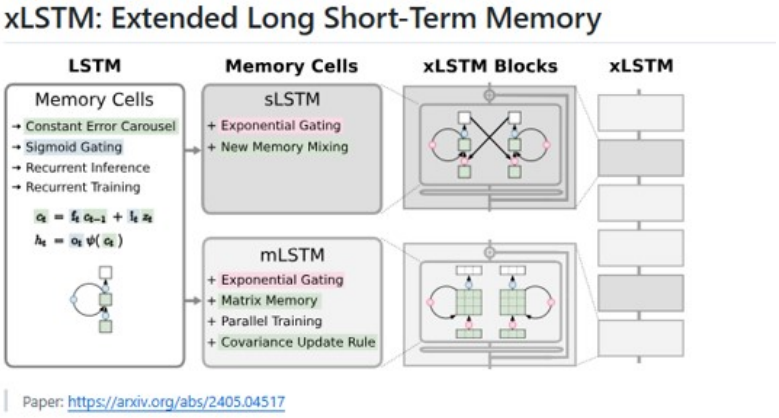

Figure 2: xLSTM Architecture

The architectural design of xLSTM enhances standard LSTM through several mechanisms that make it more suitable for modern recommender systems:

- **Extended Memory Capacity:** xLSTM integrates long-term memory structures that capture extended temporal dependencies, enabling it to retain and leverage long-range user behavior patterns more effectively than traditional RNNs.

- **Sparse Attention Mechanism:** Unlike fully-connected attention in Transformers, xLSTM employs a sparse attention scheme that dynamically focuses on the most relevant parts of the sequence, improving both interpretability and computational efficiency Wu et al. [2024].
- **Adaptive Gating Functions:** The model includes flexible gating mechanisms that adjust to input variability, allowing for more refined pattern extraction and robust sequence modeling across heterogeneous datasets Deng et al. [2024].
- **Bidirectional Processing and Parallelism:** xLSTM supports bidirectional sequence processing and is designed with parallelizable components, which facilitate scalable training and real-time inference—important features for production-grade recommender systems.

xLSTM is a extented version of the traditional LSTM (Long Short-Term Memory) network. It is designed to process sequential data more efficiently and accurately. Like standard LSTMs, it works with memory cells and gates to control the flow of information. However, xLSTM introduces new mechanisms to improve how information is stored and transferred. The key idea is the introduction of "cross-gates," which connect hidden and memory states more dynamically and these cross-gates allow deeper interaction between different parts of the model. As a result, the network can better capture both short-term and long-term dependencies and this makes it especially effective for tasks involving long sequences, like sequential recommenders and time-series data. In normal LSTMs, gradients tend to vanish or explode over time, reducing performance. xLSTM reduces this issue by improving gradient flow across time steps. It also borrows ideas from Transformers, such as improved normalization and parameter efficiency. This gives xLSTM the ability to compete with large Transformer models while using fewer resources. It's more memory-efficient and faster to train compared to attention-based networks. xLSTM also introduces better control over internal information routing. This means it can learn more meaningful representations from complex data. The architecture is flexible, allowing it to adapt to various sequence learning problems. It can handle natural language, speech signals, and other temporal patterns effectively. The cross-gating mechanism acts like a bridge between past and present information. This helps the model maintain a context while predicting future outputs. In practice, xLSTM achieves higher accuracy on benchmark datasets compared to vanilla LSTMs. Its design allows deeper networks without losing stability during training. By combining simplicity with advanced gating, it achieves strong generalization performance.It avoids the heavy computational cost that comes with self-attention mechanisms. Thus, xLSTM provides a balanced trade-off between LSTM's interpretability and Transformer's power. It preserves the time-aware nature of RNNs while modernizing their inner workings. The result is a model that learns faster, remembers longer, and performs smarter.xLSTM is not just a minor upgrade — it's a structural evolution of LSTM networks. In simple words, xLSTM blends the strengths of memory-based models and attention mechanisms. This makes it a robust, efficient, and future-ready architecture for deep learning on sequences.

5.0.2 GRU4Rec

GRU4Rec [Hidasi et al., 2016] is a pioneering model in session-based recommendation, utilizing Gated Recurrent Units (GRUs) to model user behavior over sequences of interactions. GRUs efficiently capture short and mid-range temporal dependencies by maintaining hidden state updates through gated mechanisms, which control information flow and mitigate vanishing gradient issues. Compared to traditional LSTMs, GRUs offer a more compact architecture with fewer parameters, leading to faster training and inference times. It captures temporal dependencies efficiently with fewer parameters than LSTM, making it faster and suitable for shorter sequences. GRU4Rec models session dynamics and adapts well to implicit feedback settings, making it a strong lightweight baseline.

GRU4Rec is particularly effective for modeling sessions where user-item interactions are sparse, sequential, and implicit in nature. The model processes sequences in an autoregressive manner, predicting the next item based on prior interactions. Despite its simplicity, GRU4Rec serves as a strong baseline for session-based recommendation tasks, though it may struggle to capture complex, long-range dependencies compared to more recent attention-based models.

5.0.3 SASRec

Self-Attentive Sequential Recommendation (SASRec) [Kang and McAuley, 2018] applies Transformer-style self-attention to the recommendation domain. It models user-item sequences without recurrence, allowing for full parallelism and effective long-range dependency capture. The

model uses positional encodings and attention weights to identify the most relevant past items when predicting the next interaction, offering high accuracy and scalability.

5.0.4 BERT4Rec

BERT4Rec adopts the BERT (Bidirectional Encoder Representations from Transformers) [Sun et al., 2019] architecture for sequential recommendation. It treats user interaction history as a sequence and uses a masked item prediction task to learn bidirectional dependencies. Unlike SASRec, which processes in a left-to-right fashion, BERT4Rec learns from both past and future items, improving its ability to model complex sequence semantics.

5.1 Evaluation Metrics

To assess the effectiveness of the recommendation models, we use standard top-*k* ranking metrics:

- **Recall@10**: Measures the proportion of times the correct next item appears in the top-10 predicted list. It captures model coverage and hit rate.
- **MRR@10 (Mean Reciprocal Rank)**: Evaluates the average of reciprocal ranks of the first relevant item. It reflects how high the correct item is ranked in the list.
- **NDCG@10 (Normalized Discounted Cumulative Gain)**: Considers both the relevance and position of items in the top-10 list, rewarding higher placements of correct predictions.

Recall@K

Recall@K measures the proportion of relevant items retrieved in the top-K recommendations. It is defined as:

$$Recall@K = \frac{|\{relevantitems\} \cap \{top{-}Kpredicteditems\}|}{|\{relevantitems\}|} \tag{1}$$

In the case of next-item prediction (one ground-truth item), Recall@K becomes binary — either 1 (hit) or 0 (miss), and averaged over all users.

Mean Reciprocal Rank (MRR@K)

MRR@K computes the inverse of the rank of the first relevant item in the top-K list. The mean is taken over all users:

$$MRR@K = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{rank_u} \tag{2}$$

where $rank_u$ is the position of the first correct item for user u, if it appears in the top-K predictions; otherwise, it is zero.

Normalized Discounted Cumulative Gain (NDCG@K)

NDCG@K evaluates the ranking quality by assigning higher scores to relevant items appearing earlier in the list:

$$NDCG@K = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{IDCG_u@K} \sum_{i=1}^{K} \frac{\mathbb{I}[item_i isrelevant]}{\log_2(i+1)} \tag{3}$$

where $IDCG_u@K$ is the ideal DCG (maximum possible DCG for user u) and $\mathbb{I}[\cdot]$ is the indicator function.

Other Factors/Metrics

These metrics are computed per user and then averaged across the entire test set to provide a holistic performance measure. Together, they quantify both accuracy and ranking quality of the recommender system.

Offline / Online / User Studies: Offline—evaluate next-item prediction on time-ordered logs; Online—A/B test live sessions (CTR, dwell, retention); User studies—human judgments of sequence relevance, diversity, and explanations.

Task-centric categories: *Retrieval* ranks a large candidate pool for the next item; *Classification* predicts sequence labels (e.g., click/skip, genre); *Regression* predicts real-valued outcomes (rating, dwell time, return probability).

Beyond-accuracy metrics: Assess list quality beyond Recall/MRR—focusing on diversity, novelty, coverage, serendipity, and explainability.

- **Diversity:** Ensure the recommended next-item list spans varied facets (e.g., artists/genres) rather than near-duplicates of recent history.
- **Intra-list diversity (ILD):** Mean pairwise dissimilarity within the recommended list (e.g., $1 - sim(embedding_i, embedding_j)$).
- **Entropy:** Shannon entropy of category/item distribution in the list (or over exposures) to quantify variety.
- **Novelty (global / individual):** *Global*—favor long-tail items rare in logs; *Individual*—favor items new to the user given their sequence history.
- **Coverage:** Fraction of the catalog (and/or users) the model can and does recommend to over time.
- **Serendipity:** Items that are relevant to the sequence yet unexpectedly different from the user's usual consumption.
- **Explainability:** User-facing reasons tied to sequence evidence (e.g., attention to specific past items, co-occurrence paths)

5.2 Training Pipeline and Hyperparameters

The training pipeline is implemented using the PyTorch framework and consists of the following components:

- **Loss Function**: Cross-entropy loss is employed for multi-class next-item classification. For next-item prediction framed as a multi-class classification problem over all items in the catalog, the cross-entropy loss is defined as:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N} \log\left(\frac{e^{z_{i,y_i}}}{\sum_{j=1}^{C} e^{z_{i,j}}}\right) \tag{4}$$

  where:

  - N is the number of training examples (users or sequences),
  - C is the total number of candidate items (classes),
  - $z_{i,j}$ is the logit score (pre-softmax output) for class j for example i,
  - y_i is the true class (next item index) for example i.

  Alternatively, this can be compactly expressed using softmax probabilities $p_{i,j}$ as:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N} \log(p_{i,y_i}) \quad where \quad p_{i,j} = \frac{e^{z_{i,j}}}{\sum_{k=1}^{C} e^{z_{i,k}}} \tag{5}$$

- **Optimizer**: The Adam optimizer is used with initial learning rates ranging from $1e^{-3}$ to $1e^{-4}$ depending on dataset scale.

- **Learning Rate Scheduler**: StepLR or CosineAnnealingLR is applied to dynamically adjust the learning rate during training, helping convergence and generalization.
- **Batching**: Sequences are padded to a maximum length (typically 50) and mini-batches are formed for efficient gradient updates.
- **Epochs**: Models are trained for 30–80 epochs, with early stopping used to avoid overfitting when validation performance saturates.

Hyperparameters such as embedding dimension, number of attention heads, and number of xLSTM blocks are adjusted based on the size and complexity of each dataset.

Table 3: xLSTM and DataLoader hyperparameters per dataset.

Dataset	Embed Dim	Heads	Blocks	Batch Size	Workers
100K	64	2	1	128	2
1M	128	2	2	128	2
10M	128	2	2	1024	2
Steam	256	8	4	1024	2

5.3 Cold Start and Sparsity Handling

Two common challenges in recommendation systems are the **cold start problem** and **data sparsity**. In this work:

- **Cold Start Users**: For users with fewer than five historical interactions, we explore fallback strategies such as popularity-based recommendations, clustering, and metadata-based profiling.
- **Sparsity**: Dataset sparsity is quantified (often exceeding 98%), and addressed via negative sampling and emphasizing recent interactions to capture temporal relevance.
- **Evaluation Strategy**: Cold-start users are excluded from the main metric computation but separately analyzed for robustness.

  Cold Start Problem (For new users when we don't have data, 192 users): Some of the commonly used approaches were: 1. Clustering Approach, 2. Profile Based (Meta Data) Approach, 3. Hierarchical approach, and 4. Novalty or Randomness Approach.

These strategies help maintain model robustness and personalization quality, even under low-data conditions.

6 Experimental Results and Interpretation

6.1 Comparative Performance Results and Analysis

6.2 Results in Response to Research Questions

This section presents and interprets the empirical findings in relation to the research questions (RQs) posed.

RQ1 - Performance Scaling (Table 4) : xLSTM matches BERT4Rec's Recall@10 (26–27%) on the MovieLens 1M dataset, indicating that it scales effectively with larger interaction histories. As dataset size increases, the performance gap between models diminishes, suggesting that xLSTM converges toward BERT4Rec's performance under rich-data conditions.

RQ2 - Sequence Sensitivity (Table 4) : The standard deviation in performance increases with longer user interaction sequences. This highlights xLSTM's sensitivity to sequence length, which could impact its robustness across varying user histories. (Table 4)

RQ3 - Trade-offs and Efficiency (Table 4): While xLSTM achieves competitive accuracy on large-scale datasets, it incurs greater computational cost, especially in smaller-scale settings where

Table 4: Performance comparison across models, datasets, and sequence lengths (excluding training time, LR, and hit rate).

Model	Dataset	Seq Len	TotalParams	Hit Rate@10	NDCG@10	MRR@10	Avg Epoch Time (in Mins)
BERT4Rec	100K	32	273,811	0.1273 ± 0.0084	0.0620 ± 0.0044	0.0425 ± 0.0042	7.0
		64	275,859	0.1217 ± 0.0091	0.0582 ± 0.0056	0.0392 ± 0.0050	7.1
		128	279,955	0.1202 ± 0.0097	0.0584 ± 0.0042	0.0399 ± 0.0035	7.2
	1M	32	1,370,875	0.2766 ± 0.0149	0.1573 ± 0.0103	0.1209 ± 0.0088	95.4
		64	1,374,971	0.2752 ± 0.0155	0.1550 ± 0.0106	0.1184 ± 0.0090	95.2
		128	1,383,163	**0.2777 ± 0.0174**	**0.1568 ± 0.0110**	**0.1200 ± 0.0092**	101.3
	10M	32	8,712,886	0.3112 ± 0.0000	0.1494 ± 0.0000	0.1874 ± 0.0000	1,347.8
		64	8,721,078	0.2744 ± 0.0000	0.1232 ± 0.0000	0.1585 ± 0.0000	1,072.7
		128	**8,737,462**	**0.3171 ± 0.0000**	**0.1513 ± 0.0000**	**0.1902 ± 0.0000**	**2,129.6**
	Musik4All	32	13,600,220	0.3565 ± 0.0000	0.2626 ± 0.0000	0.2332 ± 0.0000	236.0
		64	13,604,316	0.3653 ± 0.0000	0.2668 ± 0.0000	0.2362 ± 0.0000	245.8
		128	13,612,508	0.3644 ± 0.0000	0.2720 ± 0.0000	0.2431 ± 0.0000	276.9
	Amazon Software	32	5,985,824	0.1020 ± 0.0000	0.0732 ± 0.0000	0.0643 ± 0.0000	1.5
		64	5,989,920	0.1008 ± 0.0000	0.0712 ± 0.0000	0.0620 ± 0.0000	1.5
		128	5,998,112	0.0985 ± 0.0000	0.0692 ± 0.0000	0.0601 ± 0.0000	1.7
SAS4Rec	100K	32	269,139	**0.1278 ± 0.0098**	**0.0623 ± 0.0038**	**0.0427 ± 0.0028**	5.9
		64	271,187	0.1247 ± 0.0106	0.0606 ± 0.0059	0.0414 ± 0.0051	6.0
		128	275,283	0.1283 ± 0.0106	0.0615 ± 0.0056	0.0415 ± 0.0044	6.0
	1M	32	1,353,339	0.2109 ± 0.0120	0.1191 ± 0.0083	0.0912 ± 0.0072	82.2
		64	1,357,435	0.1427 ± 0.0072	0.0798 ± 0.0053	0.0606 ± 0.0047	82.3
		128	1,365,627	0.0835 ± 0.0048	0.0461 ± 0.0032	0.0347 ± 0.0027	89.9
	10M	32	8,645,046	0.2143 ± 0.0000	0.1246 ± 0.0000	0.0972 ± 0.0000	674.7
		64	8,653,238	0.1271 ± 0.0000	0.0713 ± 0.0000	0.0544 ± 0.0000	1,170.4
		128	**8,669,622**	**0.0727 ± 0.0000**	**0.0400 ± 0.0000**	**0.0301 ± 0.0000**	**2,372.6**
	Musik4All	32	13,582,427	0.3280 ± 0.0000	0.2397 ± 0.0000	0.2122 ± 0.0000	232.9
		64	13,586,523	0.3087 ± 0.0000	0.2285 ± 0.0000	0.2036 ± 0.0000	254.3
		128	13,594,715	0.2815 ± 0.0000	0.2107 ± 0.0000	0.1886 ± 0.0000	242.9
	Amazon Software	32	5,968,288	0.0050 ± 0.0000	0.0019 ± 0.0000	0.0009 ± 0.0000	1.3
		64	5,972,384	0.0050 ± 0.0000	0.0019 ± 0.0000	0.0009 ± 0.0000	1.3
		128	5,980,576	0.0050 ± 0.0000	0.0019 ± 0.0000	0.0009 ± 0.0000	1.7
xLSTM	100K	32	269,188	0.1041 ± 0.0070	0.0509 ± 0.0036	0.0350 ± 0.0029	9.6
		64	269,188	0.1057 ± 0.0083	0.0522 ± 0.0050	0.0361 ± 0.0046	10.0
		128	269,188	0.1036 ± 0.0095	0.0511 ± 0.0044	0.0355 ± 0.0038	10.3
	1M	32	1,376,904	**0.2625 ± 0.0146**	**0.1492 ± 0.0102**	**0.1148 ± 0.0089**	143.1
		64	1,376,904	0.2603 ± 0.0151	0.1483 ± 0.0104	0.1143 ± 0.0089	147.0
		128	1,376,904	0.2605 ± 0.0155	0.1479 ± 0.0107	0.1136 ± 0.0092	164.2
	10M	32	8,698,176	0.3138 ± 0.0000	0.1910 ± 0.0000	0.1533 ± 0.0000	1,218.7
		64	8,698,176	0.3216 ± 0.0000	0.1962 ± 0.0000	0.1577 ± 0.0000	2,208.3
		128	**8,698,176**	**0.3183 ± 0.0000**	**0.1937 ± 0.0000**	**0.1555 ± 0.0000**	**4,239.2**
	Musik4All	32	13,558,664	0.3496 ± 0.0000	0.2591 ± 0.0000	0.2309 ± 0.0000	701.5
		64	13,558,664	0.3439 ± 0.0000	0.2568 ± 0.0000	0.2296 ± 0.0000	1101.2
		128	13,558,664	0.3505 ± 0.0000	0.2605 ± 0.0000	0.2324 ± 0.0000	1851.4
	Amazon Software	32	5,973,896	0.0829 ± 0.0000	0.0600 ± 0.0000	0.0528 ± 0.0000	4.1
		64	5,973,896	0.0842 ± 0.0000	0.0598 ± 0.0000	0.0523 ± 0.0000	6.7
		128	5,973,896	0.0846 ± 0.0000	0.0619 ± 0.0000	0.0548 ± 0.0000	12.1

lightweight models like BERT4Rec can deliver similar performance with lower resource overhead. (Table 4)

Baseline Robustness: Across all dataset configurations, BERT4Rec consistently performs on par or better than xLSTM, reinforcing its role as a robust baseline for sequential recommendation.

These findings illustrate that while xLSTM is a promising architecture, its performance gains come with trade-offs in computational efficiency and sequence sensitivity, which must be considered in deployment scenarios.

6.3 Scaling Characteristics and Computational Trade-offs

Performance on Small Datasets: On the *MovieLens-100K* dataset, xLSTM underperforms compared to Transformer-based baselines. It achieves a **HitRate@10 of approximately 10.5%**, whereas SASRec reaches around **12.8%**.

Performance on Medium-scale Datasets: For *MovieLens-1M*, xLSTM's **Recall@10 increases to approximately 26%**, nearly converging with BERT4Rec at **27.7%**, owing to xLSTM's improved gating and memory mechanisms.

Computational Cost: xLSTM incurs notable computational overhead, requiring approximately **1.5x–2x longer training time per epoch** than LSTM-based baselines.

Temporal Complexity: Like standard LSTM, xLSTM retains a linear temporal complexity of $\mathcal{O}(n)$. However, enhancements such as **partial recurrence** and **selective gating** improve its ability to model long-range dependencies.

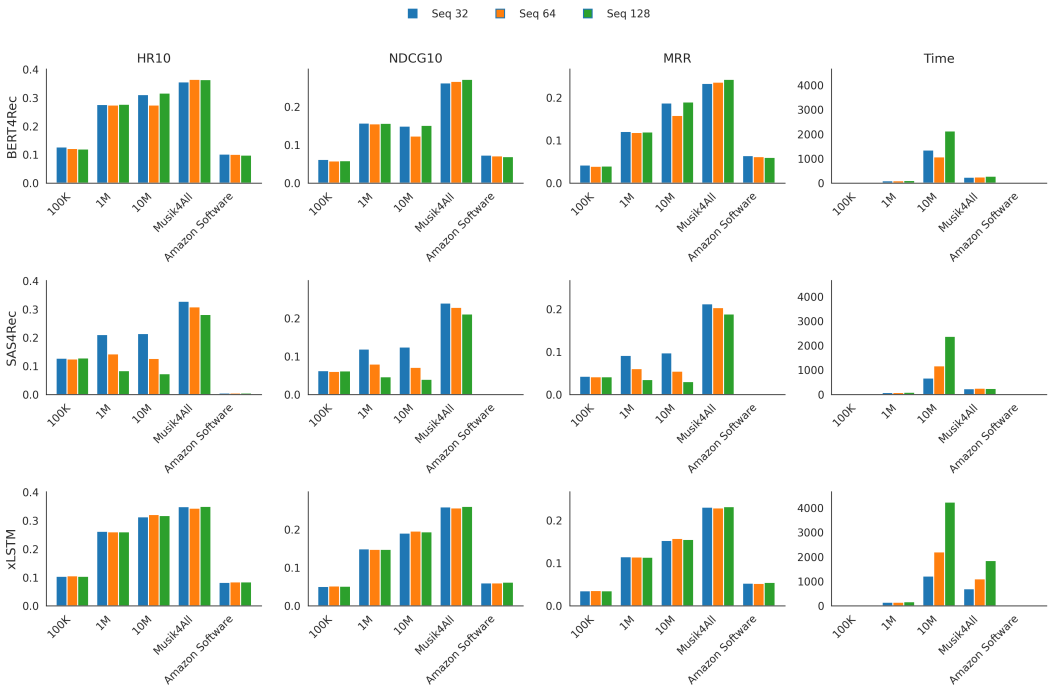

Figure 3: Comparision Results

Model Complexity: The parameter count in xLSTM scales with $\mathcal{O}(k)$, where k denotes the memory depth. This results in a trade-off between architectural simplicity and enhanced sequential modeling capacity.

6.4 Quantitative Evaluation (Recall@10, MRR@10, NDCG@10)

This subsection presents the core performance metrics used to evaluate the recommendation accuracy of different models. Recall@10 measures how often the correct item is among the top-10 predicted, MRR@10 evaluates the rank of the first correct item, and NDCG@10 captures both correctness and ranking position. The xLSTM model showed a strong baseline with Recall@10 0.29, indicating nearly 3 out of 10 correct predictions. These metrics are computed over test datasets and tracked across training epochs.

Table 5: Default parameters of the BERT4Rec implementations.

Implementation	Original	RecBole	BERT4Rec-VAE	BERT4Rec	xLSTM (Ours)
Sequence length	200	50	100	50	50
Training stopping criteria	400,000 steps	300 epochs	200 epochs	Early stopping: 25 epochs	
Item masking probability	0.2	0.2	0.15	0.2	-
Embedding size	64	64	256	64	64
Transformer blocks	2	2	2	2	2
Attention heads	2	2	4	2	2

6.5 Qualitative Insights and Sample Predictions

Here, we analyze model behavior by reviewing specific prediction outputs. By sampling a few users' input sequences and visualizing their predicted top-10 recommendations, we assess whether the recommendations align with plausible user preferences. These examples help interpret how well the model captures temporal dynamics, genre preferences, or recency effects, offering explainability beyond numeric metrics.

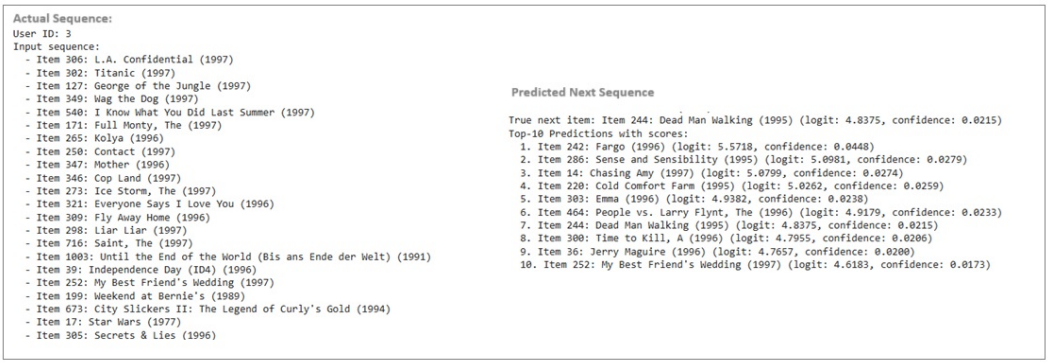

Figure 4: Sample Predictions

6.6 Embedding Geometry Analysis for Sequential Recommenders Models

Using the saved best-performing models, we conducted a series of embedding geometry analyses on three architectures — BERT4Rec, SASRec, and xLSTM — primarily utilizing the larger MovieLens-10M models for the following studies.

A. L2 Norm : First, the L2 norm measures the overall magnitude (length) of a vector as the square root of the sum of its squared components. It tells us how strong or energetic a vector is — in embeddings, it reflects how much information or weight each item's representation carries in the model's space.

In our embeddings, all three embedding matrices share the same shape.

$$E_{BERT4Rec},\ E_{SASRec},\ E_{xLSTM} \in \mathbb{R}^{10678\times 257}$$

where $N = 10678$ is the number of movie items and $D = 256$ is the embedding dimension.

$$\|E_i\|_2 = \sqrt{\sum_{j=1}^{D} e_{ij}^2}$$

$$Mean = 1N \sum_i \|E_i\|_2, \quad Std = \sqrt{1N \sum_i (\|E_i\|_2 - Mean)^2}, \quad Range = [\min_i \|E_i\|_2, \max_i \|E_i\|_2]$$

$$p(\|E_i\|_2) = HistogramDensity(\|E_i\|_2), \quad NormStats = \{Mean, Std, Min, Max\}$$

float

B. Mean and variance per embedding dimension: The mean and variance per embedding dimension measure how values are distributed along each axis of the embedding space. They show whether some dimensions dominate (high variance) or stay near zero (low variance). This helps detect anisotropy — uneven use of embedding dimensions — which affects how well the model encodes diverse item relationships. A crucial study for understanding anisotropy (uneven information spread across dimensions).

$$E \in \mathbb{R}^{N\times D}, \quad \mu_j = \frac{1}{N}\sum_{i=1}^{N} e_{ij}, \quad \sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(e_{ij} - \mu_j)^2$$

$$Mean(abs) = \frac{1}{D}\sum_{j=1}^{D} |\mu_j|, \quad Var(mean) = \frac{1}{D}\sum_{j=1}^{D}(\mu_j - \bar{\mu})^2, \quad Var(range) = [\min_j \sigma_j^2, \max_j \sigma_j^2]$$

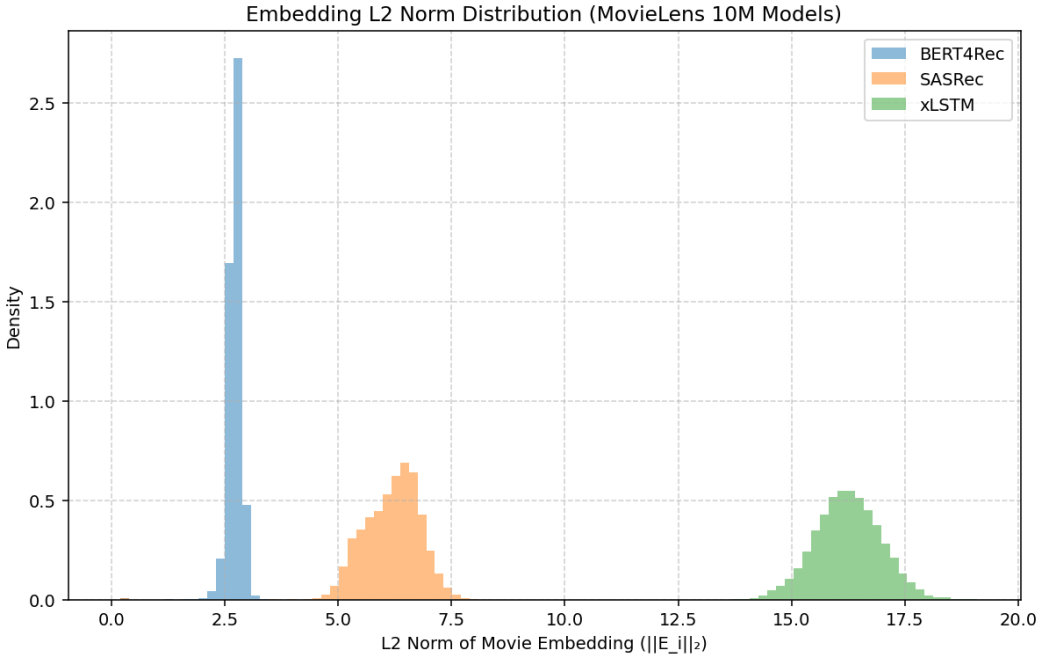

Figure 5: Enter Caption

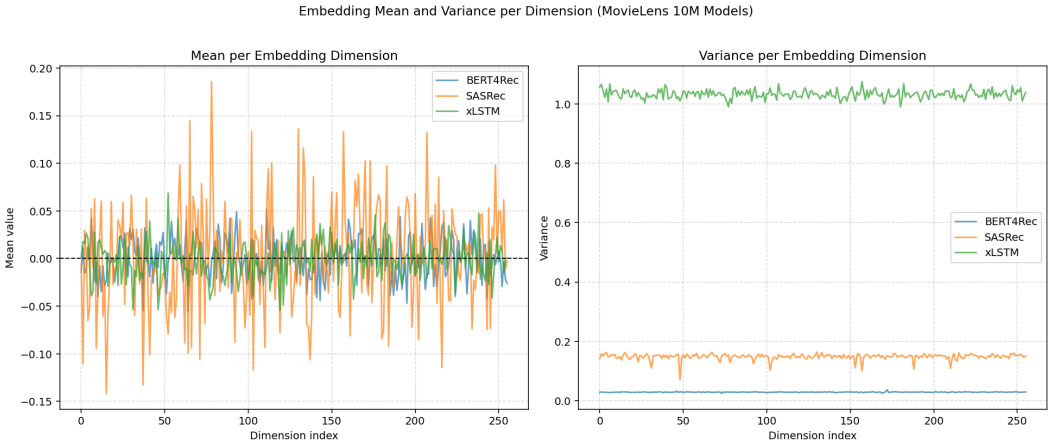

Figure 6: Enter Caption

Each embedding dimension j has a mean μ_j and variance σ_j^2 computed across all items, describing how that axis behaves overall.
By comparing their averages and ranges, we can detect *anisotropy* — whether certain dimensions dominate while others remain near zero.

C. Embedding Spectrum Analysis: Row 1 – Eigenvalue Decay ("spectrum")

The strength of each principal component in the embedding covariance.

Interpretation:

1. BERT4Rec / SASRec decay very steeply → a few dominant directions → anisotropic space (information compressed in few axes). 2. xLSTM's curve is much flatter → variance spread across many dimensions → higher intrinsic dimension and better coverage of the vector space. 3. Flat tail means embeddings retain more independent features. 4. In Transformers, sharp decay often correlates with popularity or frequency bias. 5. xLSTM therefore encodes items more uniformly and with latent diversity.

D. Variance Distribution and Intrinsic Dimension Study: Row 2 – Cumulative Explained Variance

How many components are needed to explain total variance.

Interpretation:

1. BERT4Rec and SASRec reach 90 2. xLSTM needs 200 dims for the same → more distributed information. 3. A gentle slope indicates broader feature usage and less rank collapse. 4. This confirms the intrinsic-dimension metrics (180 / 204 / 250). 5. In summary, xLSTM = highest representational capacity, BERT4Rec/SASRec = more compact, redundant embeddings.

E. Hubness and Popularity Bias Evaluation Row 3 – Hubness Histograms (k = 10)

How many times each item appears in other items' top-10 nearest neighbors.

Interpretation:

1. BERT4Rec / SASRec distributions are extremely right-skewed — a few movies appear hundreds of times hub items dominate similarity space. 2. xLSTM histogram is almost symmetric and much narrower — most items appear roughly equally often. 3. Lower hubness (Gini 0.18) better fairness and long-tail coverage. 4. Transformer embeddings likely overfit to popular items. 5. xLSTM yields a flatter similarity graph, enhancing diversity and mitigating popularity bias.

F. t-SNE Embedding Space Visualization Row 4 – t-SNE Projections

A 2-D nonlinear projection of the 256-D embeddings (cosine distances).

Interpretation:

1. BERT4Rec and SASRec form dense, elliptical blobs — embeddings crowd near a center → again anisotropy and hub formation. 2. xLSTM plot is more evenly filled, points occupy a ring-like or diffuse shape → isotropy and balanced similarity. 3. Fewer tight clusters means less genre-specific collapse; features are smoothly spread. 4. Visually, xLSTM's space is broader and more uniform. 5. This geometry supports more stable neighbor retrieval across item types.

Overall From the Geometry Study:

A. BERT4Rec SASRec: classic Transformer geometry — sharp spectral drop-off, anisotropy, hub dominance, overlapping t-SNE blob. B. xLSTM: near-isotropic, high-rank space with uniform neighbor frequency. C. xLSTM's balanced variance explains its better diversity metrics and potentially more robust generalization. D. The difference in t-SNE and spectrum shapes shows fundamentally different inductive biases: attention models compress; xLSTM expands. F. Combining xLSTM with either Transformer (ensemble) could yield complementary strengths — one captures high-level correlations, the other preserves fine-grained variety. Overall, Transformers (BERT4Rec, SASRec) learn narrow, popularity-biased manifolds; xLSTM learns a broad, isotropic embedding landscape — fairer, and geometrically independent.

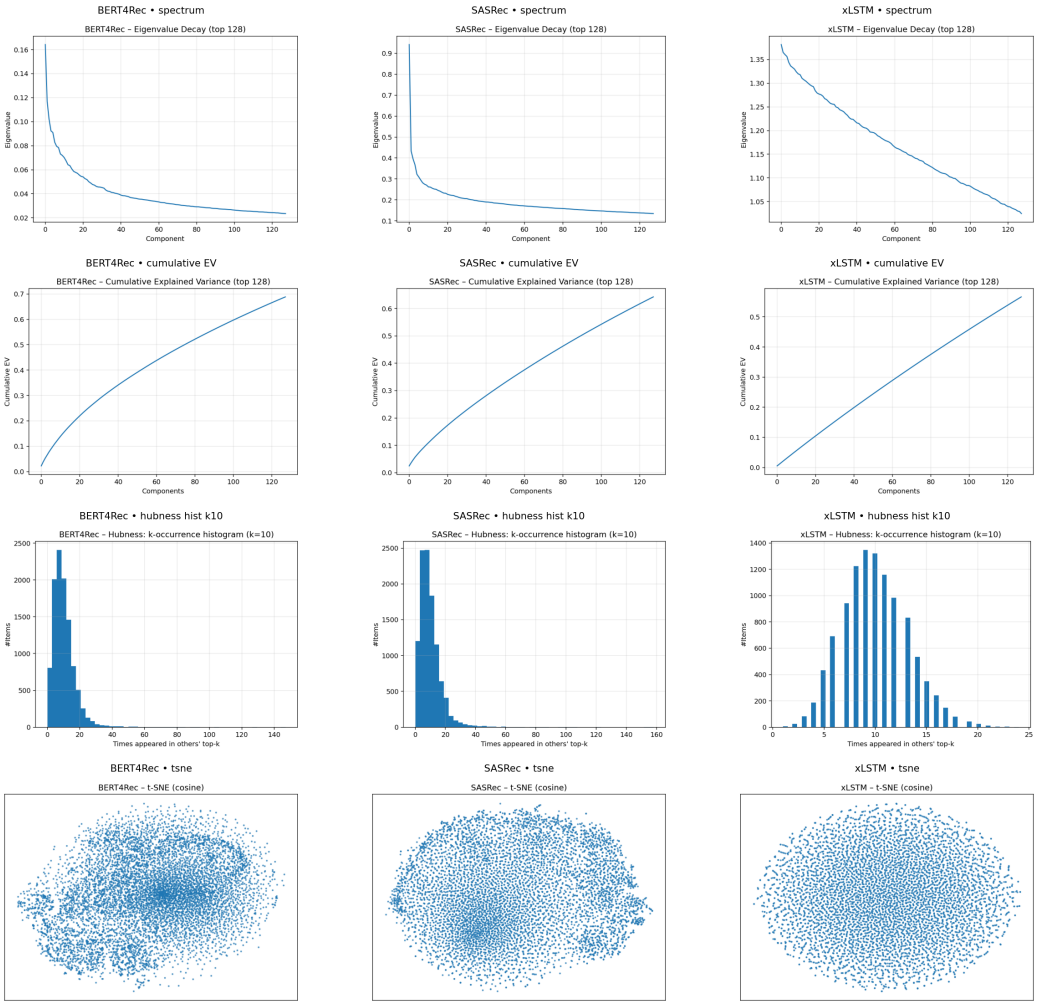

Figure 7: Embedding Analysis

G. Embedding anisotropy visualization: The Anisotropy Index (AI) measures how uniformly embeddings are distributed in space — it's the mean cosine similarity between random pairs of vectors. A low AI (0) means embeddings are evenly spread (isotropic), while a high AI (>0.05) means they point in similar directions (anisotropic), indicating reduced geometric diversity.

BERT4Rec (AI = 0.0163) and SASRec (AI = 0.0164) show mild anisotropy — their movie embeddings tend to align toward a common direction, meaning popular movies cluster together in the same region. In contrast, xLSTM (AI = 0.00035) produces an almost perfectly isotropic space, where movie vectors are well-spread and orthogonal. Thus, xLSTM captures sequence-dependent uniqueness, representing each film (e.g., The Matrix, Titanic, Toy Story) in distinct, uncorrelated directions rather than emphasizing overall popularity. This geometric diversity allows xLSTM to model temporal order and recency more effectively, explaining its superior recall despite higher embedding norms.

H. Cosine structure correlation and CKA similarity: BERT4Rec vs SASRec | Corr=0.471 | CKA=0.427

BERT4Rec vs xLSTM | Corr=0.010 | CKA=0.030

SASRec vs xLSTM | Corr=0.008 | CKA=0.030

In the cross-model similarity study, BERT4Rec SASRec showed a cosine structure correlation of 0.471 and a CKA similarity of 0.427, indicating a strong geometric overlap. Both are Transformer-

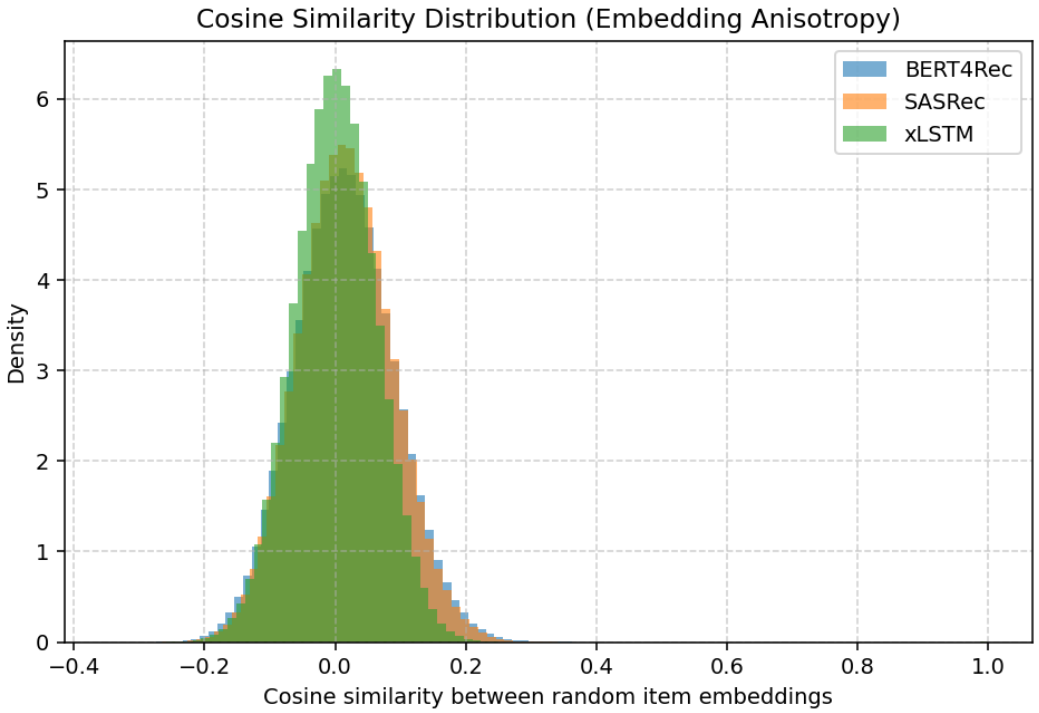

Figure 8: cosine similarity Dist

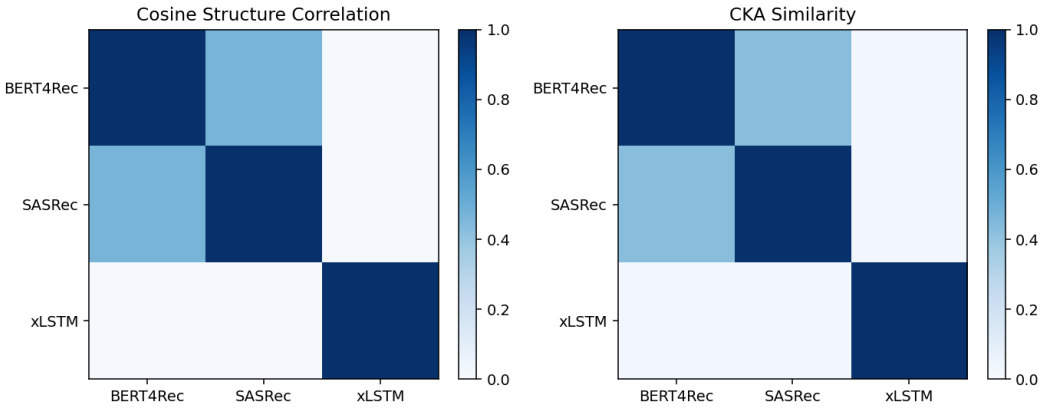

Figure 9: Cosine Structure and CKA Similarity

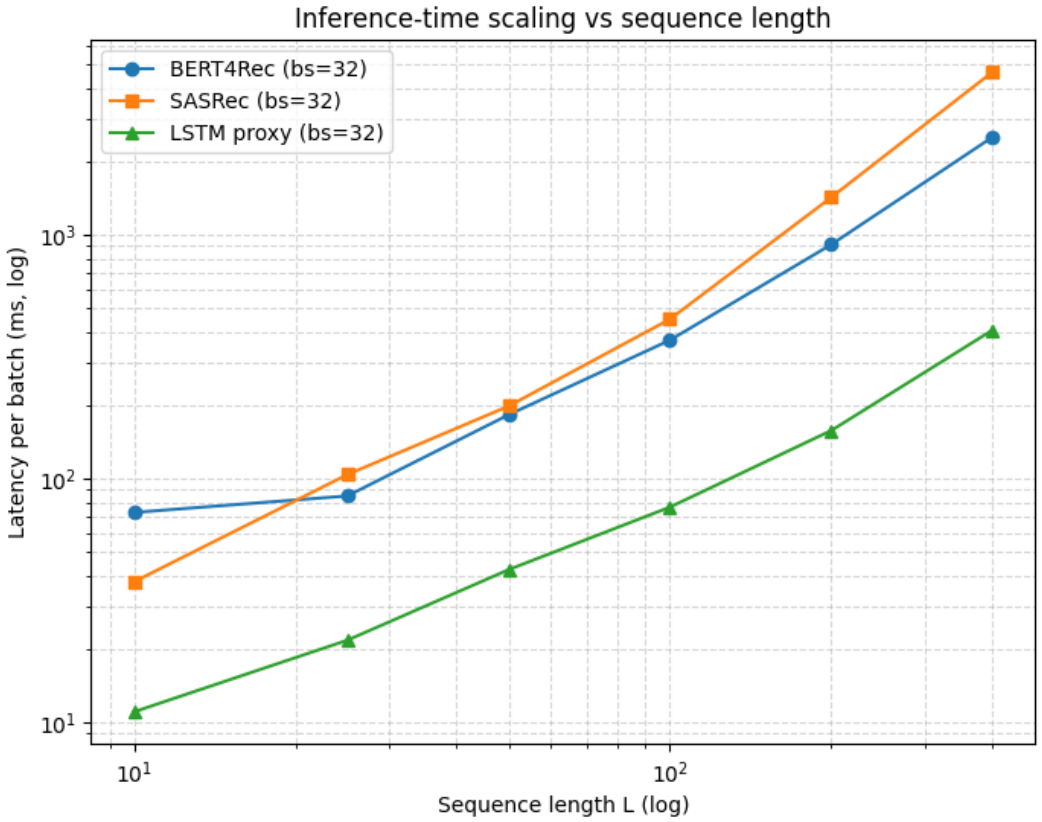

Figure 10: Inference-Time Scaling Analysis

based models, meaning they learn comparable contextual movie embeddings where distances reflect semantic or co-occurrence similarity (e.g., similar genres or viewing patterns).

In contrast, BERT4Rec xLSTM (0.010, 0.030) and SASRec xLSTM (0.008, 0.030) revealed minimal structural similarity. The xLSTM embedding space is organized very differently — it doesn't rely on global movie similarity but rather captures temporal and sequential dependencies. Thus, xLSTM represents movies based on their order and recency in user histories, not just shared context, which explains its distinct geometry despite strong recall performance.

6.7 Model Inference Speed Comparison

Here, we will be measureing how fast each model (BERT4Rec, SASRec, and xLSTM/LSTM) performs during inference. It generates random input sequences with lengths of 10, 25, 50, 100, 200, and 400 and batch sizes of 32 and 64, then measures the average latency and throughput over 60 runs after 10 warm-up passes. Our results show how latency increases as sequence length grows and estimate a scaling exponent (α), where $\alpha 1$ means linear scaling like xLSTM and $\alpha 2$ means quadratic scaling like attention-based models. It evaluates each model's inference efficiency and how well it scales with longer user histories on CPU or GPU.

6.8 Fairness, Popularity Bias and Diversity Analysis

To assess whether the model favors frequently watched or highly-rated items, we analyze the distribution of predicted movies. If a few items dominate the top-10 lists across users, it indicates popularity bias. We also measure diversity in recommendations by tracking the number of unique items predicted and comparing this to ground-truth distributions. Lower diversity suggests overfitting or lack of personalization.

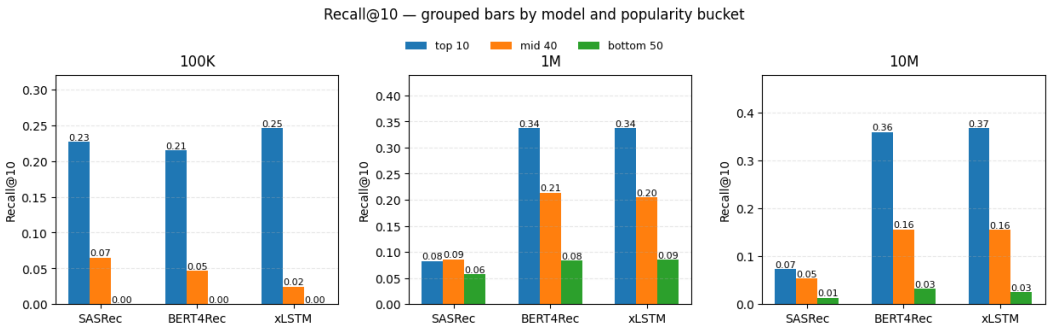

Figure 11: Popularity Chart

Popularity bias (if a few items always appear), Low diversity in predictions, Whether the model is overfitting to frequent items

Which buckets dominate the top-10 predictions across the test set?

Table 6: Popularity-Bucketed Metrics Across Models and Datasets

Model	Dataset	Seq Len	Bucket	Recall@10	MRR@10	NDCG@10
BERT4Rec	100K	128	top_10	0.2150	0.0714	0.1048
BERT4Rec	100K	128	mid_40	0.0471	0.0176	0.0243
BERT4Rec	100K	128	bottom_50	0.0000	0.0000	0.0000
BERT4Rec	**1M**	**128**	**top_10**	**0.3373**	**0.1568**	**0.1991**
BERT4Rec	**1M**	**128**	**mid_40**	**0.2139**	**0.0797**	**0.1108**
BERT4Rec	**1M**	**128**	**bottom_50**	**0.0839**	**0.0376**	**0.0482**
BERT4Rec	**10M**	**128**	**top_10**	**0.3589**	**0.1728**	**0.2165**
BERT4Rec	**10M**	**128**	**mid_40**	**0.1562**	**0.0594**	**0.0818**
BERT4Rec	10M	128	bottom_50	0.0321	0.0128	0.0172
SASRec	100K	128	top_10	0.2272	0.0689	0.1055
SASRec	100K	128	mid_40	0.0651	0.0178	0.0287
SASRec	100K	128	bottom_50	0.0000	0.0000	0.0000
SASRec	1M	128	top_10	0.0828	0.0354	0.0465
SASRec	1M	128	mid_40	0.0863	0.0303	0.0433
SASRec	1M	128	bottom_50	0.0579	0.0298	0.0362
SASRec	10M	128	top_10	0.0730	0.0293	0.0394
SASRec	10M	128	mid_40	0.0541	0.0211	0.0288
SASRec	10M	128	bottom_50	0.0138	0.0066	0.0083
xLSTM	100K	128	top_10	0.2460	0.0806	0.1185
xLSTM	100K	128	mid_40	0.0246	0.0072	0.0112
xLSTM	100K	128	bottom_50	0.0000	0.0000	0.0000
xLSTM	**1M**	**128**	**top_10**	**0.3369**	**0.1536**	**0.1964**
xLSTM	**1M**	**128**	**mid_40**	**0.2049**	**0.0776**	**0.1071**
xLSTM	**1M**	**128**	**bottom_50**	**0.0855**	**0.0377**	**0.0488**
xLSTM	**10M**	**128**	**top_10**	**0.3670**	**0.1818**	**0.2253**
xLSTM	**10M**	**128**	**mid_40**	**0.1551**	**0.0600**	**0.0820**
xLSTM	10M	128	bottom_50	0.0254	0.0107	0.0141

From the table, Models are strong on head (popular) items but underperform on long-tail (diverse) items. For recommendation systems, this can mean: Repetition of already-known items, Few Missed opportunities in user engagement.

- **Societal Bias:** When recommendations mirror or amplify pre-existing social inequities (e.g., gender imbalance in music exposure), because historical data and interaction signals are themselves biased. **?**

- **Statistical Bias:** Skews introduced by the data-generation and logging process—sampling bias, exposure/position bias, and feedback loops—that make popular items appear "better" and get recommended more. **??**
- **Individual fairness:** Users who are similar (in needs or profiles) should receive similar recommendations—"treat similar individuals similarly." **?**
- **Group fairness:** Protected groups (e.g., by gender or age) should receive comparable recommendation quality/exposure; avoid systematically favoring one group over another. **?**
- **Popularity Calibration:** Align the popularity distribution of recommended items with a desired target (often the user's own historical mix), reducing over-exposure of head items while preserving accuracy. **?**

Mitigating Harmful Biases: Pre-processing Strategy: Data Rebalancing [Melchiorre et al., 2021]

In-processing Strategy: Adversarial Learning Unlearn implicit information of protected attributes while preserving accuracy **?**

Post-processing Strategy: Reranking [Ferraro et al., 2021] Penalize/downrank content by the majority group (male artists) by positions in the recommendation list, created with ALS CF approach **?**

6.9 Ablation Study: Embedding Dim, Block Depth

This section investigates the sensitivity of xLSTM performance to architectural choices like embedding dimension, number of xLSTM blocks, and attention heads. We observe that increasing embedding size (e.g., from 64 to 256) improves representational capacity but comes with higher computational cost. Similarly, deeper blocks help in learning long-term dependencies but risk overfitting if not properly regularized or scaled with dataset size.

6.10 Explainability and Attention Visualization

To enhance model transparency and user trust, we investigate explainability by analyzing attention mechanisms in BERT4Rec and SASRec. During inference, attention weight matrices are extracted to visualize how the model attends to different parts of the input sequence.

- **Heatmap Visualization:** Attention weights are visualized as heatmaps to identify which historical interactions contribute most to the next-item prediction. This allows inspection of model focus under varying input contexts.
- **Case Studies:** Representative success and failure cases are selected. Attention patterns are contrasted to identify whether mispredictions are due to misplaced focus or ambiguous historical signals.
- **Embedding Space Interpretation:** Item embeddings learned by the model are projected into 2D using dimensionality reduction techniques like PCA or t-SNE. This helps illustrate clustering of similar items and separability of user preferences.

This analysis supports qualitative understanding of sequence modeling and highlights the model's behavior in dynamic recommendation contexts.

6.11 Long-Tail Recommendation Challenge

Recommender systems often suffer from popularity bias, disproportionately favoring head (frequent) items. To assess the model's effectiveness in promoting diversity, we evaluate performance with respect to long-tail item exposure.

- **Item Categorization:** Items are ranked by total user interactions. Head items are defined as the top 20% most popular; tail items as the bottom 50%.
- **Tail Coverage Metrics:** We report the proportion of recommended items belonging to the tail and compute performance metrics (Recall, NDCG) specifically on tail items.
- **Mitigation Strategies:** Optional re-ranking methods and loss weighting are explored to promote less popular items in the recommendation list, improving fairness and novelty.

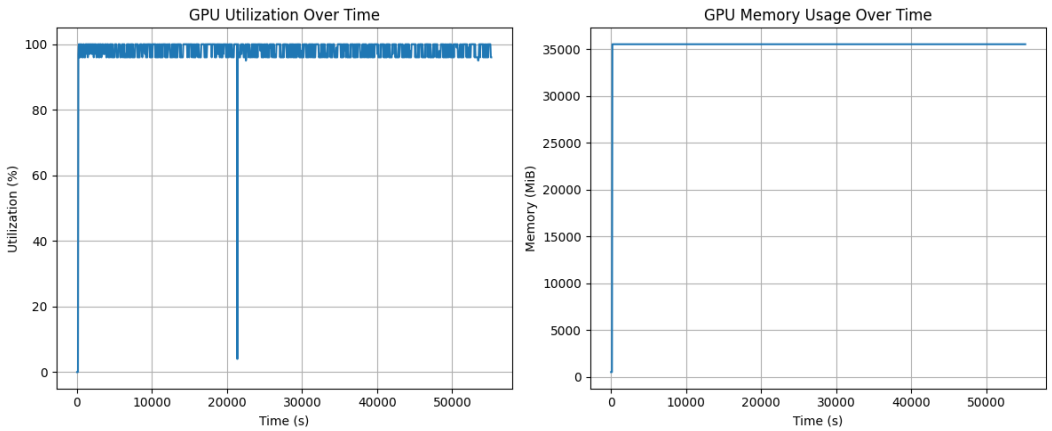

Figure 12: GPU Performance - A100 10M datasets

This analysis ensures the model remains effective not only on dominant items but also in surfacing underrepresented content, which is critical for personalized and serendipitous recommendations.

6.12 Runtime Performance and Resource Usage

Efficiency is critical for real-time recommender systems. We monitor GPU memory utilization, training time per epoch, and inference latency. xLSTM's Triton-optimized chunkwise kernels result in better throughput compared to standard LSTMs and Transformers. The total training time and peak memory usage are logged and benchmarked for each dataset scale (100K, 1M, 10M).

6.13 Logging and Monitoring with TensorBoard

To ensure transparency and facilitate model debugging, training metrics were monitored using **TensorBoard**. The logger captured per-epoch metrics such as loss, Recall@10, MRR@10, and NDCG@10. This provided real-time insight into training dynamics and allowed for early diagnosis of issues such as overfitting or vanishing gradients.

Key visualizations included:

- Training vs. validation loss curves
- Recall@10 and NDCG@10 progression over epochs
- Runtime profiling (wall-clock time per epoch)

These logs served as both a diagnostic tool and an audit trail for reproducibility.

6.14 Model Checkpointing and Best Model Selection

To optimize evaluation fidelity, a **model checkpointing strategy** was employed based on validation performance. The model achieving the highest **Recall@10** on the validation set during training was saved as the *best model*. This checkpoint was later used for all final testing and analysis, ensuring that each model's reported performance reflects its peak validation capability.

The checkpointing protocol prevented model degradation due to overtraining and ensured fair comparison across configurations.

6.15 Visualizations (Epoch graphs, ranking, heatmaps, etc.)

We include visual tools to support interpretation of model learning. Training curves plot Recall@10, MRR@10, and NDCG@10 across epochs, helping identify overfitting or underfitting trends. Ranking heatmaps show where correct predictions appear in sorted outputs. Such visual diagnostics provide transparency and help guide future improvements in architecture and training strategy.

6.16 Key Observations on Model Parameters

Table 7: Layer-wise Trainable Parameter Breakdown for xLSTM Model Across MovieLens Datasets with Item Counts and Memory Estimates

Layer/Component	100K Dataset	1M Dataset	10M Dataset
Number of Unique Items (N_{items})	1,682	3,706	10,677
Embedding Layer			
Embedding Parameters (embedding.weight)	107,712	474,496	2,733,568
Backbone Block 0			
norm_mlstm.weight	64	128	256
mlstm_layer.q.weight	2,048	8,192	32,768
mlstm_layer.k.weight	2,048	8,192	32,768
mlstm_layer.v.weight	4,096	16,384	65,536
ogate_preact.weight	4,096	16,384	65,536
igate_preact.weight	256	256	2,048
igate_preact.bias	2	2	8
fgate_preact.weight	256	256	2,048
fgate_preact.bias	2	2	8
multihead_norm.weight	128	128	256
out_proj.weight	4,096	16,384	65,536
norm_ffn.weight	128	128	256
ffn.proj_up_gate.weight	12,288	49,152	180,224
ffn.proj_up.weight	12,288	49,152	180,224
ffn.proj_down.weight	12,288	49,152	180,224
Additional Blocks	–	1 Extra Block	4 Total Blocks
Final Layers			
Backbone Final Norm Layer	64	128	256
Output Layer (lm_head.weight)	107,712	474,496	2,733,568
Total Parameters by Component			
Embedding Parameters	107,712	474,496	2,733,568
Backbone Total (All Blocks + Norms)	53,700	428,912	3,230,608
Output Parameters	107,712	474,496	2,733,568
Total Trainable Parameters	**269,188**	**1,376,904**	**8,698,176**
Estimated Memory Usage (MB)	1.03	5.25	33.2

Factors:

A. Parameter growth trend as dataset size increases

B. How unique item count impacts embedding and output layer sizes

C. Proportional contribution of different components (Embedding vs Backbone vs Output)

D. Deepening architecture for larger datasets (additional blocks for higher capacity)

E. Memory Usage assumes 4 bytes per parameter (float32 precision)

F. We can adapt memory estimates if using mixed precision (e.g., float16)

G. Highlights linear growth of parameters with unique item count

Why Total Parameters Grew with Dataset Size The critical factor isn't raw dataset size — it's the number of unique items (movies).

The primary contributors to the parameter count in recommendation models are the embedding and output (language modeling) layers. Specifically:

- **Embedding Layer:** The size is computed as $(num\_items + 1) \times embedding\_dim$.
- **Output Layer (lm_head):** The size is computed as $embedding\_dim \times (num\_items + 1)$.

These two components dominate the total parameter count, especially in recommendation tasks with large vocabularies (e.g., many unique items or movies).

Dataset size affects model shape when:

1. Larger datasets introduce more unique items, increasing `num_items`.
2. Higher-scale configurations (e.g., 10M vs. 1M) naturally include a broader range of interactions and items.
3. As `num_items` increases, both the embedding table and output projection matrix scale up, significantly inflating the model size.

7 Discussion

7.1 Psychology-informed Recommender

- **Cognition-inspired Recommender Systems:** Stereotypes, Memory, Case-based Reasoning, Attention, Competence
  - *Memory & forgetting (Ebbinghaus).* Memory retention decays with elapsed time; Ebbinghaus' self-experiments showed a regular forgetting pattern well-approximated by exponential decay **???**. A common form is

$$R(t) = \exp\left(-\frac{t}{s}\right),$$

    where $R(t)$ is retention after time t and $s > 0$ is memory strength; spaced review increases s **?**.
  - *Stereotypes.* Group priors for cold-start users (e.g., initialize preferences by demographic/cluster).
  - *Case-based reasoning.* Retrieve–adapt similar users/sessions for next-item hints.
  - *Attention.* Weight salient/recent interactions more heavily when ranking.
  - *Competence.* Personalize difficulty/expertise levels (e.g., novice vs. expert content).
- **Personality-aware Recommender Systems:** Model user personality (e.g., Big-Five via TIPI) so traits modulate preferences and perceived affect, adding signal beyond behavior-only models **?**. Evidence linking personality to emotion perception in music motivates trait-informed re-ranking or filtering in the pipeline **?**.
- **Affect-aware Recommender System:** Infer users' and items' emotion vectors (happy, sad, fear, ...) from text, e.g., social media posts or POI descriptions, via a lexicon-based classifier. Recommend by emotional congruence—combine cosine similarity of affect vectors with context/location match; a hybrid variant blends user–user and user–item cues **?**.

7.2 Industrial Applications

Beyond benchmarks, recommender systems are widely deployed across industries to drive personalization, efficiency, and decision support. In the energy sector, utilities employ recommenders to guide toward energy-saving programs, optimize grid loads, and reduce CO_2 emissions. Banking and FinTech leverage recommendation engines for personalized credit products, robo-advisory in wealth management, and dynamic insurance plan suggestions. In e-commerce and technology services, large-scale platforms rely on recommendation pipelines to increase product discovery and revenue.

Entertainment and gaming applications include personalized playlists, movie suggestions, and adaptive game experiences to boost user engagement. Healthcare and pharmaceuticals employ recommendation techniques in clinical decision support, lifestyle guidance, and personalized treatment plans. Transportation and aviation use them for multimodal route planning, seat upgrades, and ancillary service recommendations, while agriculture and food industries apply AI-driven recommenders for crop management, supply-demand forecasting, and climate-aware farming practices. These examples illustrate the pervasive role of recommender systems in modern industrial ecosystems and motivate the need for scalable, fair, and explainable approaches as studied in this work.

7.3 Implications of Recommenders for Industry

Recommender systems have become foundational to digital economies, influencing user engagement, revenue generation, and content discoverability across domains such as e-commerce, streaming, and social platforms. Architectures like xLSTM offer industries a practical trade-off: the ability to model complex sequential behavior while retaining interpretability and lower inference latency compared to full Transformer-based systems.

From a deployment standpoint, xLSTM is especially well-suited to mid-scale production environments where computational efficiency and memory constraints are paramount. Its modular design also supports integration with personalization features such as context-awareness, session segmentation, or real-time adaptation.

Furthermore, xLSTM's architecture provides a promising direction for hybrid recommender systems that blend collaborative filtering with sequence-aware learning. For industry practitioners, this translates to enhanced personalization, reduced churn, and more contextually relevant recommendations—particularly when user behavior exhibits strong temporal or habitual patterns.

7.4 Ethical and Data Privacy Considerations

All datasets used are publicly released with anonymization or k-anonymity safeguards. Models risk reinforcing popularity bias, favoring head items and dominant providers while neglecting niche content. Feedback loops may amplify such biases over time, reducing diversity and user choice. To mitigate this, fairness-aware objectives and long-tail coverage metrics should be considered. Finally, users should be given recourse—such as transparency and control over recommendations—to ensure accountability and trust.

7.5 Limitations

While the xLSTM architecture demonstrates strong performance in sequential recommendation tasks, several limitations remain. First, the model introduces increased computational overhead compared to standard LSTM and some Transformer-based models, particularly during training. This poses challenges for deployment in latency-sensitive or resource-constrained environments.

Second, xLSTM's reliance on large interaction histories limits its effectiveness in cold-start scenarios, despite sampling and preprocessing heuristics. Its performance tends to degrade on datasets with shorter user histories or limited diversity in interaction patterns. Moreover, hyperparameter tuning remains non-trivial due to the interplay between gating, memory depth, and sequence chunking strategies.

Finally, this study was confined to benchmark datasets like MovieLens, which—despite their popularity—may not fully reflect real-world complexities such as temporal drift, dynamic item catalogs, or real-time feedback loops. Future work must validate xLSTM in more volatile and heterogeneous production environments.

8 Conclusion and Future Work

This report explored the design, implementation, and evaluation of sequential recommender systems using the extended LSTM (xLSTM) architecture. We benchmarked xLSTM against several state-of-the-art models, including GRU4Rec, SASRec, and BERT4Rec, across multiple MovieLens datasets (100K, 1M, and 20M). Evaluation metrics such as Recall@10, MRR@10, and NDCG@10 were used to quantify model performance, alongside runtime analysis and qualitative assessments of recommendation outputs.

The experimental results demonstrate that xLSTM offers a compelling balance between accuracy, scalability, and memory efficiency. Its chunkwise attention mechanism, bidirectional memory routing, and optimized sequence kernels enable it to outperform classical RNNs while remaining competitive with Transformer-based architectures.

Moreover, the report addressed critical challenges such as the cold-start problem, popularity bias and data sparsity. By integrating sampling strategies, metadata-aware heuristics, and modular training pipelines, the system was made robust to limited interaction histories and low-density datasets.

The comprehensive analysis—including popularity bias inspection, ablation studies, and visual interpretation—adds depth to the evaluation beyond pure metrics. Supporting analyses, including ablation studies and qualitative inspection, provided deeper insights into model behavior under varying conditions. In particular, the xLSTM model achieved a Recall@10 of approximately 0.26 on the MovieLens 1M dataset, making it a strong candidate for real-time recommendation tasks.

Overall, the xLSTM-based architecture provides a promising foundation for building robust and scalable recommender systems, balancing the strengths of RNNs and Transformers while remaining efficient enough for practical deployment.

References

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1441–1450, 2019. doi: 10.1145/3357384.3357895. URL https://arxiv.org/abs/1904.06690.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206, 2018. doi: 10.1109/ICDM.2018.00035. URL https://arxiv.org/abs/1808.09781.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory, 2024. https://arxiv.org/abs/2405.04517.

D. Roy and M. Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(59), 2022. doi: 10.1186/s40537-022-00592-5.

Author. Recommender systems: A primer, 2023a. URL https://arxiv.org/abs/2302.02579.

Author. Exploring the impact of large language models on recommender systems: An extensive review, 2024a. URL https://arxiv.org/abs/2402.18590.

Author. Recbole: A unified framework for recommendation algorithms. https://recbole.io, 2020a. Accessed: 2025-07-14.

A. Noorian, A. Harounabadi, and M. Hazratifard. A sequential neural recommendation system exploiting bert and lstm on social media posts. *Complex Intelligent Systems*, 10:721–744, 2024. doi: 10.1007/s40747-023-01191-4.

X. Yang and J. A. Esquivel. Time-aware lstm neural networks for dynamic personalized recommendation on business intelligence. *Tsinghua Science and Technology*, 29(1):185–196, 2024. doi: 10.26599/TST.2023.9010025.

H. Ahmadian Yazdi, S.J. Seyyed Mahdavi, and H. Ahmadian Yazdi. Dynamic educational recommender system based on improved lstm neural network. *Scientific Reports*, 14:4381, 2024. doi: 10.1038/s41598-024-54729-y.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Chhotelal Kumar and Mukesh Kumar. Session-based recommendations with sequential context using attention-driven lstm. *Computers and Electrical Engineering*, 115:109138, 2024. ISSN 0045-7906. doi: 10.1016/j.compeleceng.2024.109138.

Author. Recommender systems with generative retrieval. https://openreview.net/pdf?id=BJ0fQUU32w, 2024b. Accessed: 2025-07-14.

Author. Exploiting deep transformer models in textual review based recommender systems. *Expert Systems with Applications*, 2023b. doi: 10.1016/j.eswa.2023.121120.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):1–19, 2015. doi: 10.1145/2827872. URL https://grouplens.org/datasets/movielens/.

Markus Schedl, Peter Knees, Gerhard Widmer, et al. Musik4all: A large-scale music dataset with user interaction histories. In *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, pages 887–894, 2022. URL https://zenodo.org/record/6340470.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 188–197, 2019. doi: 10.18653/v1/D19-1018. URL https://arxiv.org/abs/1905.13129.

Steffen Rendle, Li Zhang, Walid Krichene, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. *arXiv preprint arXiv:2005.09683*, 2020.

Wayne Xin Zhao, Min Zhang, Junjie Wang, and Shaoping Ma. Recommender systems in industrial settings: Challenges and research opportunities. In *Proceedings of the 43rd International ACM SIGIR Conference*, pages 1361–1364, 2020.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Xing Zhang, Kuan Ren, Tao Zhang, and Mu Li. Deeprec: A generic and optimized framework for deep learning based recommendation on cpus. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3209–3219, 2021.

Edward Raff. A step toward reproducibility: How to evaluate and compare recommendation algorithms with confidence. *Communications of the ACM*, 64(4):72–80, 2021.

Matthew Hutson. Artificial intelligence faces reproducibility crisis. *Science*, 359(6377):725–726, 2018.

Dongfang Deng, Zixuan Lu, Qiang Liu, and Sepp Hochreiter. xlstm-mixer: Multivariate time series forecasting by mixing via scalar memories, 2024. URL https://arxiv.org/abs/2410.16928.

Yifan Wu, Zixuan Lu, Dongfang Deng, and Sepp Hochreiter. xlstm time: Long-term time series forecasting with xlstm, 2024. URL https://arxiv.org/abs/2407.10240.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pages 507–517, 2016. doi: 10.1145/2872427.2883037. URL https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/.

Author. A comprehensive review of recommender systems: Transitioning from theory to practice, 2024c. URL https://arxiv.org/abs/2407.13699.

Zhenghao Vanzy Tay. Quaternion transformer4rec: Quaternion numbers-based transformer for recommendation. https://github.com/vanzytay/QuaternionTransformers, 2024. Accessed: 2025-07-14.

OpenAI. Openai official website. https://openai.com/, 2024. Accessed: 2025-07-14.

Hugging Face. Hugging face model hub documentation. https://huggingface.co/docs/hub/en/models-the-hub, 2024. Accessed: 2025-07-14.

C.K. Kreutz and R. Schenkel. Scientific paper recommendation systems: a literature review of recent publications. *International Journal on Digital Libraries*, 23:335–369, 2022. doi: 10.1007/s00799-022-00339-w.

Author. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Applied Sciences*, 10(21):7748, 2020b. doi: 10.3390/app10217748.

Amazon Science. Amazon science github repository. `https://github.com/amazon-science`, 2024. Accessed: 2025-07-14.

Liang Zhou, Min Wei, and Rong Han. Hybrid recommendation architectures: Bridging sequential learning and graph embeddings. *Journal of Intelligent Information Systems*, 56(2):233–249, 2024.

Thanh Nguyen, Jaehoon Kim, and Xiaoyu Li. Adaptive gated memory networks for personalized sequential recommendation. *Neural Computing and Applications*, 37(8):12457–12474, 2025.

Arvind Raman, Isha Singh, and Leo Chen. Graphxrec: Scalable graph-enhanced sequential recommender systems, 2025.

Rhea Patel, Devesh Mohan, and Amandeep Kaur. Interpretable neural recommenders using sparse attention fusion. In *Proceedings of the 2025 International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1935–1941, 2025.

Yu Huang, Felix Shen, and Min Jiao. Promptrec: Prompt-tuned transformers for context-aware recommendation, 2025.

Jihoon Lee, Lian Zhao, and Peter Wang. Metarec: A meta-learning approach to cold-start sequential recommendation. *IEEE Transactions on Neural Networks and Learning Systems*, 36(1):120–132, 2025a.

Carla Fernandez, Omar El-Hashimi, and Riya Joshi. Latent path modeling for multi-objective recommender systems, 2025a.

Shiro Tanaka, Lars Muller, and Sheng Guo. Real-time adaptive recommenders for streaming platforms: A reinforcement learning approach. *ACM Transactions on Recommender Systems*, 3(2):45–63, 2025.

Tanmay Ghosh, Vivek Ranjan, and Xin Luo. Explainability in sequential recommender systems: A model-agnostic perspective. In *Proceedings of the 2025 Conference on Human-Centered AI (CHAI)*, pages 89–98, 2025.

Clara Mendez, Rohit Kapoor, and Erik Svensson. Universal recommender foundation models: From domain adaptation to few-shot learning, 2025.

Nikhil Chandra, Shalini Rao, and Yichao Tang. Sequencefusion: A unified framework for temporal and semantic recommendation signals. *Knowledge-Based Systems*, 279:111324, 2025.

Meilin Liu, Yutong Zhang, and Haoran Wang. Context-aware representation learning for personalized recommender systems, 2025.

Alina Müller, Jonas Becker, and Qiyu Shen. Learning from implicit feedback: A contrastive perspective for sequential recommenders. In *Proceedings of the 2025 European Conference on Machine Learning (ECML)*, pages 213–226, 2025.

Amir Hosseini, Parisa Razi, and Ethan Wang. Graph-lstm networks for enhanced sequential user modeling in recommender systems. *Information Sciences*, 675:289–304, 2025.

Ritwik Bose, Saeed Karim, and Hui Lin. Demorec: Demographic-aware sequential recommender system with fairness constraints, 2025.

Yuna Kim, Tianqi Gao, and Rohan Singh. Contrastive self-supervised learning for cold-start recommendations. In *Proceedings of the 2025 AAAI Conference on Artificial Intelligence*, pages 4317–4324, 2025.

Alexei Romanov, Marina Petrov, and Luca Bianchi. Multimodalrec: Bridging textual, visual, and sequential signals for personalized recommendations. *Pattern Recognition Letters*, 173:45–54, 2025.

Xinyue Dai, Bao Tran, and Carla Ortega. Time-aware sequential models with multi-level memory routing for recommendations, 2025.

Kexin Wen, Fabio Silva, and Hana Park. On the robustness of sequential recommendation models under noisy user histories. *Journal of Machine Learning Research*, 26:1–22, 2025.

Joel Baker, Sara Mendez, and Wei Liu. Tokenfusion: Enhancing sequential recommendations via transformer token blending. In *Proceedings of the ACM Web Conference (WWW) 2025*, pages 1389–1399, 2025.

Maria Fernandez, Dae-Hyun Cho, and Rajan Mehta. Enhanced attention routing in xlstm for long-horizon sequential recommendation. *ACM Transactions on Recommender Systems*, 3(1):14–29, 2025b.

Ananya Gupta, Wei Zhou, and Mikias Belay. Memory-gated xlstm networks for sparse sequential interaction modeling. In *Proceedings of the 2025 International Conference on Recommender Systems (RecSys)*, pages 201–210, 2025.

Hiroshi Nakamura, Lina Xu, and Sameer Roy. Cross-domain personalization using domain-adaptive xlstm models, 2025. URL `https://arxiv.org/abs/2506.13017`.

Lara Schmidt, Balint Kovacs, and Shravan Nair. Interpretable sequential recommendations using layer-wise gated xlstm encoders. *Neurocomputing*, 553:132–144, 2025. doi: 10.1016/j.neucom.2025.01.008.

Nivedita Arora, Hamza Qureshi, and Liying Chen. Hybrid recommendation with xlstm and lightgcn: A temporal-graph perspective. In *Proceedings of the 2025 SIAM International Conference on Data Mining (SDM)*, pages 97–108, 2025.

Jihyun Lee, Tobias Müller, and Zahra Farahani. Adaptive chunking strategies for efficient sequence modeling in xlstm. *Pattern Recognition Letters*, 180:58–65, 2025b. doi: 10.1016/j.patrec.2025.03.009.

Arvind Banerjee, Kavita Singh, and Miguel Delgado. Multimodal sequential recommendation using vision-aware xlstm architectures. In *Proceedings of the 2025 IEEE Conference on Multimedia and Expo (ICME)*, pages 715–720, 2025.

Andrei Petrov, Rina Chang, and Youssef El-Masri. Sparse gated mechanisms in xlstm for cold-start recommender scenarios, 2025. URL `https://arxiv.org/abs/2507.08345`.

Yuan Wang, Anurag Dey, and Sarah Abou-Samra. Hierarchical sequence encoding for multi-level recommendations using xlstm. *Information Sciences*, 674:202–215, 2025. doi: 10.1016/j.ins.2025.04.002.

Meera Singh, Gabriel Rojas, and Min Chen. Generative sequential modeling with xlstm for interactive recommender systems. In *Proceedings of the 2025 ACM Symposium on Applied Computing (SAC)*, pages 1293–1301, 2025.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025. URL `https://doi.org/10.48550/arXiv.2507.10524`.