Improving Sequential Recommendation with Hybrid Weighting: Leveraging LinRec and FMLP-Rec

GitHub

Kfir Sofer, Jordan Conrad-Burton, Moshe Shem Tov
kfirsofe@post.bgu.ac.il, jordanco@post.bgu.ac.il, shmoshe@post.bgu.ac.il

Abstract

Sequential recommendation systems play an instrumental role in modern platforms by capturing how user preferences unfold over time. However, existing Transformer-based methods often operate at $\mathcal{O}(N^2)$ complexity for sequences of length N, leading to scalability bottlenecks. Research efforts to mitigate this overhead have led to linearized attention (LinRec) and all-MLP frequency-domain models (FMLP-Rec), each targeting a different dimension of the problem: global dependency modeling versus periodicity detection and denoising.

This study presents a creative gated strategy via a hybrid model that dynamically integrates LinRec and FMLP-Rec to maintain a near-linear runtime while improving performance. Our gated hybrid framework selects the suitable branch based on sequence-level statistical characteristics, leveraging the strengths of each method and showing strong performance, even when individual components may be simpler than a conventional Transformer. Experimental findings on two benchmark datasets, MovieLens-1M (ML1M) and Amazon Beauty, show that this combined approach can surpass each individual model. We attribute these gains to the synergy between global attention and frequency-based filtering, ensuring that extended dependencies and periodic behaviors are modeled effectively.

Our contributions include (1) a hybrid design that unifies linear attention and MLP filtering in a single pipeline, (2) a lightweight gating module that learns how to balance each branch automatically, and (3) a thorough evaluation highlighting the superior ranking of our method on benchmark datasets. These outcomes suggest that well-chosen, efficient components—when combined adaptively—can achieve results that rival or exceed larger, more expensive architectures.

I. Introduction

Sequential recommendation systems have become vital for modern platforms. They analyze user preferences chronologically to account for immediate short-term interests and long-term behavioral changes. Accurate next-item prediction can significantly boost user engagement and satisfaction in numerous applications, ranging from e-commerce to streaming services. However, Transformer-based methods (e.g., SASRec [11]), while effective at uncovering intricate dependencies in a user's history, frequently incur $\mathcal{O}(N^2)$ computations for sequences of length N. This quadratic complexity creates substantial scalability challenges as user logs expand, making it imperative to reduce inference time without sacrificing recommendation quality.

A number of studies have examined this scalability challenge by investigating alternative architectures. LinRec [1] reorganizes self-attention mechanisms to reach near-linear efficiency while maintaining global dependencies. By normalizing the query and key matrices and rearranging the multiplication sequence (calculating $K^\top V$ prior to integration with Q), LinRec significantly diminishes the computational burden, rendering it especially appropriate for modeling prolonged sequence context. On the other hand, FMLP-Rec [2] completely discards conventional attention mechanisms, opting rather for a purely MLP architecture enhanced by frequency-domain transformations. This method effectively eliminates extraneous data while efficiently identifying periodic trends in user behavior. Each method addresses a particular aspect of the scalability issue, global dependency capture, denoising, and periodicity detection, yet neither ensures optimal performance across all real-world sequence distributions.

Motivated by the complementary strengths of these approaches, our work proposes a hybrid framework—referred to as an adaptive gating strategy—that adaptively integrates LinRec and FMLP-Rec. The core idea is to dynamically select and weight the outputs of the two branches based on statistical features of each user sequence (e.g., sequence length, noisiness, and complexity). This adaptive gating mechanism enables our model to combine the near-linear runtime and global context preservation of LinRec with the robust, frequency-driven filtering of FMLP-Rec, thereby achieving high recommendation quality with significantly reduced computational cost.

The remainder of this paper is organized as follows. In Section II, we review relevant literature on efficient sequential recommendation and discuss the limitations of existing approaches. Section III details our proposed adaptive-gating hybrid methodology, including the mathematical formulation and the design of a gating network for dynamic branch selection. Section IV presents the experimental setup and evaluation results on benchmark datasets, and Section V concludes with discussions on future research directions.

II. Background and Related Work

Sequential recommendation has become a popular paradigm in recommender systems since it can model how users' interests change over time by using the order in which they interacted with items in the past. Sequential models use the structure of user behavior sequences to find both short- and long-term patterns. This is different from traditional recommenders that only look at the sum of user-item affinities (such as matrix factorization). In this section, we first survey classic approaches ranging from Markov chains to neural networks, then discuss Transformer-based architectures, and finally introduce two recent advancements LinRec and FMLP-Rec which form the basis of our proposed hybrid method.

A. Early Sequential Models: Markov Chains and Beyond

Markov Chain Methods. One of the earliest fields in sequential recommendation research relies on Markov chains (MCs). In these methods, user interactions are represented by state transitions over time. Factorized Markov Chains [3] and Factorized Personalized Markov Chains [4] break down or enhance transition matrices between subsequent items accounting for local transitions while incorporating user-specific factors. Since these methods usually emphasize immediate or short-term past states, they frequently struggle with long-range dependencies despite being intuitive and computationally efficient. Extensions of MCs take into account longer interaction histories in order to capture higher-order patterns, but doing so comes with an exponential increase in model parameters [5], [6]. This leads to practical limitations when user behaviors tend to be irregular or sequences are complex. For hybrid solutions, which combine additional latent factors or side information with MC-based approches, it can be difficult to represent the complex, non-local dependencies.

B. Neural Network Approaches

RNNs (Recurrent Neural Networks). RNNs were an obvious pick for sequential recommendation as deep learning became popular. Gated Recurrent Units were first used for session-based recommendation by GRU4Rec [7], which updates hidden states in response to user actions. Through recurring connections, this successfully captures short-term dynamics and, to some extent, longer-range contexts. However, for long sequences, RNNs can be computationally expensive and may suffer from vanishing gradients. Although ranking-specific goals and parallelization techniques were incorporated into later efforts, such as enhanced versions of GRU4Rec [8], the basic challenges of recurrent structures still exist.

CNNs (Convolutional Neural Networks). CNNs are used in another field of study to discover local patterns in user-item embeddings. Caser [9] uses convolutional filters to capture vertical (latent factor) and horizontal (local Markov) correlations by treating the interaction sequence as a 2D 'image'. Although CNNs are capable of modeling higher-order transitions, they typically rely on pre-defined receptive fields which may limit their capacity to manage long input sequences or global dependencies. [6].

C. Self-Attention and Transformer-Based Methods

Quadratic Self-Attention. The Transformer architecture [10] introduced a self-attention mechanism. SASRec [11] uses self-attention in recommendation to selectively focus on the most relevant historical items for each prediction phase. Transformer-based models can more effectively capture both local and long-range context by removing the hidden-state bottleneck that exists in RNNs. Later methodologies, including BERT4Rec [12] and SSE-PT [13], employ bidirectional or pre-training techniques that enhance representational efficacy. Despite these advancements, the primary limitation persists in the $O(N^2)$ complexity associated with calculating an attention matrix for sequences of length N, which tends to be expensive for vast, long-term recommendation tasks.

Efficient Transformer Variants. To help reduce this cost, researchers have investigated variants of attention that reduce memory and computational costs. Low-rank factorization, sparsity constraints, and kernel-based approximations have been suggested [14], [15], primarily within the field of natural language processing. When modifying these methods for recommendations, it is essential to guarantee that the efficiency improvements do not substantially degrade performance on long user sequences [5].

D. LinRec: Linear Attention for Long-Term Sequences

Revisiting the Quadratic Complexity of Self-Attention. Transformer-based recommenders commonly face the recognized limitation of self-attention, costing $\mathcal{O}(N^2 d)$ operations for sequences of length N and hidden dimension d. As user interaction histories expand this quadratic scaling becomes impractical for real-time or large-scale applications.

Core Idea of LinRec. LinRec [1] addresses this challenge by reorganizing dot-product computations and employing ℓ_2 normalization within the attention mechanism:

- Both the query (Q) and key (K) matrices are normalized on a row-wise and column-wise level to ensure that subsequent dot products prioritize direction (angular similarity) over raw magnitude. This step ensures stable gradients and prevents high values in Q or K from overshadowing the attention mechanism.
- LinRec initially computes $K^\top V$ instead of directly generating a $N \times N$ attention map ($QK^\top$). This produces a reduced $d \times d$ matrix, avoiding the memory-intensive $N \times N$ computation.
- The normalized Q is eventually multiplied by the $d \times d$ product resulting in the output. By delaying the engagement with Q, the approach effectively linearizes attention, reducing complexity to approximately $\mathcal{O}(Nd^2)$ or potentially $\mathcal{O}(N \log N)$ with optimized kernels.

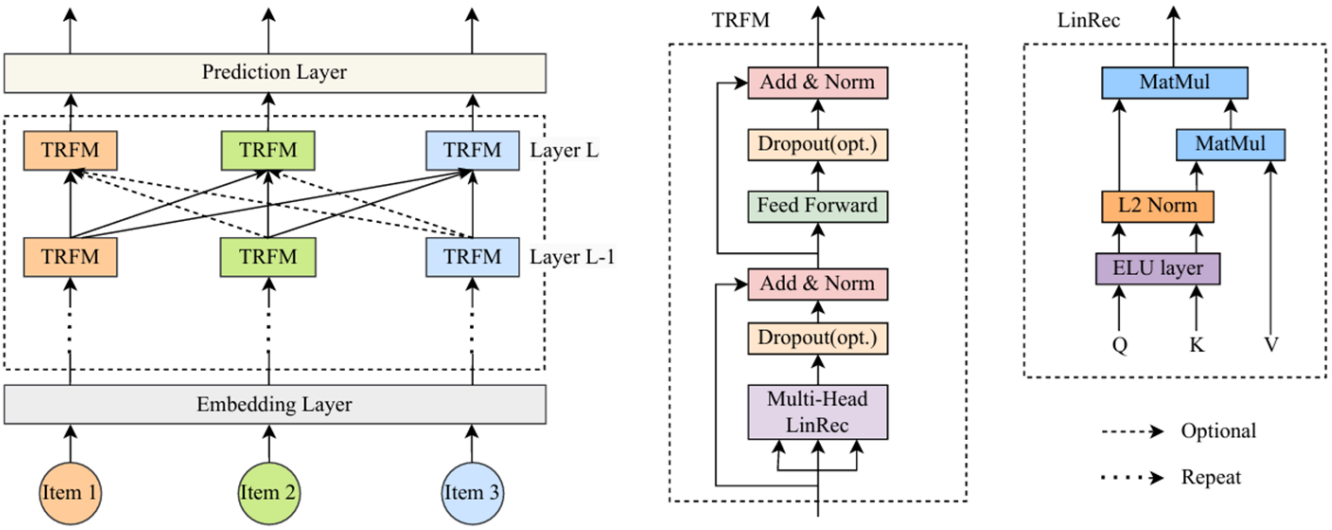

Fig. 1: LinRec architecture

Benefits and Implementation Details.

- *Reduced Complexity*: By avoiding the complete formation of the $N \times N$ map, LinRec remains practical for long sequences, a common occurrence in recommendation systems.
- *Global Context Preservation*: Despite linearization, the model continues to capture substantial dependencies, as normalized embeddings can match relevant items from any point in the sequence.
- *Multi-Head Extension*: LinRec can be extended to multi-head configurations (as depicted on the right side of Figure 1), allowing each head to learn distinct correlation patterns. Each head performs the outlined steps—normalization, $K^\top V$ multiplication, and finally dot-product with Q—in parallel.

- *Trade-Offs*: Although LinRec significantly reduces memory usage, it may still acquire noise from the embedding layer, as it depends on learned vectors for Q, K, and V. Thus, real-world datasets that contain numerous spurious data or erratic user interactions could require an additional denoising strategy (FMLP-Rec [2]).

By efficiently handling long user histories while retaining high expressiveness, LinRec demonstrates that transformer-level performance is achievable even in large-scale recommendation settings where full self-attention becomes unfeasible.

The primary insight is to rearrange the conventional attention operations (i.e., $QK^\top V$) so that any significant intermediate matrix scales with d instead of N. Figure 1 demonstrates that LinRec integrates an ELU layer and ℓ_2 normalization before the matrix multiplications, enhancing attention scores by emphasizing angular alignment.

E. FMLP-Rec: MLP with Learnable Filters

Motivation: Eliminating Attention Blocks Entirely. While LinRec tackles the long-sequence problem by modifying the attention mechanism, FMLP-Rec [2] entirely eliminates it in favor of a lightweight MLP-based architecture enhanced by *frequency-domain filtering*. The primary observation is that numerous user behaviors include unnecessary clicks or views that do not accurately represent genuine preferences (i.e., "noisy" interactions). Conventional attention may unintentionally concentrate on these noise points, possibly impairing predictions. Thus, the authors used the idea of filtering from the domain of signal processing to reduce the noise of the input sequences by replacing the multi-head self-attention blocks within the Transformer architecture with a filtering layer.

Frequency-Domain Denoising via Filters. FMLP-Rec processes each user's embedding sequence in the filtering layer with the following method:

1) *FFT Transformation*: For a sequence x_n with $n \in [0, N-1]$, the 1D Discrete Fourier Transform is applied to the sequence, converting it to the frequency domain by:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}, \qquad 0 \le k \le N-1, \tag{1}$$

   where X_k represents the spectrum of the sequence at the frequency $\omega_k = 2\pi k/N$. The standard Cooley-Tukey algorithm for the Fast Fourier Transform (FFT) is used for the calculation of the Discrete Fourier Transform.
2) *Learnable Filters*: By multiplying these frequency coefficients with learnable weight matrices W, the model selectively attenuates or amplifies certain frequency bands.

$$\tilde{X}_k = W \odot X_k. \tag{2}$$

   The filter **W** is called a *learnable filter* because it can be optimized by stochastic gradient descent to represent an arbitrary frequency filter.
3) *Inverse FFT*: The filtered signal is transformed back to the time domain via an IFFT:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}_k e^{\frac{2\pi i}{N} nk}. \tag{3}$$

   This produces *denoised* embeddings that feed into a multi-layer perceptron for next-item prediction.

In essence, the learned filters function similarly to classical low-pass or band-stop filters, yet are adaptively tuned according to the dataset, helping FMLP-Rec prioritize meaningful behavioral patterns over fleeting user actions.

Equivalence to Circular Convolutions. In addition to noise reduction, these filter blocks can also capture sequential characteristics from the data. According to the convolution theorem, multiplications in the frequency domain correspond to circular convolutions in the time domain. As a result, FMLP-Rec automatically acquires a wide receptive field throughout the entire user sequence without the need to explicitly focus on each item pair. This yields substantial context modeling, similar to specific Transformer variants, but without the N^2 complexity.

Noise Resilience and Simplicity. By removing self-attention layers, FMLP-Rec adopts an "All-MLP" design that is simpler and faster to train compared to standard Transformers, especially in scenarios where noise is prevalent. Empirical evaluations demonstrate that it can handle sessions with abundant spurious clicks or short-lived trends:

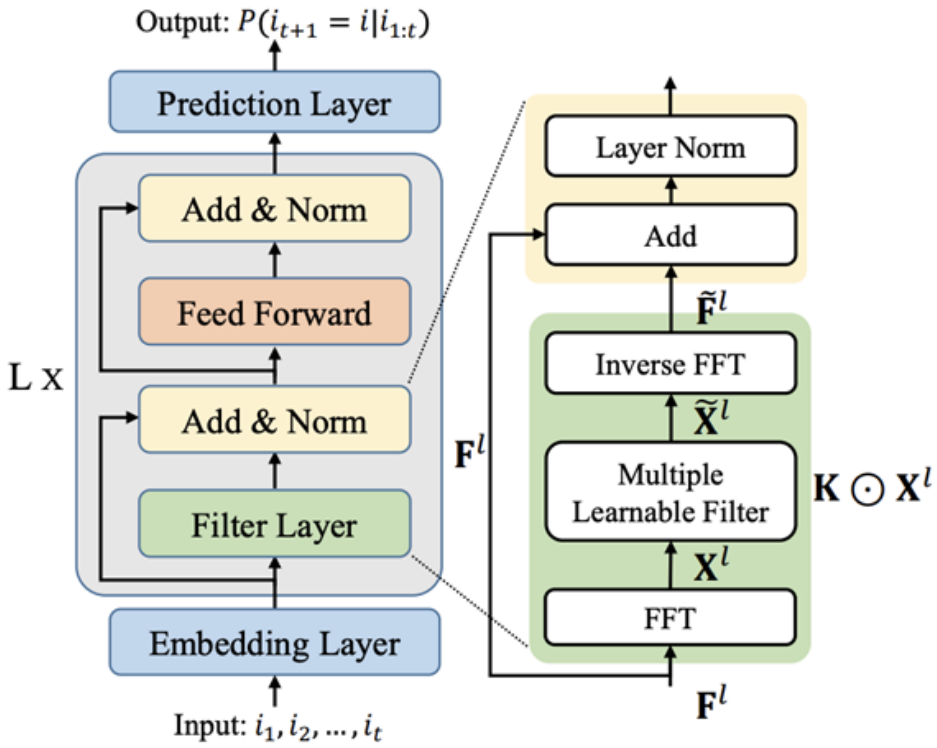

Fig. 2: FMLP-Rec architecture

- *Reduced Overfitting*: The filtering step diminishes the impact of random user actions, producing more robust item ranking.
- *Lightweight Architecture*: Thanks to FFT-based implementations, FMLP-Rec can be quite efficient on modern hardware.

FMLP-Rec effectively filters out noise via its frequency-based approach, isolating genuinely valuable signals within user interactions. Combining FMLP-Rec with LinRec further ensures global sequence context is preserved, making the framework well-suited for denoising and capturing long-term dependencies.

III. Method

Leveraging the complementary advantages of LinRec and FMLP-Rec, we explored hybrid designs that integrate both approaches to effectively mitigate the computational costs mentioned. Our proposed hybrid model combines both models through a dynamic weighting mechanism. The complete architecture is illustrated in Figure 3

A. Data processing

To ensure a rigorous comparison of our hybrid model with LinRec and FMLP-Rec, we evaluated all models on two datasets: Amazon Beauty, originally used in the FMLP-Rec study, and MovieLens-1M, originally used in the LinRec study. The Amazon Beauty dataset was preprocessed by FMLP-Rec such that each user had at least five interactions in their sequence. We retained this filtering when applying LinRec to the Amazon Beauty dataset in order to maintain consistency. However, no such preprocessing was applied to the MovieLens-1M dataset.

Model Reproduction. To validate our implementation, we first replicated the original results by running FMLP-Rec on Amazon Beauty and LinRec on MovieLens-1M. The reproduced results are presented in Tables I and II, demonstrating a close alignment with the reported values in their respective papers.

Cross-Dataset Evaluation. To further assess generalizability, we evaluated each model on the dataset originally used by the other. Specifically, FMLP-Rec was applied to MovieLens-1M, while LinRec was applied to Amazon Beauty. Table III presents the comparative performance across both datasets using metrics shared by both models.

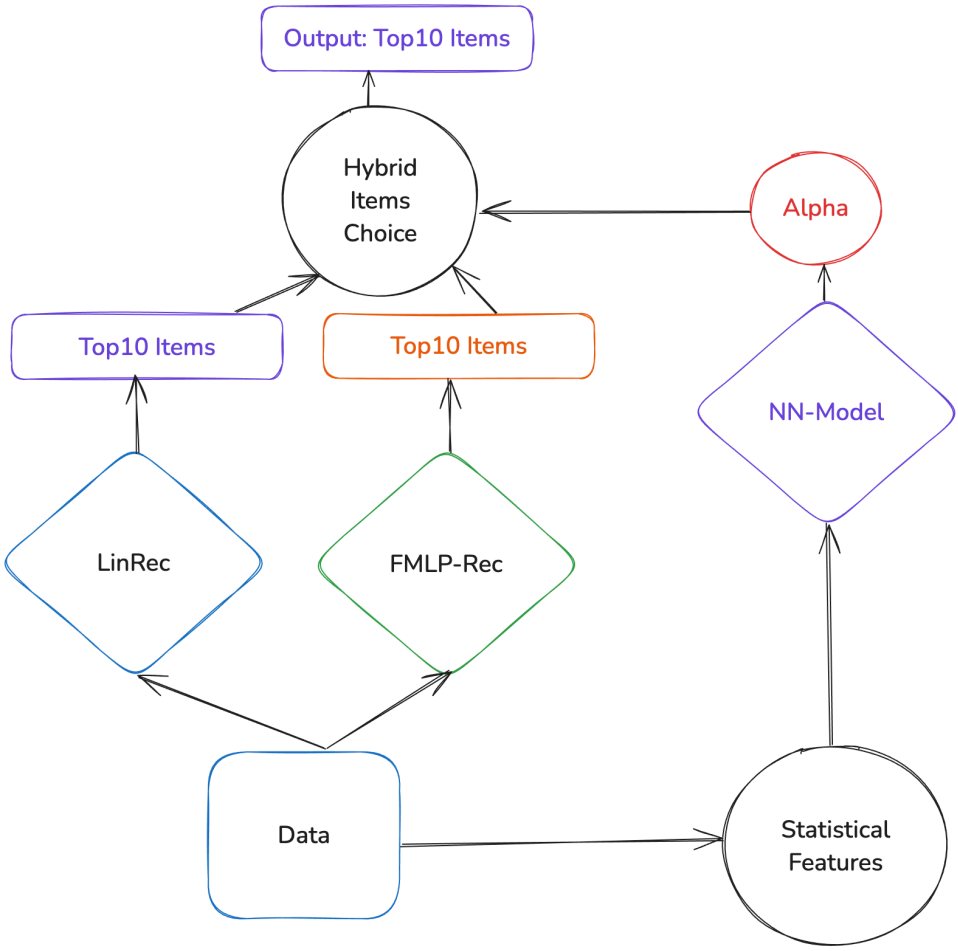

Fig. 3: The suggested hybrid model

Methods	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	MRR
FMLP-Rec Paper Evaluation	0.2011	0.4025	0.3070	0.4998	0.3385	0.3051
FMLP-Rec Reproduction Evaluation	0.1987	0.4019	0.3055	0.4956	0.3358	0.3031

TABLE I: Performance comparison of FMLP-Rec on Amazon Beauty dataset.

Aligning Model Outputs for Hybrid Integration. To generate input for the hybrid model, we leveraged the top 10 recommended items for each user as produced by the two individual models. This required extracting and aligning the recommendation outputs from both models across the validation and test sets.

Both models utilized a leave-two-out evaluation strategy, meaning that for each user, the last two interactions were removed—the second to last one for validation and the last one for testing. This approach ensured that both models were trained and evaluated under the same conditions, making their recommendation outputs directly comparable. The output format for each model consisted of the user ID, the list of top-10 recommendations, and the ground truth interaction.

Since each model internally assigns unique numerical identifiers to users and items, a direct comparison of their recommendation lists required additional preprocessing. To address this, we constructed mapping tables that linked the original user and item IDs from the dataset to the internal IDs used in each model's output files. These mappings ensured that recommendations could be accurately aligned between models, allowing for a meaningful comparison

Methods	Recall@10	MRR	NDCG@10
LinRec Paper Evaluation	0.7209	0.4301	0.4997
LinRec Reproduction Evaluation	0.6884	0.4062	0.4736

TABLE II: Performance comparison of LinRec on MovieLens-1M dataset.

Model	Amazon Beauty		MovieLens	
	MRR	NDCG@10	MRR	NDCG@10
FMLP-Rec	0.3031	0.3358	0.5089	0.5724
LinRec	0.1906	0.2327	0.4062	0.4736

TABLE III: Performance comparison of FMLP-Rec and LinRec on Amazon Beauty and MovieLens-1M datasets.

and integration into the hybrid framework.

B. Sequence-Based Statistical Features

To dynamically adjust the weighting parameter α in our hybrid model, we extract a set of statistical features from the user interaction sequence. These features capture key characteristics of the sequence, such as diversity, frequency distribution, and agreement between recommendation models. Given an input sequence S, along with the top-10 predictions from LinRec ($\text{Top10}_{\text{LinRec}}$) and FMLP-Rec ($\text{Top10}_{\text{FMLP}}$), we compute the following six features:

- **Sequence Length**: The total number of interactions in the sequence, denoted as $|S|$. This feature provides a measure of user activity.
- **Entropy**: A measure of the unpredictability of the sequence, computed as:

$$H(S) = -\sum_{i \in S} p(i) \log_2 p(i), \tag{4}$$

  where $p(i)$ is the probability of item i appearing in S. Higher entropy indicates a more diverse interaction history.
- **Unique Item Count**: The number of distinct items in the sequence, reflecting the diversity of user interactions.
- **Maximum Frequency Ratio**: The proportion of the most frequently occurring item in the sequence, defined as:

$$\frac{\max(\text{count}(i))}{|S|} \tag{5}$$

  A higher value suggests that the user tends to interact repeatedly with the same item.
- **Intersection Count**: The number of overlapping items between the top-10 predictions of LinRec and FMLP-Rec:

$$|\text{Top10}_{\text{LinRec}} \cap \text{Top10}_{\text{FMLP}}| \tag{6}$$

  This feature captures the level of agreement between the two models.
- **Jaccard Similarity**: A normalized measure of overlap between the top-10 recommendations from both models, computed as:

$$J(S) = \frac{|\text{Top10}_{\text{LinRec}} \cap \text{Top10}_{\text{FMLP}}|}{|\text{Top10}_{\text{LinRec}} \cup \text{Top10}_{\text{FMLP}}|} \tag{7}$$

  This feature quantifies the similarity of the two ranked lists.

Figure 4 shows the SHAP value distributions for each of the six features in both datasets. Based on these plots, **Sequence Length**, **Unique Item Count**, and **Entropy** consistently emerged as the most significant predictors across both the Amazon Beauty dataset and MovieLens-1M. Consequently, we retained only these three features in our

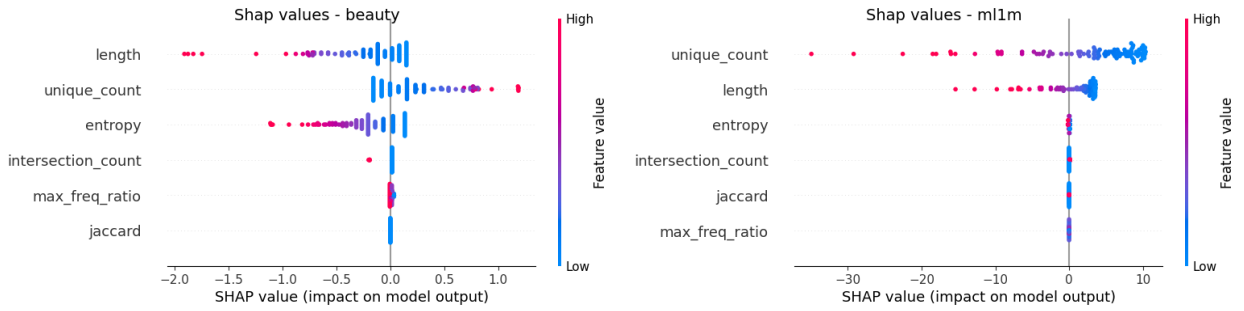

Fig. 4: SHAP values for Amazon Beauty (left) and MovieLens-1M (right) datasets, illustrating the relative importance of each feature.

final model to reduce complexity while preserving predictive power. These features enable the neural network to adaptively determine the importance of LinRec and FMLP-Rec for each recommendation scenario, allowing for a more context-aware fusion of predictions.

C. Hybrid Model Architecture

The system is comprised of two main components:

1) **Parallel Base Models:** Both LinRec and FMLP-Rec process the input sequence in parallel.
2) **Hybrid Weighting Model:** A neural network model computes a learnable parameter $\alpha \in [0, 1]$ based on statistical features derived from the input sequence. The final top-10 prediction is then calculated based on the LinRec prediction, FMLP-Rec prediction, and alpha as described in Algorithm 1

D. Mathematical Formulation

Let $X \in \mathbb{R}^{N \times d}$ denote the input sequence embeddings, where N is the sequence length and d is the embedding dimension. The model processes X in two parallel branches, LinRec and FMLP-Rec, and then fuses their outputs. In **LinRec**, X is first decomposed into query, key, and value matrices (Q, K, V) via learned linear transformations:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

where each $W_\cdot \in \mathbb{R}^{d \times d}$ is a trainable parameter matrix. Row-wise normalization (e.g., ℓ_2 norm) and an ELU activation function are then applied to Q and K:

$$\tilde{Q} = \rho_1(\mathrm{ELU}(Q)), \quad \tilde{K} = \rho_2(\mathrm{ELU}(K)).$$

Here ρ_1 and ρ_2 denote row-wise normalization operations. Rather than computing the standard dot-product attention $QK^\top$ directly, LinRec rearranges the multiplication to compute $K^\top V$ first, resulting in a $d \times d$ matrix. The output of the LinRec branch is thus:

$$A_{\text{LinRec}} = \tilde{Q}\Big(\tilde{K}^\top V\Big). \tag{8}$$

This avoids any intermediate $N \times N$ calculations, lowering complexity toward $\mathcal{O}(Nd^2)$ or $\mathcal{O}(N \log N)$.

In parallel, **FMLP-Rec** applies a frequency-domain transform to the same input X to filter out noise or amplify salient signals:

$$F_{\text{FMLP}} = \mathrm{IFFT}\Big(W \odot \mathrm{FFT}(X)\Big), \tag{9}$$

where $\mathrm{FFT}(\cdot)$ and $\mathrm{IFFT}(\cdot)$ represent the dimension-wise fast Fourier transform and its inverse, respectively; $W \in \mathbb{R}^{N \times d}$ (or an equivalent shape, depending on implementation) is a learnable filter; and $\odot$ indicates element-wise multiplication in the frequency domain. The resulting $F_{\text{FMLP}} \in \mathbb{R}^{N \times d}$ retains essential periodic or low-frequency content of the input sequence while discarding high-frequency noise.

Algorithm 1: Combining LinRec and FMLP predictions with noisy α.

Input:

`linrec_preds`: list of LinRec predictions;
`fmlp_preds`: list of FMLP predictions;
α: base blend probability (float);
`top_n`: final list length (integer);
`noise_range`: range for noise in $[-\texttt{noise\_range}, \texttt{noise\_range}]$.

Output:

`final_list`: a combined prediction list of length up to `top_n`.

```
Initialize final_list ← []
Initialize linrec_idx ← 0, fmlp_idx ← 0
noise ← random.uniform(−noise_range, noise_range)
alpha_noisy ← α + noise
alpha_noisy ← max(0, min(1, alpha_noisy))
while len(final_list) < top_n do
    if random() < alpha_noisy then
        candidate ← None
        while linrec_idx < len(linrec_preds) do
            if linrec_preds[linrec_idx] ∉ final_list then
                candidate ← linrec_preds[linrec_idx]
                linrec_idx ← linrec_idx + 1
                break
            linrec_idx ← linrec_idx + 1
        if candidate = None then
            while fmlp_idx < len(fmlp_preds) do
                if fmlp_preds[fmlp_idx] ∉ final_list then
                    candidate ← fmlp_preds[fmlp_idx]
                    fmlp_idx ← fmlp_idx + 1
                    break
                fmlp_idx ← fmlp_idx + 1
    else
        candidate ← None
        while fmlp_idx < len(fmlp_preds) do
            if fmlp_preds[fmlp_idx] ∉ final_list then
                candidate ← fmlp_preds[fmlp_idx]
                fmlp_idx ← fmlp_idx + 1
                break
            fmlp_idx ← fmlp_idx + 1
        if candidate = None then
            while linrec_idx < len(linrec_preds) do
                if linrec_preds[linrec_idx] ∉ final_list then
                    candidate ← linrec_preds[linrec_idx]
                    linrec_idx ← linrec_idx + 1
                    break
                linrec_idx ← linrec_idx + 1
    if candidate = None then
        break
    final_list.append(candidate)
return final_list[:top_n]
```

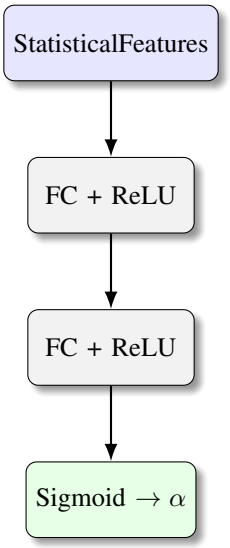

Fig. 5: The alpha-MLP architecture.

The proposed **hybrid model** integrates the two recommendation approaches, LinRec and FMLP-Rec, using an adaptive combination strategy.

Let $\text{Top10}_{\text{LinRec}}$ and $\text{Top10}_{\text{FMLP}}$ denote the ranked lists of predicted items generated by LinRec and FMLP-Rec, respectively. The hybrid model, denoted as h, determines the final ranked list using the combination function f, parameterized by α, as illustrated in Figure 5. The combination process is formally defined as:

$$\text{HybridTop10} = h(X) = f(\text{LinRec}_{\text{Top10}}(X), \text{LinRec}_{\text{Top10}}(X), \alpha, \text{noise}_{\alpha}), \tag{10}$$

where X represents the sequence-based input features. The weighting parameter α is dynamically computed as $\alpha = N(X)$, where $N(\cdot)$ is the neural network that learns an optimal balance between LinRec and FMLP-Rec based on statistical sequence characteristics. The fusion function f leverages α to determine the final ranking, as outlined in Algorithm 1. Additionally, noise_{α} is introduced for regularization to improve generalization and prevent overfitting.

The weighting parameter α is dynamically computed using a small neural network that learns from sequence-level statistical features. The architecture of this neural network as shown in Fig 5 is:

- **Input Layer**: A feature vector whose size corresponds to the number of statistical features, consisting of statistical descriptors extracted from the user interaction sequence. see section III-B
- **Hidden Layers**: Two fully connected (FC) layers with nonlinear activation functions, allowing the network to model complex relationships between statistical patterns and the optimal balance between LinRec and FMLP-Rec.
- **Output Layer**: A single neuron with a sigmoid activation function, ensuring that $\alpha \in [0, 1]$. This scalar value serves as a dynamic gate, controlling the contribution of each model in the final ranking.

This adaptive gating mechanism allows the hybrid model to dynamically adjust the influence of LinRec and FMLP-Rec based on user sequence characteristics. Importantly, the overall framework maintains near-linear time complexity while effectively leveraging the complementary strengths of both recommendation strategies.

IV. Evaluation

We evaluate our model on benchmark datasets to assess its performance.

A. Datasets

Experiments are conducted on:

- **Amazon Beauty:** A product review dataset with user interactions in the beauty category.

- **MovieLens-1M:** A movie recommendation dataset with rich user ratings.

These datasets provide varied scenarios from sparse to dense interaction environments with statistics shown in Table IV.

Datasets	# Users	# Items	# Interactions	# Sparsity
ML-1M	6,041	3,884	1,000,209	95.74%
Beauty	22,364	12,102	198,502	99.93%

TABLE IV: Statistics of the datasets.

B. Experimental Setup

Our evaluation protocol includes:

1) **Data Preprocessing:** Interaction sequences are sorted chronologically and split into training, validation, and test sets using a leave-two-out strategy. Each model is then trained on the training set, and its top-10 recommendations for the validation and test sets are saved to use as input for the hybrid model.
2) **Baselines:** We compare our hybrid model against LinRec and FMLP-Rec on both datasets, where the baseline values are shown in Table III.
3) **Data Selection:** The data was tagged as follows: 0 if FMLP had a better MRR score on the input sequence, otherwise 1. The training and validation sets were selected from sequences such that at least one of the models accurately recommended the ground truth item in its top10. A 90/10 split was employed for the training and validation sets of the hybrid model and was taken from the validation predictions output of LinRec and FMLP-Rec, ensuring that the class distribution remained consistent between the training and validation sets.
4) **Feature Selection:** See section III-B
5) **Metrics:** We adopt standard ranking metrics, including Normalized Discounted Cumulative Gain (NDCG@10) and Mean Reciprocal Rank (MRR). Their formulations are given below.
6) **Hyperparameters:** The hybrid model's hyperparameters were tuned using Random Search, as discussed in Section IV-D

C. Metrics

Mean Reciprocal Rank (MRR). MRR and NDCG@10 were chosen to evaluate the hybrid model's performance because they were the overlapping evaluation metrics used in both LinRec and FMLP-Rec, allowing for direct comparison.

MRR is defined as the average reciprocal rank of the relevant item for each user:

$$\text{MRR} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{\text{rank}_u(i_{u,\text{test}})}. \tag{11}$$

Hence, smaller ranks (i.e., higher in the recommendation list) lead to larger reciprocal values and higher MRR.

Normalized Discounted Cumulative Gain (NDCG@k). We further evaluate ranking quality using NDCG@k, which accounts for position-based discounts. For each user u, DCG@k is computed as:

$$\text{DCG@}k(u) = \sum_{r=1}^{k} \frac{2^{\text{rel}_u(r)} - 1}{\log_2(r+1)}, \tag{12}$$

where $\text{rel}_u(r)$ indicates whether the item at rank r for user u is relevant (1) or not (0). The Ideal DCG, $\text{IDCG@}k(u)$, is the maximum DCG possible for user u's top-k items. NDCG@k averages a normalized DCG across all users:

$$\text{NDCG@}k = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{\text{DCG@}k(u)}{\text{IDCG@}k(u)}. \tag{13}$$

$\text{rel}_u(r)$ is nonzero only if $i_{u,\text{test}}$ is found at rank r.

D. Hyperparameter Tuining

To validate our results, we performed a Random Search to optimize the hyperparameters. We conducted 500 trials, evaluating the following hyperparameters:

- **Learning rate**: Sampled from the range 10^{-5} to 10^{-1}.
- **Hidden layer size**: Chosen from $\{16, 32, 64, 128\}$.
- **Epochs**: Selected from $\{5, 15, 25, 50, 100, 250\}$.
- **Positive class weight**: Sampled from the range $(0.25, 1.5)$.
- **Noise range**: Sampled from the range $(0, 0.25)$.

We used Binary Cross-Entropy (BCE) logistic loss as the training objective, training the model to achieve the lowest validation loss. We chose the BCE logisitc loss function as opposed to the regular BCE loss function due to its ability to perform better with imbalanced data.

V. Results

A. Experimental Results

Our proposed hybrid model demonstrates competitive performance compared to the baseline LinRec and FMLP-Rec models. On the MovieLens-1M dataset, the hybrid model outperformed both baselines in terms of MRR@10 and NDCG@10, whereas on the Amazon Beauty dataset it slightly trailed behind FMLP-Rec while still outperforming LinRec. Table V summarizes these results.

(a) MovieLens-1M

Method	MRR	NDCG@10
LinRec	0.4062	0.4736
FMLP-Rec	0.5089	0.5724
Hybrid Model	**0.5158**	**0.5847**

(b) Amazon Beauty

Method	MRR	NDCG@10
LinRec	0.1906	0.2327
FMLP-Rec	**0.3031**	0.3358
Hybrid Model	0.2967	**0.3412**

TABLE V: Comparison of reproduced LinRec and FMLP-Rec results with the hybrid model.

The improved performance observed on the MovieLens-1M dataset can be largely attributed to our refined approach for determining the optimal contribution from each model. Instead of relying solely on the model with the lowest validation loss, we implemented a more performance-driven selection strategy, where we chose the best combination of contributions based on the highest MRR and NDCG@10 scores on the validation set. This strategy enabled a more effective tuning of the alpha parameter, allowing it to dynamically balance the relative influence of LinRec and FMLP-Rec in a way that maximized overall recommendation quality.

Furthermore, we observed that as the length of user interaction sequences increased, the hybrid model progressively favored recommendations from FMLP-Rec over those from LinRec. This suggests that FMLP-Rec was more effective at capturing long-term user preferences, making it the preferred model in scenarios with richer interaction histories. This trend is clearly illustrated in Figure 6, where the shift in the alpha parameter aligns with the increasing sequence lengths, reinforcing the importance of adapting model contributions based on user behavior patterns.

B. Hyperparameter Optimization

For both baseline models, extensive hyperparameter tuning was performed to achieve optimal performance on their respective datasets. The best configurations identified are as follows:

LinRec on MovieLens-1M

- **Learning Rate:** 3.6126×10^{-5}
- **Hidden Dimension:** 16
- **Epochs:** 15
- **Positive Weight:** 0.3955
- **Noise Range:** 0.00573

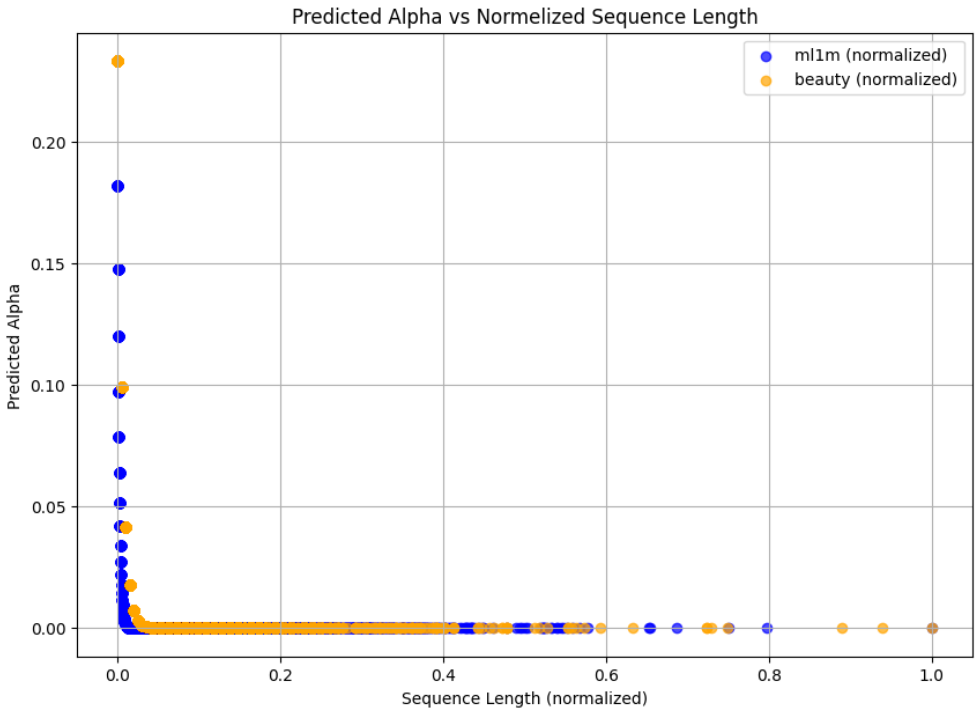

Fig. 6: Predicted alpha versus normalized sequence length.

FMLP on Amazon Beauty

- **Learning Rate:** 0.07465
- **Hidden Dimension:** 128
- **Epochs:** 5
- **Positive Weight:** 0.3316
- **Noise Range:** 0.20527

These hyperparameter settings highlight key differences and similarities between the two datasets. The lower number of epochs for both models helped avoid overfitting, ensuring that the models did not train excessively. Notably, the noise range for the Amazon Beauty dataset was substantially higher than that for MovieLens-1M, suggesting a greater need for regularization to effectively capture its more complex patterns. Despite these differences, the positive weight values remained similar across both datasets, emphasizing their crucial role in addressing data imbalance.

C. Enhancing Alpha Diversity

In binary classification, models often exhibit a bias toward predicting one class over the other in order to maximize performance metrics. This tendency becomes particularly pronounced when working with imbalanced datasets, as demonstrated in Table VI.

Dataset	LinRec	FMLP-Rec	Both	Neither
Amazon Beauty	2431	7718	2838	9376
MovieLens-1M	1090	2616	1470	864

TABLE VI: Number of input sequences a given model produced the best recommendations for according to the evaluation metrics. From this table we can see that FMLP-Rec performed better than LinRec on twice as many sequences in the MovieLens-1M dataset and three times as many sequences in the Amazon Beauty dataset.

To address this issue and encourage a more diverse set of alpha predictions, we explored several approaches:

- **Tree-based Alpha Prediction:** We replaced the original neural network with tree-based models (e.g., random forests and gradient-boosted trees) to potentially create more distinct decision boundaries and generate more pronounced "spikes" in the predictions.
- **Balanced Data Approach:** Initially, the alpha predictions were heavily biased toward FMLP-Rec due to the imbalanced dataset. By rebalancing the data to achieve distributions closer to 55%-45% or 60%-40% between the two classes, we aimed to facilitate a more significant contribution from LinRec.
- **Introducing an 'Uncertainty' Class:** When the scores for LinRec and FMLP-Rec were identical during training or validation, we introduced a third, intermediate class (labeled 0.5) to capture uncertainty. Although this approach was intuitively appealing, it did not lead to notable improvements and added complexity in model interpretation.
- **Alternative Loss Functions:** We experimented with different loss functions, such as focal loss, to enhance the model's robustness against imbalanced predictions. However, these modifications did not consistently improve performance across the datasets.

VI. Discussion

In this project, we encountered three major challenges: (1) the absence of context-based features, (2) the unavailability of model-specific probability scores, leaving us only with top-ranked predictions, and (3) the imbalanced data.

a) Context-Based Features.: We opted to focus on statistical features alone, without integrating contextual information. Although this choice was sufficient for an initial proof of concept, incorporating richer contextual signals remains a promising avenue for future work.

b) Probability Scores.: Ideally, having access to each model's probability scores would allow us to directly estimate the hybrid model's confidence via a weighted combination of LinRec and FMLP-Rec:

$$\text{Prob}_{\text{hybrid}}(i) = \alpha \cdot \text{Prob}_{\text{LinRec}}(i) + (1 - \alpha) \cdot \text{Prob}_{\text{FMLP}}(i). \tag{14}$$

However, because these probabilities were not available, we relied on a ranking-based fusion instead.

c) Performance Imbalance.: Another key obstacle was that FMLP-Rec outperformed LinRec by approximately 10% on both datasets, creating a common binary classification issue where the model consistently favors the stronger predictor. Although we attempted to mitigate this by increasing LinRec's influence, pushing too aggressively in that direction led to a decline in overall performance. Conversely, when allowing α to be learned without constraints, its maximum value hovered around 0.05, effectively defaulting the hybrid model to rely almost exclusively on FMLP-Rec.

VII. Conclusion

This paper presents a hybrid sequential recommendation model that integrates the strengths of LinRec and FMLP-Rec to achieve near-linear runtime while preserving global context and robust denoising. Our approach adaptively fuses linearized attention with frequency-domain filtering via a gating network that selects the branch based on statistical features of each sequence. Experimental results on benchmark datasets show that our method outperforms LinRec on both the MovieLens-1M and Amazon Beauty datasets. Additionally, it slightly surpasses FMLP-Rec on MovieLens-1M and achieves a higher NDCG@10 on Amazon Beauty, though its MRR@10 is marginally lower.

In summary, our contributions include: (1) the development of a unified framework that combines the complementary capabilities of LinRec and FMLP-Rec; (2) a lightweight gating mechanism that dynamically balances the two branches; and (3) extensive empirical validation showcasing substantial improvements in scalability and robustness.

Future research should explore integrating additional near-linear models into our gating-hybrid framework, as adding models that excel in different aspects can improve accuracy and could prevent the hybrid model from relying too heavily on a single model. In addition, rather than relying solely on basic statistical features from user sequences, more context-based features—such as genre for MovieLens-1M dataset—can be incorporated. A more complex research field that we propose is leveraging temporal abstraction techniques [16], [17] to capture high-level

features that better summarize user behavior. Such temporal abstractions can help cluster sequences with similar dynamics, thereby enhancing the gating network's ability to select the optimal branch. Moreover, incorporating frequent pattern mining over these temporal abstractions—as demonstrated in the medical domain [18]—could uncover recurrent behavioral patterns that further correlate users and improve recommendation performance.

REFERENCES

[1] L. Liu, L. Cai, C. Zhang, X. Zhao, J. Gao, W. Wang, Y. Lv, W. Fan, Y. Wang, M. He, *et al.*, "LinRec: Linear attention mechanism for long-term sequential recommender systems," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 289–299.

[2] K. Zhou, H. Yu, W. X. Zhao, and J. R. Wen, "Filter-enhanced MLP is all you need for sequential recommendation," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2388–2399.

[3] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing Personalized Markov Chains for Next-Basket Recommendation," in *Proceedings of the 19th International Conference on World Wide Web (WWW)*, 2010, pp. 811–820.

[4] G.-E. Yap, X.-L. Li, and P. S. Yu, "Effective next-items recommendation via personalized sequential pattern mining," in *Database Systems for Advanced Applications (DASFAA), 17th International Conference*, Busan, South Korea, Apr. 2012, pp. 48–64.

[5] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: challenges, progress and prospects," *arXiv preprint arXiv:2001.04830*, 2019.

[6] M. Xu, F. Liu, and W. Xu, "A Survey on Sequential Recommendation," in *Proceedings of the 6th International Conference on Information Science and Control Engineering (ICISCE)*, 2019, pp. 106–111, 10.1109/ICISCE48695.2019.00031.

[7] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based Recommendation with Recurrent Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[8] B. Hidasi and A. Karatzoglou, "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, 2018, pp. 843–852.

[9] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, 2018, pp. 565–573.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018, pp. 197–206.

[12] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 2019, pp. 1441–1450.

[13] L. Wu, S. Li, C.-J. Hsieh, and J. Sharpnack, "SSE-PT: Sequential recommendation via personalized transformer," in *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys)*, 2020, pp. 328–337.

[14] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[15] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[16] R. Moskovitch and Y. Shahar, "Classification-driven temporal discretization of multivariate time series," *Data Mining and Knowledge Discovery*, vol. 29, pp. 871–913, 2015.

[17] Y. Shahar, "A framework for knowledge-based temporal abstraction," *Artificial Intelligence*, vol. 90, no. 1-2, pp. 79–133, 1997.

[18] R. Moskovitch and Y. Shahar, "Medical temporal-knowledge discovery via temporal abstraction," in *AMIA Annual Symposium Proceedings*, 2009, p. 452.