International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 16, Number 2, pages 182–198, June 2026

Multi-Interest Sequential Recommendation Based on Prompt-Oriented BERT

Yuruo Su, Shuai Jiang, Sayaka Kamei and Yasuhiko Morimoto
Graduate School of Advanced Science and Engineering, Hiroshima University,
Kagamiyama 1-7-1, Higashi-Hiroshima 739-8521, Japan.
{jiangshuai, s10kamei, morimo}@hiroshima-u.ac.jp

Received: February 20, 2026
Revised: May 5, 2026
Accepted: June 8, 2026
Communicated by Hiroyuki Sato

Abstract

Existing sequential recommendation models primarily rely on item ID sequences, which ignore semantic information within user behaviors and diverse, dynamic interests. To address these challenges, this paper proposes a unified framework to model users' base and temporary interests with lower resource consumption. Our core method converts item titles, item genres, and user ratings in the order of interaction history into semi-structured natural-language prompts, thereby leveraging the semantic understanding capabilities of a pre-trained BERT model. Building on this, we design a multi-module fusion architecture: (1) a Mixture-of-Experts (MoE) network captures diverse user interests; (2) dedicated temporary interest and base preference modules model users' dynamic and stable tastes, respectively; (3) an adaptive fusion layer integrates these signals for precise prediction. Experimental results on benchmark datasets show that the proposed model outperforms all baseline models on Recall, and most models on NDCG. This study demonstrates that combining pre-trained language models with prompt engineering and multi-interest fusion modeling is a practical pathway toward building recommender systems.

Keywords: Sequential Recommendation, Multi-Interest, Mixture of Experts, BERT, Prompt

1 Introduction

With the explosive growth of online information, recommender systems have become essential tools to help users quickly find relevant content. Among various approaches, personalized recommender system is a promising line of work that models how user interests change over time based on past interactions.

Modeling the whole sequence of user interactions is a key research direction in sequential recommendation, as it allows the system to capture the user's complete interests. Most current sequential recommendation methods still rely primarily on sequences of item IDs, overlooking the rich semantic information embedded in user behaviors. Early models like BERT4Rec [21] adopt the BERT architecture [2] and masking objective but do not use a pre-trained language model (PLM). BERT4Rec randomly initializes all parameters from scratch and trains the model only on item ID sequences. To overcome the limitations of ID-based methods, PLMs such as BERT and GPT [17] have been introduced into recommender system research, showing clear advantages in modeling semantic context. UniSRec [8], for instance, leverages a pre-trained BERT encoder to create semantic representations for items from their titles, then models the sequence of these representations to capture user interest

dynamics. However, relying on PLMs without adapting their input format limits their potential in recommendation tasks.

Therefore, inspired by the success of prompt engineering in natural language processing, researchers have begun exploring prompt-based recommendation, which reformulates user behavior and item metadata into natural-language templates to better exploit the semantic capacity of PLMs. The P5 model [4] utilizes the retrained T5 [19] checkpoints as a backbone, transforms user actions, item features, and reviews into natural-language prompts, and gains considerable improvements on many tasks. Still, it does not separate different time scales, such as the gap between base preference and recent interest. The base preferences reflect a user's stable tastes—such as consistently favoring science fiction—while temporary (recent) interests refer to temporary inclinations, like preferring action movies shortly after watching one. Accurately distinguishing between these different types of interests is crucial for generating recommendations that align with a user's overall taste while also adapting to recent trends.

In this paper, to address the different interests of these timescales, we propose a unified framework which consists of the following four components: (1) prompt-based sequential recommendation with semantic inputs, (2) explicit modeling of temporary and base preferences, (3) multi-interest selection by Mixture of Experts (MoE), and (4) strict token-budget constraints. Specifically, we use semi-structured prompts to inject item metadata and user ratings into a pre-trained BERT backbone, then couple this semantic encoder with an MoE-based interest extractor and a dual-path preference decomposition module. In this way, the proposed framework links semantic prompting, multi-interest routing, and time-scale-aware preference modeling within a single recommendation pipeline. In addition, we consider the practical constraint introduced by prompt tokenization: converting item histories into natural-language prompts sharply reduces the number of retained interactions, so we further introduce a token-budget-aware sliding-window strategy to improve supervision coverage without directly extending the prompt length. These components are partially inspired by existing research, but prior studies have not addressed our setting. Rather than treating these components as independent additions, our proposed method modifies and organizes them into a unified pipeline tailored to the specific constraints of prompt-based sequential recommendation. Experiments across several benchmark datasets show that the proposed framework improves recommendation accuracy and that its main components contribute complementary benefits.

The structure of the paper is as follows. Section 2 reviews related works. Section 3 details the proposed method. Section 4 reports experimental results. Section 5 concludes the paper.

2 Related Work

2.1 Sequential Recommendation

Early sequential recommendation methods (e.g., [20], [16]) relied on Markov chain models, where the following action depends only on a few recent items. For example, FPMC [20] combines matrix factorization with a Markov chain to capture both item transitions and long-term user taste, and FOSSIL [16] extends higher-order Markov chains with item-similarity weighting to handle skip behaviors.

With deep learning, full-sequence modeling became possible. GRU4Rec [7] first used recurrent neural networks in recommendation, employing Gated Recurrent Units (GRUs) [3] to track user state across an entire session. Tang et al. designed Caser [22], which embeds recent items as a "2-D image" and applies horizontal and vertical convolutions to capture local patterns, enabling it to recognize skip behaviors that affect the next choice.

Introducing self-attention enables recommender systems to model long-range dependencies effectively, thereby improving recommendation accuracy and generalization. SASRec [9] adopts the transformer and lets every position attend to any earlier step, thus learning long dependencies and gaining both accuracy and efficiency. Inspired by BERT's success in natural language processing, BERT4Rec [21] randomly masks item IDs during training and asks bidirectional attention to recover them, introducing a masked-item prediction target that boosts performance, especially for cold-start and sparse data.

Despite these advances, all the above models share a limit at the input layer. In particular, they use only item IDs. They ignore item titles, descriptions, or attributes, which hurts performance on sparse items and new users. Therefore, this paper introduces rich prompts that combine item titles, genres, and user ratings to provide the language model with a broader input context.

2.2 Prompt-based Recommendation

Prompt learning bridges a PLM to a downstream task by crafting templates that turn the task into a fill-in-the-blank or QA form. It has recently attracted increasing attention in the recommendation domain.

One method focuses on improving model predictions by guiding the model's internal attention with soft prompts. Still, humans cannot directly interpret the meaning of these vectors. PPR [24] is one early study that adds prompt tuning to sequential recommendation. It learns a soft prompt for each user—several trainable vectors prepended to the action list—to guide personalized prediction, helpful in cold-start or few-shot cases. PPR also employs a prompt-level contrastive loss that builds different prompt views to learn robust interest signals. However, soft prompts are difficult to control manually and usually require storing additional parameters for each user or each domain.

In contrast, another method aims to enhance the interpretability and user trust in recommendation results by generating natural language explanations. PEPLER [11] combines discrete templates and continuous prompts to guide PLMs (e.g., GPT-2 [18], T5 [19]) to generate user-friendly explanations, thereby improving system transparency and user trust. P5 [4] casts recommendation tasks into a single text-to-text form using natural-language personalized prompts and utilizes retrained T5 [19] checkpoints as the backbone, demonstrating the flexibility of prompts in recommendation. Their prompts use metadata such as item titles and categories. KP4SR [27] also used T5, converting facts from a knowledge graph into textual snippets and inserting them into the sequence.

These works reveal the potential of prompt learning. Yet most PLM-based methods, still use simple templates that rely on item IDs or short descriptions. Few fuse user traits, item attributes, and rating feedback, leaving the context shallow. Thus, this paper structures prompts that pack item metadata (i.e., title, genre) and user ratings in the order of interaction history into a single sentence to form a richer semantic space and improve model understanding.

2.3 Multi-Interest Recommendation

Users usually have several interests; a single vector oversimplifies them, especially when their tastes vary. Hence, multi-interest frameworks now represent a user with several vectors. SLiRec [26] models user preferences by dividing them into long-term interests and short-term interests. It uses two dedicated modules to capture each part separately, then combines them using an adaptive gating mechanism suitable for sequences with noticeable changes in interest. CLSR [28] introduces contrastive learning by building proxy embeddings for short-term and long-term interests, forcing them to remain distinct, and then merging them with attention to enable the model to weigh them dynamically. CL4SRec [25] further integrates self-supervised contrastive learning with next-item prediction, using crop, mask, and reorder augmentations to learn richer multi-interest user representations.

Other works aim to capture diverse interests without focusing on the time scale. MIND [10] uses capsule routing to cluster history and form several "capsules," each standing for a different taste, such as action movies or science fiction. ComiRec [1] offers two mechanisms, dynamic routing and self-attention, and introduces a controller that decides how many interest vectors to keep and pushes them to differ, stopping collapse.

The above models cannot effectively combine base and temporary interests, nor can they integrate semantic analysis, leading to weak robustness. This paper effectively alleviates these problems by combining semantic prompts with BERT encoding and introducing adaptive fusion between temporary and base interests with expert-level intents.

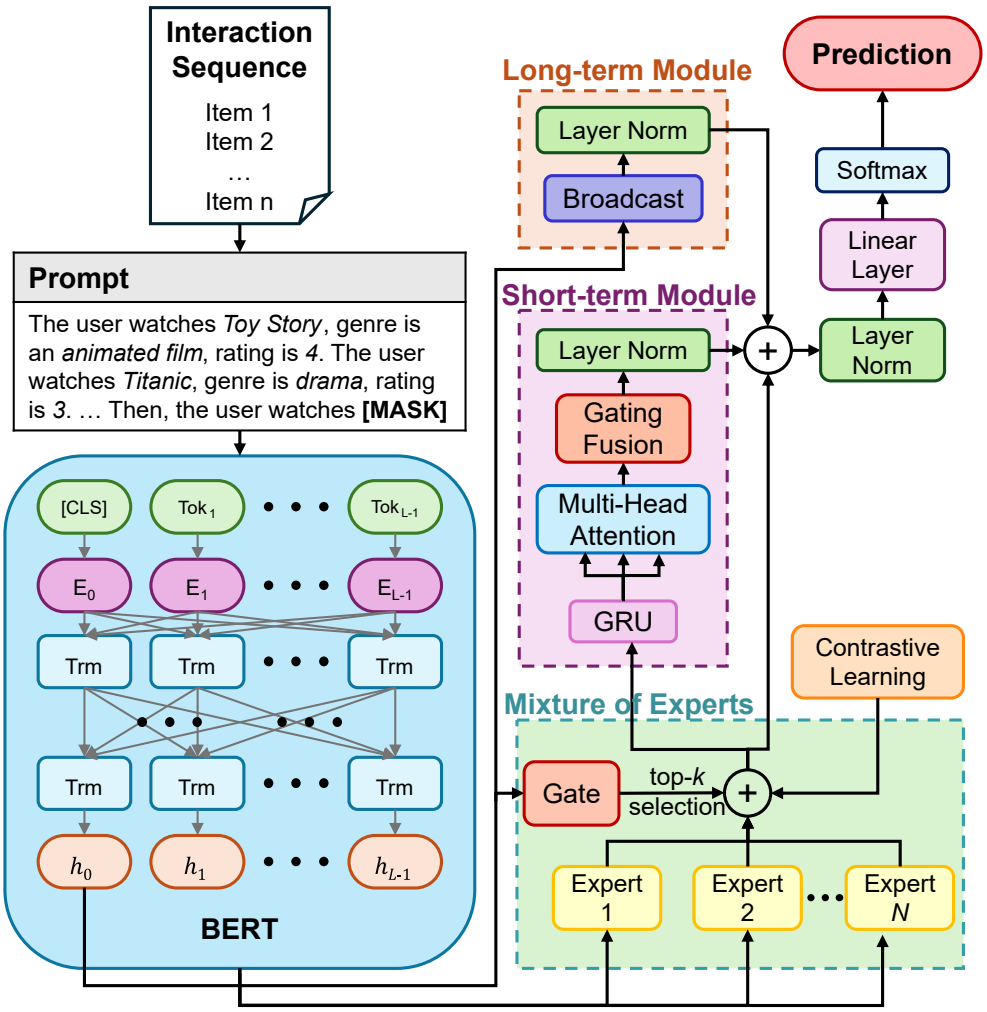

Figure 1: Overall structure of the proposed model.

3 Methodology

This study presents a semantic-enhanced sequential recommendation method built on a pre-trained BERT model and formed by two key parts: (1) a natural-language prompt constructor and (2) a multi-interest extractor with a long-term module, a short-term module, and a Mixture-of-Experts (MoE) network. The model captures users' base preferences through a long-term module and temporary interests through a short-term module. The overall data flow of the proposed framework is illustrated in Fig. 1.

First, the model merges item titles, item genres, and user ratings in the order of interaction history into prompt-based textual inputs. These prompts are tokenized and converted into embeddings by the pre-trained BERT encoder, which captures contextual semantics through its stacked transformer layers, producing encoded representations for downstream interest modeling. We stack a multi-interest module on the encoded output, which passes through a gated MoE network, a temporary interest extractor, and a base interest extractor. Then, we use a fusion layer to predict the following item at the mask position. Unlike methods that rely solely on discrete IDs, this design preserves sequence order, semantic information, and user preference signals. It combines them using hierarchical multi-interest modeling and feature fusion to achieve more comprehensive and accurate predictions.

We use the pre-trained BERT as the backbone. BERT's pre-trained vocabulary and weights let it

converge quickly with limited compute, and task-specific fine-tuning lets us easily add new features. Its masked-language-modeling objective helps the encoder capture deep bidirectional semantics.

3.1 Semi-Structured Prompt Construction

To fully use a PLM's semantic knowledge, we turn item titles, item genres, and user ratings in the order of interaction history into a semi-structured natural-language sequence and feed it directly to BERT. Because BERT is robust to redundant text, a fixed syntactic frame trims noise and highlights key signals such as item titles, genres, and user preferences, while also activating the vocabulary and co-occurrence knowledge learned during pre-training [29]. Unlike the conventional approach of only utilizing discrete IDs, this format enables the model to incorporate semantic information while preserving the sequential order of interactions. Each prompt has two parts: (1) several sentences in the same pattern that describe past actions and (2) a cloze-style question with a `[MASK]` token that asks for the next item. The general template in the code is

> The user is `prefix`: (The user `verb title_token`, genre is `genre_tokens`, rating is `rating_token`.)+ Then, the user `verb [MASK]`.

Here, `prefix` is a dataset-specific sentence such as "The user is watching movies" (MovieLens) or "The user is shopping for items" (Amazon). Because we don't use transfer learning across datasets, using different specific prefixes allows the model to recognize the dataset type, thereby activating the relevant semantic knowledge learned during pre-training. `verb` is a verb depending on the dataset, e.g, "watches" for movies, "shops" for shopping. `title_token` and `genre_tokens` come from the item metadata, and `rating_token` identifies the user's rating. The statement within parentheses is repeated to represent history. For example, if the data used is movie data, a prompt looks like this:

> The user is `watching movies`: The user `watches Toy Story`, genre is `an animated film`, rating is `4`. The user `watches Titanic`, genre is `a drama`, rating is `3`. ... Then, the user `watches [MASK]`.

The history is concatenated in time order, and (old) extra interactions are cut off so that the prompt length stays within the preset `max_tokens`. In this case, only the recent interactions are retained. After truncation, the remaining sequence is tokenized and fed into BERT for encoding.

First, the plain-text prompt is generated from the template, then tokenized into the vocabulary, with special tokens `[CLS]` and `[SEP]`. The actual following item in the user sequence is replaced by `[MASK]` as the prediction target during training. Since the model's masked prediction target is the title, which is generally encoded into varying-length tokens, we map each title to a unique special token to mitigate tokenization fragmentation and eliminate length variability. During training and inference, we restrict the prediction space to this title special token set by masking the logits of all non-title tokens. However, if all title special tokens are initialized to zero vectors, the semantic information of the original titles will be lost. To address this, we refer to Mundra et al. [13], initializing each title special token embedding as the simple mean of each word in the title from the existing embeddings. This approach ensures that the initial vectors of the title special tokens preserve semantic information while maintaining compatibility with the pre-trained model's embedding space.

BERT encodes the entire prompt into embeddings, giving one hidden vector to each token. The hidden vector at `[MASK]` is compared with all candidate items; the item with the highest score is returned as the predicted following recommendation. At the start of the sequence, the `[CLS]` vector gathers the prompt's global preference, while the other token vectors keep local behaviour cues and later support contrastive learning. This prompt design injects item metadata and user ratings into the interaction history in a single step, boosting interpretability and recall without additional feature engineering.

3.2 Mixture of Experts

User behavior typically includes one or more interest preferences, including transient interests across different domains. Inspired by the sparse-gating ideas in FAME [12], we add an MoE module for

multiple interest preferences extraction. Unlike FAME, we use the [CLS] token to control the gate, decoupling the MoE structure from the attention heads in the BERT model to extract preferences at the sequence level and specialize across various aspects of user interest. In addition, we use linear layers as each expert model to produce the same output shape for subsequent modules to learn, reducing learning difficulty.

In the following, we formally describe the model's expert routing mechanism. Let the encoder output be $\boldsymbol{H} = [\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{L-1}] \in \mathbb{R}^{L\times d}$, where L is the token sequence length and d is the hidden size. Here, $\boldsymbol{h}_0$ corresponds to the [CLS] token, and the rest represent contextual embeddings for the input tokens. We apply a linear projection to the [CLS] vector $\boldsymbol{h}_0$ to obtain expert weights:

$$\boldsymbol{p} = \mathrm{softmax}(W_g \boldsymbol{h}_0) \in \mathbb{R}^N,$$

where W_g is a trainable weight matrix and N is the number of experts. Then, $\boldsymbol{p}$ is the selection probabilities over N expert networks. For each expert e, $\boldsymbol{p}_e = [\boldsymbol{p}]_e$ denotes the gating score.

We keep the top-k indices $\mathcal{K} \subset \{1, 2, \ldots, N\}$ with the highest gate scores and set the rest to zero to reduce cost, where $k \leq N$. Each selected expert $f_e(\boldsymbol{H}) \in \mathbb{R}^{L\times d}$ is a two-layer feed-forward network that is applied independently to each row of the encoder output $\boldsymbol{H}$, producing a transformed sequence. The output of the MoE layer is:

$$\boldsymbol{Z}_{\mathrm{MoE}} = \sum_{e\in\mathcal{K}} \boldsymbol{p}_e \, f_e(\boldsymbol{H}) \;\in\; \mathbb{R}^{L\times d}.$$

To make experts learn complementary preferences, we apply window-level contrastive learning [15, 25] to the MoE output $\boldsymbol{Z}_{\mathrm{MoE}} = [\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{L-1}] \in \mathbb{R}^{L\times d}$, where each $\boldsymbol{z}_t \in \mathbb{R}^d$ is the hidden representation at position t.

We divide the sequence into sliding windows $s_u^{(j)}$ of length w for each user u, and compute the average vector in each window as:

$$\bar{\boldsymbol{z}}_u^{(j)} = \frac{1}{w} \sum_{t\in s_u^{(j)}} \boldsymbol{z}_t.$$

Then, $\bar{\boldsymbol{z}}_u^{(j)} \in \mathbb{R}^d$ is the mean representation of the j-th window of user u. We define a positive pair $(\bar{\boldsymbol{z}}_u^{(j)}, \bar{\boldsymbol{z}}_u^{(j^+)})$ as two windows from the same user, and a negative pair $(\bar{\boldsymbol{z}}_u^{(j)}, \bar{\boldsymbol{z}}_{u'}^{(j^-)})$ as windows from different users $u' \neq u$. Then, the InfoNCE loss is formulated as:

$$\mathcal{L}_{\mathrm{CL}} = -\sum_{u,j} \log \frac{\exp\big(\bar{\boldsymbol{z}}_u^{(j)} \cdot \bar{\boldsymbol{z}}_u^{(j^+)}/\tau\big)}{\exp\big(\bar{\boldsymbol{z}}_u^{(j)} \cdot \bar{\boldsymbol{z}}_u^{(j^+)}/\tau\big) + \sum_{u'\neq u} \exp\big(\bar{\boldsymbol{z}}_u^{(j)} \cdot \bar{\boldsymbol{z}}_{u'}^{(j^-)}/\tau\big)},$$

which pulls windows from the same user closer in the expert space. Here τ is the temperature, and '$\cdot$' is the inner product.

Each expert keeps a hidden size d in its two-layer multi-layer perceptron, so no extra tensor size is added. The expert gate weights $\boldsymbol{p}$ are jointly trained with the rest of the network, enabling experts and the router to converge. This objective encourages the model to cluster windows from the same user while pushing apart those from different users in the expert space.

3.3 Short and Long Term Preference Modeling

A single path cannot learn both the stable tastes that users build over the years and the instant interests driven by their latest clicks. If the model looks only at recent data, noise can hide distant clues; if it smooths globally, delicate patterns disappear [26, 28]. To fix this, we design two separate processing paths: the short-term path takes the MoE-enhanced representations as input. In contrast, the long-term path extracts the [CLS] token representation from the original BERT hidden states and broadcasts it across the sequence length. In the end, the MoE branch, the short-term path, and the long-term path each produce one sequence-level tensor, resulting in three tensors in total. These three tensors share the same sequence length and hidden dimension, and are then fed into a fusion layer that assigns data-driven weights, enabling the model to remember the overall taste and respond to the latest actions. Below, we describe each path.

3.3.1 Short-term Module

Inspired by GRU4Rec [7], because the GRU's update gate effectively performs an exponential moving average of past states, contributions from distant steps decay, making the hidden state intrinsically more sensitive to recent interactions. We extend this idea by incorporating a Multi-Head Attention (MHA) layer and a gating mechanism to enhance local context modeling.

The MoE output is then fed into a unidirectional GRU network to capture sequential dependencies in time order. The GRU updates

$$\boldsymbol{g}_t = \mathrm{GRU}(\boldsymbol{z}_t, \boldsymbol{g}_{t-1}), \qquad t = 0, \ldots, L-1.$$

where the states form $\boldsymbol{G} = [\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{L-1}]$.

An MHA mechanism computes attention weights $\alpha_{t,j}$ of position j with respect to current position t over the local neighborhood and produces a context vector:

$$\boldsymbol{s}_t = \sum_{j=1}^{L} \alpha_{t,j}(\boldsymbol{g}_j W_V),\ \alpha_{t,j} = \frac{\exp\big((\boldsymbol{g}_t W_Q)(\boldsymbol{g}_j W_K)^\top\big)}{\sum_{j'=1}^{L} \exp\big((\boldsymbol{g}_t W_Q)(\boldsymbol{g}_{j'} W_K)^\top\big)}$$

where W_Q, W_K, W_V are learnable matrices. Subsequently, a learnable gate uses the original GRU state $\boldsymbol{g}_t$ and the attention output $\boldsymbol{s}_t$ to filter noise and produce the final short-term representation. Filters noise refers to the gating mechanism's ability to dynamically weigh incoming features, attenuating uninformative or noisy signals and enhancing those most relevant to the prediction task.

A learnable gate computed from $[\boldsymbol{g}_t; \boldsymbol{s}_t]$ merges the original GRU output and the context-enhanced vector. An MHA then reads local context and uses a gate γ_t to filter noise:

$$\gamma_t = \sigma\big(W_{\mathrm{gate}} \cdot [\boldsymbol{g}_t; \boldsymbol{s}_t]\big), \qquad \hat{\boldsymbol{g}}_t = \gamma_t \odot \boldsymbol{g}_t + (1 - \gamma_t) \odot \boldsymbol{s}_t,$$

where σ is the sigmoid and $W_{\mathrm{gate}} \in \mathbb{R}^{2d \times d}$ is a learnable gate projection matrix applied to the concatenation of $\boldsymbol{g}_t$ and $\boldsymbol{s}_t$. γ_t is a learnable gate that controls the fusion of the GRU output and the attention vector. $\hat{\boldsymbol{g}}_t$ is the temporary interest representation obtained by adaptively fusing the GRU and attention outputs via γ_t.

The short-term output is

$$\boldsymbol{Z}_{\mathrm{short}} = [\hat{\boldsymbol{g}}_0, \ldots, \hat{\boldsymbol{g}}_{L-1}].$$

This design focuses on delicate patterns in the last L steps and lets the gate tune noise. The GRU state starts as zeros, and its hidden size equals BERT's, making later concatenation easy.

3.3.2 Long-Term Module

The long-term branch constructs a single global vector in one forward pass, thus incurring far lower computational cost than recurrent or self-attention layers. Following BERT's aggregation strategy, the special `[CLS]` token is placed at the first position and its embedding aggregates bidirectional context for the entire prompt. Denote the final-layer `[CLS]` embedding by $\boldsymbol{h}_0 \in \mathbb{R}^d$. We broadcast this vector along the temporal dimension:

$$\boldsymbol{Z}_{\mathrm{long}} = \mathbf{1}_L \otimes \boldsymbol{h}_0 \in \mathbb{R}^{L \times d},$$

where $\mathbf{1}_L$ is an all-ones vector of length L.

3.4 Multi-Module Fusion

The three branches capture complementary signals at different semantic and temporal scales. Specifically, $\boldsymbol{Z}_{\mathrm{MoE}}$ emphasizes diverse interest facets selected by the expert router, $\boldsymbol{Z}_{\mathrm{short}}$ focuses on recent interactions, and $\boldsymbol{Z}_{\mathrm{long}}$ provides a stable global preference summary from the whole prompt. Since these signals may contribute differently across sequence contexts and prediction positions, we do not use fixed coefficients. Instead, we adopt an input-dependent gating mechanism to adaptively determine their relative importance.

Because the outputs of the three branches share the same shape, they can be fused in a unified manner. For each position t $(0 \le t \le L-1)$ of the output $\boldsymbol{Z}_{(m)}$ from each branch $m \in \{MoE, short, long\}$, let $\boldsymbol{z}_t^{(m)} \in \mathbb{R}^d$ denote the d-dimensional hidden representation of branch m. We first concatenate the branch representations at the same position and feed them into a lightweight gating network:

$$\boldsymbol{u}_t = [\boldsymbol{z}_t^{MoE}; \boldsymbol{z}_t^{short}; \boldsymbol{z}_t^{long}],$$
$$\boldsymbol{\ell}_t = \mathrm{MLP}(\boldsymbol{u}_t) \in \mathbb{R}^3,$$

where $\boldsymbol{u}_t$ is the concatenated gate input, $\boldsymbol{\ell}_t \in \mathbb{R}^3$ is the branch-wise score vector produced from this joint input with one score for each branch, and MLP() is a small multi-layer perceptron that outputs one score for each branch. In the implementation, optional branch-reliability scores can also be appended to the gate input. The scores are then normalized by a softmax function:

$$\alpha_t^{(m)} = \frac{\exp(\ell_t^{(m)}/\tau)}{\sum_{n\in\{MoE,short,long\}} \exp(\ell_t^{(n)}/\tau)}$$

where m denotes the current branch, τ is a temperature parameter controlling the sharpness of the routing distribution. Thus, $\alpha_t^{(m)}$ is a module-wise, input-dependent, and position-dependent fusion weight. In this way, the fusion weights are computed jointly from the concatenated branch features, rather than being assigned manually or determined by fixed global parameters.

The final fused representation at position t is computed as

$$\widetilde{\boldsymbol{z}}_t = \sum_m \alpha_t^{(m)} \boldsymbol{z}_t^{(m)}$$

To preserve the information of the main path and stabilize training, we further add a residual connection from the main branch and then apply layer normalization:

$$\boldsymbol{z}_t = \mathrm{LayerNorm}(\widetilde{\boldsymbol{z}}_t + \lambda \boldsymbol{z}_t^{MoE})$$

where λ is a residual scaling factor. This design allows the model to retain the strong semantic backbone representation while still benefiting from adaptive multi-branch fusion.

Overall, the fusion layer serves as a content-adaptive selector over the three complementary signals: the MoE-enhanced semantic sequence representation, the recent-interest representation, and the stable global-preference representation.

4 Evaluation

4.1 Datasets

4.1.1 Overall

We conducted comprehensive experiments on standard public benchmark datasets from different fields. The MovieLens (ML) dataset [5] consists of over one million explicit ratings provided by users for movies, widely used for personalized recommendations research. The Amazon Beauty and Video Games datasets [6] are Amazon review corpus subsets that contain user-item interactions and product metadata, such as titles and categories. We follow the previous work and extract user sequences based on timestamp order. We sorted each user's interactions in chronological order to construct a sequential interaction, and then excluded all users with fewer than five interactions from the dataset. Detailed statistical data are shown in Table 1. Density is the number of interactions divided by the product of the number of users and the number of items, and it reflects the sparsity of the user-item matrix.

Besides, to further improve data quality and strengthen the model's ability to recommend only items users prefer, we selected only items with ratings greater than 4 as user-preferred items for datasets with a large number of users (excluding ML-1M), and excluded items with only low ratings

Table 1: Dataset statistics after preprocessing.

Dataset	**Users**	**Items**	**Reviews**	**Density** (%)
ML-1M	6,040	3,952	1m	4.16
ML-10M	71,567	10,681	10m	1.30
ML-20M	138,493	27,278	20m	0.52
Beauty	52,024	57,289	0.35m	0.01
Video Games	89,021	22,933	0.53m	0.03

for each model before learning. Therefore, the number of samples after this in Tables 2 and 3 will be less than the number shown in Table 1. To maintain consistency, we performed this processing on the inputs of all models.

4.1.2 Sliding Window

Due to BERT's original maximum token limit (typically 512), converting from ID sequences to natural-language prompts significantly reduces the number of items per input. As shown in Tables 2 and 3, since we treat each user's input sequence as a single input, "Original" represents the average and maximum number of items per input (i.e., per user) in the original dataset before truncating for model input. After fixing the maximum token length to 512, compared to the model with ID sequence input ("ID Sequence" in Table 2) and the model with prompt input ("Prompt-512" in Table 3), the number of items per input in the prompt model is significantly reduced, especially in the MovieLens dataset where users have a large amount of interaction; the item length drops from an average of 70-150 to around 30.

As an attempt, without guaranteeing a consistent maximum number of tokens, we used the `answerdotai/ModernBERT-base` model [23], an extended BERT model that can overcome the 512-token limit, as another backbone of the proposed method. We used this model to increase the maximum number of tokens to 2048, as shown as "Prompt-2048" in Table 3. After the increase, the average number of input items is close to that of the ID sequence model in Table 2, but because the input token length increases by a factor of four, the memory consumption increases exponentially by 16 times, significantly increasing the learning cost. We will describe the details in Section 4.4.3.

Therefore, while maintaining the maximum token length, to increase supervision coverage on long interaction histories, we adopt a token-budget-aware sliding-window sampling strategy during training. Instead of always predicting only the last item, we generate multiple training instances per user by moving the target position from the most recent interaction toward earlier interactions with a fixed stride, and we cap the number of windows per user to keep the training cost from becoming too high. For each target, the history is defined as all interactions prior to it, up to the maximum token length. If the resulting prompt exceeds the maximum token budget, we remove the oldest interactions first until the prompt fits the limit. This truncation is sentence-level, so each interaction statement remains complete, and the title/genre/rating fields are never cut in the middle. Finally, we append the cloze sentence with a terminal `[MASK]`, and use the held-out target item as the training label.

More specifically, for a single user, let the chronological training interaction sequence be

$$S = (v_1, v_2, \ldots, v_N),$$

where v_i denotes the i-th interacted item and N is the number of items in the sequence. We move the ending target item position i_j of the j-th window from right to left with the stride s:

$$i_j = N - (j-1)s, \quad j = 1, \ldots, J.$$

Since each target requires at least one preceding interaction, the maximum number of windows is limited by

$$J = \min\left(M, \left\lfloor \frac{N-2}{s} \right\rfloor + 1\right),$$

Table 2: Item statistics of each training set for ID-sequence method.

Dataset	Original		ID Sequence		samples (users)
	mean	max	mean	max	
ML-1M	165.6	2,314	149.2	512	6,040
ML-10M	72.3	2,793	68.4	512	69,167
ML-20M	73.1	3,177	68.9	512	136,677
Beauty	8.7	320	6.7	318	35,931
Video Games	9.3	729	7.3	512	20,548

Table 3: Item statistics of each training set for prompt methods.

Dataset	Original			Prompt-512			Prompt-2048		
	mean	max	samples	mean	max	samples	mean	max	samples
ML-1M	165.6	2,314	6,040	28.4	37	48,320	80.9	137	6,040
ML-10M	72.3	2,793	69,167	26.2	46	559,024	49.0	134	69,167
ML-20M	73.1	3,177	136,677	25.8	49	1,107,944	48.9	133	136,677
Beauty	8.7	320	35,931	6.2	37	95,468	5.7	125	35,931
Video Games	9.3	729	20,548	6.8	38	56,941	6.2	142	20,548

where M is the upper bound on the number of windows per user.

For the j-th window, the item v_{i_j} at the ending position i_j is treated as the training label, and the model is trained to predict it at the terminal `[MASK]` position, while the full item-history before token-budget truncation in this window consists of all preceding items:

$$H_j^{\text{full}} = (v_1, \ldots, v_{i_j-1}).$$

After converting H_j^{full} into a natural-language prompt, we enforce the maximum token length by removing the oldest interactions at the sentence level. Thus, the actual retained history is

$$\widetilde{H}_j = (v_{a_j}, \ldots, v_{i_j-1}),$$

where a_j is chosen as the left boundary of the retained history after truncation, so that the tokenized prompt constructed from $\widetilde{H}_j$ and the terminal `[MASK]` position does not exceed the maximum token budget. The model then uses $\widetilde{H}_j$ as the prompt history and v_{i_j} as the training label.

Overall, the sliding window is defined at the item level, whereas the maximum-length constraint is enforced at the prompt-token level after prompt construction. This design increases the number of training samples and the coverage of supervision for long histories without cutting any interaction statement in the middle.

We set the number of windows per user to $M = 8$ and the sliding stride to $s = 2$. After this, the final number of input samples for the prompt input model, shown as "Prompt-512" in Table 3, increased significantly, reducing the cost of each training epoch and allowing the model to converge faster.

4.2 Baseline Models

To verify the effectiveness of our method, we compare it with the following representative baselines.

- **BERT4Rec** [21] adopts the BERT architecture with a masked item prediction objective against item ID sequences.
- **SASRec** [9] introduces a self-attention mechanism to model dependencies between arbitrary positions in an item ID sequence.

- **GRU4Rec** [7] is a recurrent model that utilizes GRUs to encode user behavior sequences. It captures short-range sequential patterns from recent interactions.
- **CL4SRec** [25] employs contrastive learning to improve representation robustness by generating augmented sequences.
- **P5** [4] is a generative recommendation framework based on T5 that reformulates recommendation tasks as text-to-text generation.

For models with ID sequences input (BERT4Rec, SASRec, GRU4Rec, and CL4SRec), we used the maximum BERT token length (512) for data splitting, shown as "ID Sequence" in Table 2. For the prompt input model P5, we used the same sliding window splitting method as the proposed model, as shown in the "Prompt-512" of Table 3.

4.3 Metrics and Implementation Details

For baseline models including SASRec, GRU4Rec, CL4SRec, and P5, we use the official implementations released by their respective authors. For BERT4Rec, we adopt the version reproduced and systematically benchmarked in the work of Petrov et al. [14], and the framework replicated using the framework under PyTorch and Huggingface Transformers[1]. This setup ensures fair and consistent evaluation across all models. For common hyperparameters, we search the hidden dimension size over {256, 512, 768} and the weight decay over {0.01, 0.001, 0.0001}. We finally chose 768 as the hidden size and 0.01 as the weight decay in the proposed method. All other hyperparameters and initialization strategies are recommended by the authors of these methods.

Our proposed model is implemented in PyTorch and leverages the Huggingface library to obtain the pre-trained BERT backbone. All parameters are initialized using the default strategies of the pre-trained models. We train the model using the AdamW optimizer with a fixed learning rate of 1e-4. To ensure fair comparison, we set the number of transformer layers to 12, the number of attention heads to 12, and the hidden size d to 768, following the standard BERT-base configuration. The `max_tokens` is fixed at 512 for all datasets. For the MoE component, we use four expert networks and select the top-2 experts for each input during the forward pass. The short-term module uses a GRU encoder followed by a 4-head self-attention layer.

All experiments are conducted on four NVIDIA RTX 3090 GPUs with a global batch size of 16, resulting in a batch size of 4 per GPU. Following the evaluation in [21], we adopt a leave-one-out strategy for the next-item prediction task. For each user, the last item in the interaction sequence is held out as the test instance, the second-last item is used for validation, and the remaining interactions are used for training. In line with BERT4Rec [21], for each ground-truth test item, we pair it with 100 negative items that the user has not interacted with, sampled according to their popularity. The model ranks 100 negative items alongside each user's positive items for recommendation.

We evaluate model performance using Recall@K and NDCG@K, where $K \in \{5, 10\}$, following standard practice [4,21]. Recall@K measures whether the ground-truth item is present in the top-K ranked list, while NDCG@K further considers the rank position of the correct item. Higher values indicate better performance for both metrics.

4.4 Results

4.4.1 Overall Performance

Tables 4–5 summarize the performance of all methods on five benchmark datasets. Our proposed model outperforms most baselines across datasets and metrics, demonstrating the effectiveness of our semantic-enhanced framework.

GRU4Rec performs the worst, likely due to its unidirectional structure and limited capacity for long-range modeling. BERT4Rec and SASRec exhibit moderate performance. CL4SRec improves upon BERT4Rec by introducing contrastive learning to enhance representation robustness, but it

[1] `https://huggingface.co/docs/transformers/model_doc/bert`

Table 4: Overall performance for MovieLens.

Model	ML-1M		ML-10M		ML-20M	
	Recall@5	NDCG@5	Recall@5	NDCG@5	Recall@5	NDCG@5
GRU4Rec	0.3585	0.1502	0.4245	0.2761	0.6498	0.4378
SASRec	0.3619	0.2605	0.4286	0.2893	0.6560	0.4951
BERT4Rec	0.3624	0.2691	0.4266	0.2897	0.6558	0.4947
CL4SRec	0.3751	0.2954	0.4641	0.3471	0.7024	0.5379
P5	0.4041	0.3250	0.6396	0.4834	0.7959	0.6521
Proposed	**0.4142**	**0.3357**	**0.6446**	**0.4932**	**0.8159**	**0.6610**

Table 5: Overall performance for Amazon datasets.

Model	Beauty		Video Games	
	Recall@5	NDCG@5	Recall@5	NDCG@5
GRU4Rec	0.3470	0.2781	0.3829	0.2081
SASRec	0.3505	0.2713	0.3860	0.3540
BERT4Rec	0.3560	0.2721	0.3867	0.3551
CL4SRec	0.3760	0.2941	0.5471	0.4362
P5	0.4001	**0.3159**	0.5691	0.4544
Proposed	**0.4010**	0.3158	**0.5805**	**0.4637**

also lacks semantic integration from user- and item-level metadata. P5 achieves competitive results among the baselines due to its prompt-based generative formulation. It benefits from reformulating recommendations as a text-to-text generation task and from incorporating item metadata into natural-language prompts. These results confirm that our model effectively combines the representational ability of PLMs with structured prompt design to accurately recommend. Performance improvements are especially pronounced on denser datasets such as MovieLens, where richer interaction histories and more informative item metadata enable better semantic understanding. Specifically, on the ML-20M dataset, the proposed framework achieves a 2.0% improvement in Recall@5 over the P5 model, highlighting its effectiveness at capturing nuanced user preferences through prompt-based modeling.

Next, we conduct a horizontal comparison of different datasets. Performance on the Amazon Beauty dataset is lower than on ML because this dataset is far sparser and contains more cold-start users (See Table 1). Beauty also offers little textual detail, making prompt design harder, while the diverse, noisy interactions in Video Games make precise ranking (NDCG) harder. Even so, our model still gains from the limited semantics.

4.4.2 Resource Consumption

In addition to recommendation performance, we further compare the resource consumption of the proposed model with P5, as both methods are prompt-based recommendation models. The other baselines are mainly ID-sequence-based sequential recommendation models, whose input representations, backbone architectures, and training objectives differ from those of prompt-based PLM models. Therefore, their resource consumption is not directly comparable in this setting. For a fair comparison between prompt-based methods, Table 6 reports the peak GPU memory usage per GPU in MiB, the training time per epoch in h:mm, and the number of trainable parameters for P5 and the proposed model under the same "Prompt-512" setting as defined in Table 3.

As shown in Table 6, the proposed model requires substantially less training time than P5 across all datasets. This efficiency mainly stems from differences in backbone architecture. P5 is built on T5, which follows an encoder-decoder architecture and therefore introduces additional decoder-side computation during training. In contrast, our method uses an encoder-only BERT backbone

Table 6: Prompt-based model resource consumption.

Dataset	P5			Proposed		
	VRAM	Time	Params	VRAM	Time	Params
ML-1M	9,887	2:22		4,821	0:13	156M
ML-10M	9,893	18:50		5,045	2:55	161M
ML-20M	10,003	21:10	223M	5,375	5:25	169M
Beauty	11,225	1:29		7,743	0:37	226M
Video Games	11,439	0:49		5,565	0:19	174M

Table 7: Ablation results on ML-20M.

Model	Recall@5	Recall@10	NDCG@5	NDCG@10
PB (w/o MoE, ST, LT)	0.7612	0.8351	0.5397	0.5891
+ST+LT (w/o MoE)	0.7811	0.8395	0.5448	0.5936
+MoE+LT (w/o ST)	0.8146	0.9021	0.6597	0.6889
+MoE+ST (w/o LT)	0.8158	0.9036	0.6604	0.6891
Proposed	**0.8159**	**0.9049**	**0.6610**	**0.6901**

and predicts masked tokens at the terminal `[MASK]` position, resulting in a simpler, faster training procedure. This difference is also reflected in GPU memory usage: the proposed model consistently consumes less memory than P5 on all datasets.

Regarding model size, P5 has a fixed number of parameters across datasets because it does not introduce dataset-specific special title tokens into the model vocabulary. In contrast, the proposed model maps each item title to a unique special token to avoid tokenization fragmentation and to constrain the prediction space. As a result, the size of the embedding layer grows with the number of items in each dataset, leading to dataset-dependent parameter counts. This explains why the proposed model has more parameters on the Beauty dataset than on the other datasets: as shown in Table 1, Beauty contains the largest number of items among the evaluated datasets. Nevertheless, the proposed model uses fewer parameters than P5 on four of the five datasets, and even on Beauty, where its parameter count is slightly larger due to the enlarged item-title vocabulary, it still achieves lower GPU memory consumption and shorter per-epoch training time. These results support the practical efficiency of the proposed prompt-oriented BERT framework.

4.4.3 Ablation Study

Module Ablation We conduct an ablation study on the ML-20M dataset to understand how each component contributes to the final performance. It offers substantially longer interaction sequences and a larger, moderately sparse user-item matrix than smaller datasets. Therefore, each component's contribution will be clearer, as turning off any module will lead to a clearer performance drop.

To better demonstrate the effects of pre-training and prompt-based input construction, we introduce a variant that loads a pre-trained BERT-base model from Huggingface and uses natural-language prompts as input. We call this variant Prompt-BERT (PB); it does not include the MoE, short-term (ST), or long-term (LT) preference modeling modules.

From Table 7, except for the worst result, PB, we observe that removing the MoE module results in the most significant performance drop. This result suggests that the expert network is critical in dynamically selecting suitable sub-models based on user preference states, enabling the model to capture diverse interests more effectively. Removing the ST module results in a greater performance decline than eliminating the LT component, likely because temporary preferences reflect users' recent behaviors, which are more sensitive and better indicate next-item prediction. The module's features can explain the difference in these results. The BERT backbone (PB) effectively models global dependencies across the prompt but overlooks local sequential patterns. This limitation is

Table 8: Ablation results on ML-1M and Beauty.

Model	ML-1M		Beauty	
	Recall@5	NDCG@5	Recall@5	NDCG@5
BERT4Rec	0.3624	0.2691	0.3560	0.2721
Proposed-2048	0.2950	0.2134	0.3568	0.2754
Proposed-512	**0.4142**	**0.3357**	**0.4010**	**0.3158**

compensated for by the ST module, which explicitly models recent behaviors through recurrent and attention-based mechanisms. In contrast, the LT module primarily contributes stable preference signals derived from the overall context. Its removal has a relatively minor impact on prediction performance. The results show that all three modules play distinct but essential roles in modeling user preferences, and the whole model achieves the best performance when combined.

Dataset Ablation As we described in Section 4.1.2, to maintain a similar average number of items, we compared the data processing method of the `answerdotai/ModernBERT-base` model [23], whose maximum token length is 2048, with the window method of the proposed model, whose maximum token length is 512. Due to the exponential increase in learning costs, we conducted full experiments only on small-scale datasets: the ML-1M dataset, with relatively dense items and a high average token length, and the Beauty dataset, with relatively sparse items and a low average token length. The results are shown in Table 8.

We can see in Table 8 that, after increasing the maximum token length ("Prompt-2048"), the results reach the same level as BERT4Rec, but are not as good as those after windowing ("Prompt-512"). This is because, despite increasing the maximum token length to 2048, as shown in Tables 2 and 3, the average amount of items still falls short of the level of ID Sequence, thus making it impossible to effectively learn base preference. Therefore, the window method for data processing can effectively improve the model's learning efficiency on datasets with limited GPU memory while fully utilizing the dataset.

5 Conclusion

In this paper, we introduce a semantic-enhanced sequential recommendation model that combines pretrained BERT with prompt-based input construction. To effectively capture users' diverse interests, we design a unified architecture that integrates structured natural language prompts, a Mixture-of-Experts module, temporary interest modeling, and base preference modeling.

The main contribution of this work is to show that several previously separate ideas can be organized into a coherent, prompt-oriented recommendation framework under realistic token and memory constraints. Our results suggest that neither semantic prompting nor preference decomposition alone is sufficient to capture the full complexity of next-item prediction; their combination brings a more effective recommendation framework. By integrating semantic prompts, sequence-level multi-interest routing, temporary/base preference modeling, and adaptive fusion, the proposed method provides a practically effective way to improve sequential recommendation with PLMs.

As future work, we plan to consider the following:

- We will explore automatic prompt template generation to better adapt to different user and item scenarios, and the use of external knowledge graphs to enrich semantic representations and support reasoning.
- We will also extend the model to incorporate multimodal features, such as images and tags, thereby improving the fluency and adaptability of prompt expressions.
- We will use dynamic masking and constrained decoding to enable the model to understand semantic information, reducing information loss more directly.

- Additionally, we will integrate larger PLMs, such as BERT-large, T5, or GPT-style architectures, as backbones to further enhance semantic understanding, generalization, and robustness.

Acknowledgment

This work was supported by JST SPRING, Grant Number JPMJSP2132, and JSPS KAKENHI (25K15130).

References

[1] Yue Cen, Yunshan Ma, Wenjie Wang, Weinan Zhang, Jian Tang, Xian Chen, Xiaoqiang Zhu, and Rongrong Ji. Controllable Multi-Interest Framework for Recommendation. In *Proc. 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2942–2951, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 4171–4186, 2019.

[3] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (GRU) neural networks. In *Proc. IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[4] Song Geng, Senzhang Liu, Zhaochun Fu, Zhibo Zhou, Pengjie Li, Xiaofei Zhou, and Philip S. Yu. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In *Proc. 16th ACM Conference on Recommender Systems*, pages 299–315, 2022.

[5] F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 2015.

[6] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proc. 25th International Conference on World Wide Web*, pages 507–517, 2016.

[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proc. 6th International Conference on Learning Representations*, 2016.

[8] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proc. 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 585–593, 2022.

[9] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proc. IEEE International Conference on Data Mining (ICDM)*, pages 197–206, 2018.

[10] Chao Li, Guangyou Zhou, Yu Zhang, Yun Ding, Jinyang Gao, and Hui Xiong. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proc. 28th ACM International Conference on Information and Knowledge Management*, pages 2615–2623, 2019.

[11] Lei Li, Yongfeng Zhang, and Li Chen. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 2023.

[12] Mingrui Liu, Sixiao Zhang, and Cheng Long. Facet-Aware Multi-Head Mixture-of-Experts Model for Sequential Recommendation. In *Proc. 18th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 127–135, 2025.

[13] Nandini Mundra, Aditya Nanda Kishore, Raj Dabre, Ratish Puduppully, Anoop Kunchukuttan, and Mitesh M Khapra. An empirical comparison of vocabulary expansion and initialization approaches for language models. *arXiv preprint arXiv:2407.05841*, 2024.

[14] Aleksandr Petrov and Craig Macdonald. A systematic review and replicability study of BERT4Rec for sequential recommendation. In *Proc. 16th ACM Conference on Recommender Systems*, pages 436–447, 2022.

[15] Rui Qiu, Zelong Huang, Hongzhi Yin, Meng Wang, and Quoc Viet Hung Liu. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proc. 15th ACM International Conference on Web Search and Data Mining*, pages 813–823, 2022.

[16] Ruining R He and Julian McAuley. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *Proc. 16th IEEE International Conference on Data Mining*, pages 191–200, 2016.

[17] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[18] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[20] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In *Proc. 19th International Conference on World Wide Web*, pages 811–820, 2010.

[21] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proc. 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1441–1450, 2019.

[22] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proc. 11th ACM International Conference on Web Search and Data Mining*, pages 565–573, 2018.

[23] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proc. 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, 2025.

[24] Yuhui Wu, Ruobing Xie, Yongchao Zhu, Hongxia Zhang, Yu Zhang, Fei Sun, and Yong Yu. Personalized prompt for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36:3376–3389, 2024.

[25] Xing Xie, Fei Sun, Zeyu Liu, Wenwu Ou, and Yongfeng Jiang. Contrastive learning for sequential recommendation. In *Proc. IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1259–1273, 2022.

[26] Zhiwei Yu, Defu Lian, Ahmad Mahmoody, Xing Xie, Enhong Chen, and Yongfeng Zhang. Adaptive user modeling with long and short-term preferences for personalized recommendation. In *Proc. 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4213–4219, 2019.

[27] Jin Zhai, Xiaorui Zheng, Chang-Dong Wang, Li Lin, and Philip S. Yu. Knowledge prompt-tuning for sequential recommendation. In *Proc. 31st ACM International Conference on Multimedia*, pages 6451–6461, 2023.

[28] Yihong Zheng, Chang Gao, Jie Chang, Xiangnan Huang, Xiang Wang, and Tat-Seng Chua. Disentangling long and short-term interests for recommendation. In *Proc. ACM Web Conference*, pages 2256–2267, 2022.

[29] Dawei Zhu, Michael A Hedderich, Fangzhou Zhai, David Ifeoluwa Adelani, and Dietrich Klakow. Is BERT robust to label noise? A study on learning with noisy labels in text classification. *arXiv preprint arXiv:2204.09371*, 2022.