Reformulating Sequential Recommendation: Learning Dynamic User Interests with Content-enriched Language Modeling

Junzhe Jiang
School of Data Science, University of Science and Technology of China
Hefei, Anhui Province, China
jzjiang@mail.ustc.edu.cn

Shang Qu
School of the Gifted Young, University of Science and Technology of China
Hefei, Anhui Province, China
qushang@mail.ustc.edu.cn

Mingyue Cheng
School of Computer Science, University of Science and Technology of China
Hefei, Anhui Province, China
mycheng@ustc.edu.cn

Qi Liu
School of Computer Science, University of Science and Technology of China
Hefei, Anhui Province, China
qiliuql@ustc.edu.cn

Abstract

Recommender systems are essential for online applications, and sequential recommendation has enjoyed significant prevalence due to its expressive ability to capture dynamic user interests. However, previous sequential modeling methods still have limitations in capturing contextual information. The primary reason for this issue is that language models often lack an understanding of domain-specific knowledge and item-related textual content. To address this issue, we adopt a new sequential recommendation paradigm and propose LANCER, which leverages the semantic understanding capabilities of pre-trained language models to generate personalized recommendations. Our approach bridges the gap between language models and recommender systems, resulting in more human-like recommendations. We demonstrate the effectiveness of our approach through experiments on several benchmark datasets, showing promising results and providing valuable insights into the influence of our model on sequential recommendation tasks. Furthermore, our experimental codes are publicly available.

1 Introduction

Recommender systems are crucial engines of various online applications, including e-commerce, advertising, and online videos. They play a pivotal role in discovering user interests and alleviating information overload, thus assisting users in their decision-making process. Over the past years, collaborative filtering (CF) [20] has become one of the most prevalent techniques due to its effectiveness and efficiency in large-scale recommendation scenarios.

As one type of model-based CF technique, the matrix factorization algorithm [11] has been recognized as one of the most successful techniques in the recommendation community. Despite its effectiveness, this approach is not good at capturing dynamic user interests since temporal properties are largely overlooked in user-item interaction matrix. Later, a series of works are devoted to modeling evolving user preference. Among them, sequential recommendation [19, 7, 1] has become increasingly prevalent since the sequence of historical behavior ordered by interacted time can clearly reflect each user's long-term and short-term preference, as shown in Figure 1(a).

Preprint. Under review.

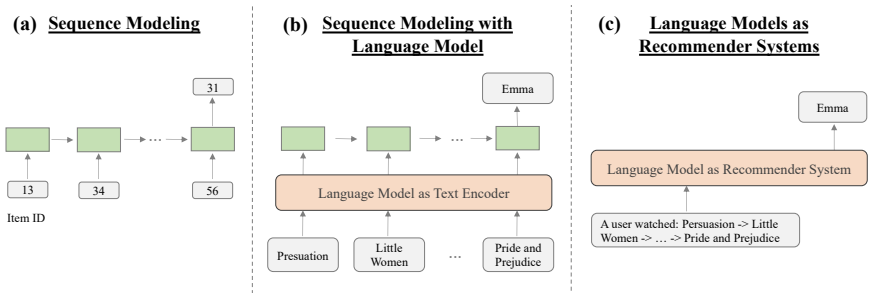


Figure 1: A paradigm shift in sequential recommender systems.

By carefully revisiting recent progress in the sequential recommendation, we notice much of the great progress has been driven by achievements in network architecture or pre-training algorithms in natural language processing [5]. For one thing, a large body of advanced architecture models [9, 17, 27] has facilitated the building of high-capacity recommendation models. For another, some researchers attempt to leverage the powerful capacity of language modeling in semantic understanding to perform content-based sequential recommendations [29, 26]. In addition, we also notice a few related advancements attempt to integrate pre-trained language models (PLMs) into recommender systems, leveraging their ability to capture semantic information and achieving promising results [14]. However, these works do not depart from the previous paradigm, merely adding a text encoder and continuing to use traditional sequential modeling methods, as depicted in Figure 1(b).

With the continuous improvement of model capabilities, we ponder on whether the paradigm of sequential recommendation can be revolutionized to recommend items to users in a way that mimics human reasoning. Some recent works have attempted to use powerful PLMs as recommender systems by formulating recommendation as a language modeling task, with the item IDs [6] or titles [31] as input to the language model. Despite leveraging the inferential capabilities of language models for generating sequential recommendations, these methods have yielded unsatisfactory performance, falling behind the previous state-of-the-art results.

In light of the above, language models perform suboptimally to sequential recommendation systems is their lack of domain-specific knowledge and mastery of item-related textual content. For example, in movie recommendation, *The Matrix* could be understood by the language model as a mathematical matrix, rather than being associated with hacker-related information, which would significantly reduce the effectiveness of the recommendation. Moreover, language models are limited by the scarcity of pre-training data and the constraint of input text length during inference, which makes it challenging to incorporate sufficient item content knowledge to facilitate recommendations.

To address these issues, we propose LANguage modeling for Content-Enriched Recommendation (LANCER) to employ PLMs as sequential recommender by integrating domain-specific knowledge and user behavior. Specifically, we introduce a **knowledge prompt** that facilitates the acquisition of domain-specific knowledge, and a **reasoning prompt** that integrates the acquired knowledge and item content information to generate personalized recommendations. By transforming the sequential recommendation into a text generation task, we change the paradigm of sequential recommendation, and Figure 1(c) illustrates the new paradigm.

The contributions of this paper are summarized as follows:

- We propose LANCER that incorporates domain knowledge and item content prompts into PLMs, leveraging their semantic understanding abilities to generate user recommendation results. This is a new paradigm for sequential recommendation systems.
- Our designed model offers the advantage of integrating external knowledge to enhance the modeling of item content understanding. Our approach bridges the gap between language models and recommender systems by fusing knowledge from both domains.
- We conduct extensive experiments on public datasets and demonstrate that our method achieves promising results, outperforming state-of-the-art methods in real-world recommendation scenarios. Furthermore, we provide valuable insights into the influence of our model on sequential recommendation tasks.

2 Preliminaries

Consider a sequential recommendation task, where the input is the items interacted by a user, formulated by a sequence of items $i_1, i_2, ..., i_{n-1}$, and the output is the prediction of the next item i_n that the user may interest. Traditional methods encode each item as a vector and use sequential modeling methods to predict the next item using classification layer, and calculate the result through the predicted probability distribution $p(i_n \mid i_1, i_2, ..., i_{n-1})$.

When it comes to language models as sequential recommender system, we need to change the above definition. In our work, we represent each item in a natural language way rather than using a discrete representation (such as ID). Specifically, we utilize information consistent with human perception, such as using the movie title. For item i_k, we map it to encoded tokens $e_k = [t_{k_1}, t_{k_2}, ..., t_{k_m}]$. We use a prompt $f(\cdot)$ to convert the historical sequence of interactions between users and items into natural language text $s = f([e_1, e_2, \ldots, e_{n-1}])$ as context input to the model. Then we need the model to generate several tokens and decode these tokens into natural language, representing the next possible items the user may interact with. The output of the model will be in natural language, thus the probability distribution we need to calculate is equation 1 until the model generates the $[EOS]$ (End Of Sequence) token.

$$p(e_n \mid s) = \prod_j p(t_{n_j} \mid t_{n_{<j}}, s). \tag{1}$$

We adopt a transformer-based autoregressive PLM to validate our method. The interaction between users and items is unidirectional sequential data arranged in chronological order, making the autoregressive model suitable for modeling this type of problem. We utilize the autoregressive nature of the model to predict the next token based on preceding tokens. This prediction task is achieved by minimizing the negative log-likelihood of the subsequent token:

$$\mathcal{L} = \sum_n -\log p\left(t_n \mid t_1, \ldots, t_{n-1}; \Theta_{LM}\right), \tag{2}$$

where t_n is the next token to be predicted, and Θ_{LM} represents all parameters in Transformer.

3 Methodology

In this section, we introduce the structure of LANCER, elaborating on its input/output and basic principles. In Section 3.1, we introduce the framework and operation of the model. In Section 3.2, we describe how we use a knowledge prompt to learn domain-specific knowledge in the recommendation domain and obtain vector representations for each item. In Section 3.3, we explain how our designed reasoning prompt integrates a user's interests, historical behavior and domain-specific knowledge to enable language model-based recommendations. Finally, in Section 3.4, we elaborate on how to map the text generated by our model to a specific item for recommendation.

3.1 Model Architecture

The proposed LANCER architecture is shown in Figure 2. For each item in the user interaction history, we extract all text content information and input it into a language model with fixed parameters to train the knowledge prompt and obtain a vector representation h_i for each item. Afterward, we use an attention mechanism to combine domain knowledge with history of user interaction as the reasoning prompt, allowing the model to generate the next item that the user may be interested in. Finally, the generated text is mapped to one of the candidate items recommended to the user.

3.2 Knowledge Prompt

Once directly applying a language model for item recommendation, a major challenge arises as language models may not possess the knowledge of the relevant domains. Previous works [31, 6, 3] attempt to use movie IDs or movie titles as input, relying on language model to make recommendations. However, these approaches fail to fully harness the text understanding capabilities of PLMs.

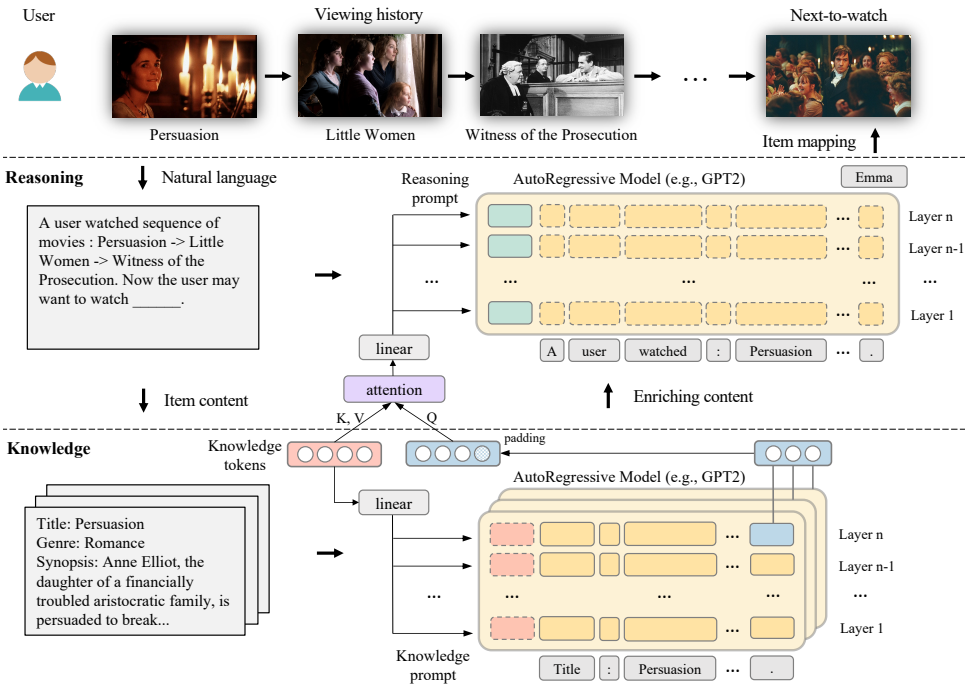


Figure 2: Illustration of the proposed LANCER, where the dashed line indicates the trainable parameters and the solid line indicates the frozen parameters.

Movie IDs, being numerical information, can not convey the underlying meaning to the language model, and even movie titles may not accurately reflect the genre and details of the movie.

To address the above issues, we design a continuous prompt to learn recommendation domain information while avoiding catastrophic forgetting that may occur with language model retraining. Inspired by Prefix-tuning [13] and P-tuning [16, 15], we add continuous prefix prompts to guide the PLM without changing the parameters. We define this as a knowledge prompt. The knowledge prompt adds parameters in front of each layer of the language model, and the effect of these continuous parameters propagates to the whole transformer activation layer and then to subsequent tokens. In terms of implementation, we set knowledge tokens with a fixed length and corresponding embedding spaces, and then generate the hidden states of each layer's attention blocks in the transformer through a fully connected layer. We do not directly optimize the related parameters because updating the prompt parameters directly leads to unstable optimization and a slight drop in performance [13]. Our calculation process is as follows:

$$KP = MLP(Emb([k_1, \ldots, k_\theta])), \tag{3}$$

where k_i denotes i-th knowledge token, θ denotes the number of knowledge tokens. Suppose a PLM with L layers, for an attention computation in the l-th layer, a distinct prefix of key-value pairs K_l and V_l is learnt, which are augmented to become:

$$K_l^{'} = [KP_{l,K}; K_l], \quad V_l^{'} = [KP_{l,V}; V_l], \tag{4}$$

where $K_l^{'}, V_l^{'} \in \mathbb{R}^{(\theta+M)\times d}$, M denotes the input length of the language model, and d denotes the dimension of hidden layers.

3.3 Reasoning Prompt

With the help of knowledge prompts, we can provide more accurate recommendations for a specific domain. However, this method lacks a modeling of user behavior and a deeper understanding of

item information. Using only knowledge prompts, the model obtains the same domain information for different users. Yet, user preferences often lie within a more personalized subset of domain information, rather than the entire domain. For instance, a user who enjoys science fiction movies may benefit from a knowledge prompt that selects more science fiction items based on their viewing history, resulting in more accurate recommendations.

To address this issue, we propose a method called reasoning prompt to learn dynamic user interest. Reasoning prompt combines domain knowledge and user behavior to generate prompts for the language model to provide personalized recommendations. Specifically, reasoning prompt consists of two parts: first, the embeddings of knowledge tokens are mapped to the embedding space and then shared with the knowledge prompt via a fully connected layer to obtain a hidden state; second, for each item, we use the last hidden state h_i of the last input token of computed by language model with the knowledge prompt, which considers all previous tokens, to represent the item i. Equation 5 shows how the hidden layer is calculated:

$$h_i = \Theta_{LM}(\mathcal{D} \mid K_l^{'}, V_l^{'}), \tag{5}$$

where $\mathcal{D}$ denotes the item i detailed text information. Therefore, a user's behavior sequence can be expressed as $[h_1, \ldots, h_{t-1}]$. We will pad if the length of the sequence less than the fixed N.

Next, we use an attention mechanism to combine domain knowledge and user behavior. To obtain user-specific domain information, we extract specialized domain information based on the user's interest history, which can help to provide more effective recommendations. We use the user's historical behavior representation as query, domain information as key and value, and extract the unique prompt prefix for each user to help the subsequent model to better recommend:

$$RP = MLP(softmax\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{d_k}}\right)\mathcal{V}). \tag{6}$$

In equation 6, $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ denotes query, key and value of attention mechanism respectively. They are calculated as follows:

$$\mathcal{Q} = \mathbf{w}_q[h_1, \ldots, h_{t-1}] + \mathbf{b}_q, \tag{7}$$

$$\mathcal{K} = \mathbf{w}_k KP + \mathbf{b}_k, \tag{8}$$

$$\mathcal{V} = \mathbf{w}_v KP + \mathbf{b}_v, \tag{9}$$

where $\mathcal{Q} \in \mathbb{R}^{N \times d}$, $\mathcal{K}, \mathcal{V} \in \mathbb{R}^{\theta \times d}$, and $\mathbf{w}$ and $\mathbf{b}$ denote the weight and bias parameters of the trainable linear layer separately.

3.4 Item Mapping

In the generation phase, we use beam search decoding to generate output sequences with the 10 highest scores, then truncate each output to include only the first new item. Beam search decoding at step t can be expressed as:

$$E_t = \underset{E \in \mathcal{C}_t, |E|=w}{\arg\max} \log p(E \mid \Theta_{LM}(s, RP)), \tag{10}$$

$$\mathcal{C}_t = \{[e_{t-1}, e] \mid e \in \mathcal{V}, e_{t-1} \in E_{t-1}\}, \tag{11}$$

where $\mathcal{V}$ is the vocabulary set, w is the number of beams, and E is the current set of top-w outputs.

At this point, we have obtained the results of the model prediction, but in order to calculate the accuracy of our prediction, we need to map the predicted result to one of all items in database. Each output is matched to an item in the database according to the following procedure. First, the text sequence is passed through the embedding layer of the autoregressive model, resulting in a sequence of token embeddings. We proceed to perform max pooling over the token embeddings to obtain a fixed length vector representation of the output. Finally, we compute the cosine similarity in equation

Table 1: Statistics of the datasets after processing.

Dataset	Num. User	Num. Item	Num. Inteaction	Sparsity	Avg. Actions/User	Avg. Actions/Item
MovieLens	6040	3231	72480	99.63%	12.0	22.43
MIND	91935	44908	2248027	99.94%	24.45	50.06
Goodreads	120968	28480	1095758	99.97%	9.06	38.47

12 between the vector representations of the output i_{pred} and the titles of each item in the database $\mathcal{I}$, then identify the item with the highest similarity. The representations of the total set of items can be computed only once and used throughout the generation process.

$$i^{'} = \arg\max_{k \in \mathcal{I}} CosSim(i_{pred}, i_k). \tag{12}$$

4 Experiments

In this section, we first describe the experimental setup and compare them with other competitive baselines. We then analyze the experimental results of recommendation performance.

4.1 Experimental Setup

Datasets. We conduct experiments on three public datasets:

- MovieLens[1] is a widely used movie recommendation dataset. For each user, we order their rated movies chronologically and use the movie name as a movie's representation. Specifically, we conduct experiments on a 1M-sized dataset.
- MIND[2] [28] is a dataset for English news recommendation. We use the provided historical interactions to construct user behavior sequences and use the title of the news article to represent each item. In this paper we use the small version of MIND dataset.
- Goodreads[3] [25, 24] datasets contain reviews from the Goodreads book review website, and a variety of attributes describing the items. For each user, we order the books they added to their public shelf according to interaction time to obtain historical sequences, and use the title to denote each book. We use a subset on poetry from Goodreads dataset.

For all three datasets, we limit the sequence length to between 5 and 10 items, with sequences of less than 5 items are omitted and sequences of more than 10 items are truncated from the beginning to include only the last 10 items. Detailed statistics of the processed datasets are presented in Table 1.

Evaluation Metrics. In this work, Recall and NDCG are employed as evaluation metrics. The former is an evaluation of un-ranked retrieval sets while the latter reflects the order of ranked lists. Here, we consider top-N (e.g., N = 5, 10) for recommendations. We follow the leave one-out strategy described in [9] to split the interactions into training, validation, and testing sets. For the testing stage, we adopt the all-ranking protocol to evaluate recommendation performance. To evaluate our model, we compute Recall and NDCG for the top 5 and 10 items respectively.

Implementation Details. We use the GPT-2 [18] pre-trained model with 124M parameters from Hugging Face [4] as our backbone model for experiments. During training, we employ the Adam [10] optimizer with a linear schedule learning rate decay. For different datasets, we keep the following hyperparameters consistent: batch size of 16, maximum token length of 512 for the language model input, maximum input length of 32 tokens for each item (truncated for items exceeding the limit). Due to the varying sizes of the datasets, there are differences in the learning rate and epochs. We employ the Sequential Tuning approach from [12], performing Fixed-LM Prompt

[1] https://grouplens.org/datasets/movielens/

[2] https://msnews.github.io/

[3] https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home

[4] https://huggingface.co/gpt2

Table 2: Recommendation performance comparison of different recommendation models over three datasets, in which the best results are in bold and the second best results are underlined.

Dataset	Metric	Model							
		Caser	GRU4Rec	SASRec	BERT4Rec	FDSA	MV-RNN	SASRec-C	LANCER
MovieLens	Recall@5	0.0225	0.0737	<u>0.0967</u>	0.0776	0.0901	0.0853	0.0944	**0.1086**
	Recall@10	0.0330	0.1316	<u>0.1589</u>	0.1391	0.1407	0.1399	0.1454	**0.1616**
	NDCG@5	0.0151	0.0450	<u>0.0624</u>	0.0466	0.0595	0.0527	0.0618	**0.0712**
	NDCG@10	0.0185	0.0635	<u>0.0823</u>	0.0664	0.0757	0.0703	0.0781	**0.0884**
MIND	Recall@5	0.0312	0.0981	0.1087	0.0853	0.1063	0.1024	<u>0.1107</u>	**0.1168**
	Recall@10	0.0575	0.1569	0.1707	0.1359	<u>0.1728</u>	0.1619	**0.1753**	0.1675
	NDCG@5	0.0181	0.0628	0.0714	0.0544	0.0682	0.0656	<u>0.0723</u>	**0.0766**
	NDCG@10	0.0265	0.0816	0.0913	0.0706	0.0896	0.0847	**0.0930**	**0.0930**
Goodreads	Recall@5	0.1058	0.1871	0.2043	0.1892	0.1713	0.1872	<u>0.2045</u>	**0.2121**
	Recall@10	0.1452	0.2472	0.2641	0.2487	0.2351	0.2486	<u>0.2663</u>	**0.2673**
	NDCG@5	0.0770	0.1394	<u>0.1536</u>	0.1391	0.1195	0.1385	0.1533	**0.1600**
	NDCG@10	0.0898	0.1587	0.1729	0.1583	0.1400	0.1583	<u>0.1733</u>	**0.1779**

Tuning and Prompt+LM Fine-tuning consecutively. A more detailed experimental setup can be found in the appendix. Our model is trained on NVIDIA Tesla A100 (40G). Our code is available at `https://anonymous.4open.science/r/lancer-3109/`.

Compared Methods. We compare the performance of our model to various baselines, including ID-based methods and content-based methods. In addition, we evaluate the performance of recent methods which also use language models as the main model architecture. Specifically, we consider the following baselines.

- Caser [22] learns local sequential patterns using horizontal and vertical convolutional layers.
- GRU4Rec [7] uses recurrent neural networks to model interaction history for session-based recommendation.
- SASRec [9] uses a left-to-right transformer architecture to identify relevant items in historical interactions and in turn predict the next item.
- BERT4Rec [21] employs bidirectional transformer layers, and captures bidirectional item relationships by training the model to predict masked items at random positions in the sequence.
- FDSA [30] combines heterogeneous item features using vanilla attention, then separately models item and feature-level sequences using self-attention.
- MV-RNN [2] employs the RNNs as a sequence encoder to capture user interests from hybrids of item IDs and content features. We implement feature embeddings using the BERT pre-trained model[5].
- SASRec-C [9] is a variant of SASRec, in which we directly regard the content feature representation as sequence input. This baseline is an ID-agnostic baseline.

4.2 Results and Analyses

Recommendation Performance Evaluation. In terms of sequential recommendation, our experimental results are shown in Table 2. Based on the results, we can conclude that LANCER achieves better performance than previous models in movie, news and book recommendations. Therefore, it can be seen that LANCER can integrate various textual features of items well, utilizing the powerful text understanding and generation capabilities of PLMs, as well as incorporating knowledge and item features in the recommendation field, bringing more personalized, personified, and precise recommendations for each user.

Additionally, we notice several findings regarding our model. First, LANCER performs more prominently on MovieLens, which we attribute to two factors: 1) the relatively small size of the dataset allows our model to leverage the extensive knowledge contained in the pre-trained model,

[5] `https://huggingface.co/bert-base-uncased`

	Interaction History	Top-5 Predictions	Target Item
Case 1	Bachelor Party → Final Destination → Blame It on Rio → Rocky III → Tank Girl → Jaws 2 → Rocky IV → House II: The Second Story → Superman III → Maximum Overdrive	**Superman IV: The Quest for Peace** Exorcist II: The Heretic Pet Sematary Jaws 3-D Amityville 3-D	**Superman IV: The Quest for Peace**
Case 2	Howard the Duck → Superman IV: The Quest for Peace → Species II → My Favorite Martian → Universal Soldier: The Return → Battlefield Earth → Time Tracers → The Giant Gila Monster → The Killer Shrews → Carnosaur 3: Primal Species	The Island of Dr. Moreau Mission to Mars Judge Dredd **Carnosaur 4: The Return of Michael Myers** Carnosaur	**Carnosaur 2**
Case 3	Battlefield Earth → Mixed Nuts → Robin Hood → The Good Earth → Held Up → Citizen Kane → Good Will Hunting → High Noon → The Godfather: Part II → Trekkies	The Godfather: Part III The Godfather Vertigo Star Wars: Episode I - The Phantom Menace Welcome to the Dollhouse	**Hero**

Figure 3: Case study of three users from the MovieLens-1M dataset in terms of top-5 ranked predictions generated by LANCER. The texts in green indicate predictions that map to the target.

Table 3: Ablation results for various LANCER models and P5-single on three different datasets, in which the best results are in bold. R and N denote Recall and NDCG respectively.

Model	MovieLens				MIND				Goodreads			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
P5-single	0.0503	0.0659	0.0370	0.0421	0.0369	0.0588	0.0244	0.0314	0.1358	0.1778	0.1019	0.1154
LANCER-id	0.0715	0.1030	0.0465	0.0567	0.0931	0.1328	0.0620	0.0748	0.1582	0.1734	0.1304	0.1354
LANCER-bare	0.0263	0.0459	0.0130	0.0193	0.0974	0.1511	0.0551	0.0724	0.1492	0.1822	0.0836	0.0944
LANCER-title	0.0998	0.1493	0.0654	0.0814	0.1104	0.1556	0.0730	0.0878	0.1992	0.2267	0.1527	0.1618
LANCER-full	**0.1086**	**0.1616**	**0.0712**	**0.0884**	**0.1168**	**0.1675**	**0.0766**	**0.0930**	**0.2121**	**0.2673**	**0.1600**	**0.1779**

unleashing its full potential. 2) Movie titles provide limited information, and incorporating more information (such as genre, synopsis, director, etc.) and user movie-watching behavior can lead to more precise reasoning and better recommendations. Second, we notice that the improvement in the @5 metrics is greater than that in the @10 metrics, which we attribute to the generation capabilities of the language model. Generative language models often perform better in the first few generations, but the quality of generations tends to decline as the number of generations required increases [23, 8, 4]. However, we use a generative self-regressive model that can better fit the characteristics of sequential data, which is also why we choose GPT-2 as our backbone.

Case Study. In Figure 3, we select three user examples from the MovieLens-1M dataset and present the top 5 output texts directly generated by the proposed LANCER. It is evident that our model is capable of generating recommendations that align reasonably well with user preferences. Furthermore, we observe that model outputs are realistic movie titles in almost all scenarios, even when the model fails to accurately recommend the next item. Case 2 is among the few examples where the model output do not exactly match a movie title in the database. In this case, the original model output is mapped to *Carnosaur 2*, which is the correct next item. In Case 3, although the target item does not appear in the top 5 predictions, several predicted movies share overlapping genres with the target. Both *Welcome to the Dollhouse* and *Hero* are "Comedy|Drama" movies, and *The Godfather* also lies in the "Drama" category.

Ablation Study. In this section, we mainly discuss the effect of incorporating item content information into sequential recommendation and the impact of knowledge prompts and reasoning prompts. We design four LANCER recommendation methods: LANCER-full represents the same setting as the main experiment, LANCER-title uses only item title text without considering other richer content information, LANCER-id uses only item IDs as text representation, and LANCER-bare removes all templates and prompts and simply concatenates item titles for recommendation. We also implement P5-single, which is trained only on single sequential recommendation tasks due to dataset limitations. We adopt the same 13 different prompts for sequential recommendation as in [6]. We report Recall and NDCG (@5, @10) and the experimental results are shown in Table 3.

According to the experimental results, it can be observed that the LANCER-full incorporating domain knowledge information and item content information achieve the best results on all three datasets. Additionally, we can draw the following conclusions: 1) using IDs for recommendation is weaker than those using textual information, which we believe is due to the language model's better

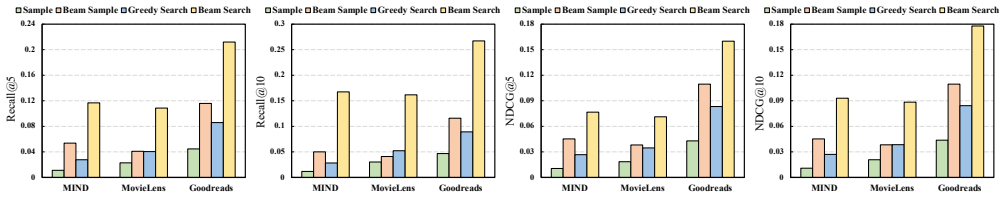


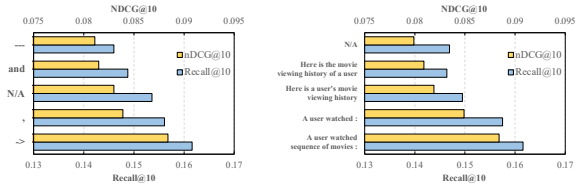
Figure 4: The impact of different decoding strategy on recommendation results.

understanding of well-defined textual information. 2) The recommendation results of P5-single are weaker than LANCER-id, which demonstrates the advantages of autoregressive models in sequential recommendation tasks. 3) LANCER-title performs best on the MIND dataset, followed by Goodreads, and worst on MovieLens. We believe this is due to the different domain characteristics and data types of the three datasets. The MIND dataset, as a news recommendation dataset, already contains sufficient information in the news title to summarize the specific content, and people often determine their interest by browsing news titles. The MovieLens dataset often has short movie titles with limited information, serving only as a sales tool to attract people to watch the movie or seek more information, so people can not determine their interest based solely on a movie title.

Study of Decoding Strategy. In this section, we explore the impact of different decoding strategies and mapping methods on recommendation performance. We adopt four different decoding strategies: Sample, which generates a token sequence by sampling and then calculates the top n items with the highest cosine similarity in the item database; Beam Sample, which simultaneously generates n token sequences, and for each generated result, calculates the item with the highest cosine similarity to obtain n recommendation results; Greedy Search, which is similar to Sample but adopts a greedy search to generate the token sequence; and Beam Search, which is similar to Beam Sample but uses beam search to generate the token sequence. The experimental results are shown in Figure 4. We can observe that the beam search strategy outperforms the sampling strategy, and mapping n generated results to the top item with the highest cosine similarity is better than mapping a single generated result to the top n items with the highest cosine similarity. We believe that this is because in the high-dimensional vector space composed of all texts, the item text representations are only sparse points, so the generated text can easily match the correct item in the top-1 similarity.

Study of Input Templates. In this section, we explore the impact of using different natural language templates to package user and item interaction sequences on the recommendation results. First, we fix the text instruction of the input text and try using different texts to separate each item. The results as shown in Figure 5a. Afterwards, we fix the separator between each item and modify the text instruction of the input text. We conduct experiments on the MovieLens-1M dataset and compare the results as shown in Figure 5b.

According to the experimental results, we find that different natural language templates have a certain impact on the recommendation results. Firstly, incorporating text instructions at the beginning of interaction sequences can enhance the performance. Furthermore, language models show a preference for logical expressions. More logical expressions are favored by the language model, while some inferior templates may even yield negative effects when compared to recommendations without any additional text modification. This observation can be attributed to the characteristics of PLMs, which include human expression and behavioral logic in their training corpus. Therefore, prompts with templates that are more in line with human cognition will perform better.



(a) Separators. (b) Text instructions.

Figure 5: The impact of different natural language templates on sequential recommendation results.

5 Conclusion

In conclusion, we propose LANCER, a novel paradigm that utilizes a language model as a sequential recommender, effectively integrating domain-specific knowledge and general knowledge of the PLM to learn dynamic user interests. Our approach employs a knowledge prompt to facilitate the acquisition of item content and a reasoning prompt to enable the integration of both domain-specific knowledge and user behavior for personalized recommendation. We highlight the potential of our approach in bridging the gap between language models and recommender systems and provide experimental results to support its effectiveness in incorporating item content understanding. We hope that this work will inspire future researchers to delve into content-based modeling for recommendation using language models, thereby fostering iterative advancements in the field of recommender systems.

References

[1] Mingyue Cheng, Zhiding Liu, Qi Liu, Shenyang Ge, and Enhong Chen. Towards automatic discovering of deep hybrid network architecture for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 1923–1932, 2022.

[2] Qiang Cui, Shu Wu, Qiang Liu, Wen Zhong, and Liang Wang. Mv-rnn: A multi-view recurrent neural network for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 32(2):317–331, 2018.

[3] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.

[4] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[5] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 143–153, 2021.

[6] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315, 2022.

[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[8] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[9] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[12] Lei Li, Yongfeng Zhang, and Li Chen. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 41(4):1–26, 2023.

[13] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

[14] Peng Liu, Lemei Zhang, and Jon Atle Gulla. Pre-train, prompt and recommendation: A comprehensive survey of language modelling paradigm adaptations in recommender systems. *arXiv preprint arXiv:2302.03735*, 2023.

[15] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, 2022.

[16] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

[17] Zhiding Liu, Mingyue Cheng, Zhi Li, Qi Liu, and Enhong Chen. One person, one model - learning compound router for sequential recommendation. In Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu, editors, *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pages 289–298. IEEE, 2022.

[18] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

[19] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[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 *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[22] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[23] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[24] Mengting Wan and Julian J. McAuley. Item recommendation on monotonic behavior chains. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 86–94. ACM, 2018.

[25] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian J. McAuley. Fine-grained spoiler detection from large-scale review corpora. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2605–2610. Association for Computational Linguistics, 2019.

[26] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. Transrec: Learning transferable recommendation from mixture-of-modality feedback. *arXiv preprint arXiv:2206.06190*, 2022.

[27] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*, 2019.

[28] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606, 2020.

[29] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. *arXiv preprint arXiv:2303.13835*, 2023.

[30] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326, 2019.

[31] Yuhui Zhang, HAO DING, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. Language models as recommender systems: Evaluations and limitations. In *I (Still) Can't Believe It's Not Better! NeurIPS 2021 Workshop*, 2021.