Text Is All You Need: Learning Language Representations for Sequential Recommendation

Jiacheng Li
University of California, San Diego
j9li@eng.ucsd.edu

Ming Wang
Amazon, United States
mingww@amazon.com

Jin Li
Amazon, United States
jincli@amazon.com

Jinmiao Fu
Amazon, United States
jinnmiaof@amazon.com

Xin Shen
Amazon, United States
xinshen@amazon.com

Jingbo Shang
University of California, San Diego
jshang@eng.ucsd.edu

Julian McAuley
University of California, San Diego
jmcauley@eng.ucsd.edu

ABSTRACT

Sequential recommendation aims to model dynamic user behavior from historical interactions. Existing methods rely on either explicit item IDs or general textual features for sequence modeling to understand user preferences. While promising, these approaches still struggle to model cold-start items or transfer knowledge to new datasets. In this paper, we propose to model user preferences and item features as language representations that can be generalized to new items and datasets. To this end, we present a novel framework, named Recformer, which effectively learns language representations for sequential recommendation. Specifically, we propose to formulate an item as a "sentence" (word sequence) by flattening item key-value attributes described by text so that an item sequence for a user becomes a sequence of sentences. For recommendation, Recformer is trained to understand the "sentence" sequence and retrieve the next "sentence". To encode item sequences, we design a bi-directional Transformer similar to the model Longformer but with different embedding layers for sequential recommendation. For effective representation learning, we propose novel pretraining and finetuning methods which combine language understanding and recommendation tasks. Therefore, Recformer can effectively recommend the next item based on language representations. Extensive experiments conducted on six datasets demonstrate the effectiveness of Recformer for sequential recommendation, especially in low-resource and cold-start settings.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

sequential recommendation, language models

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599519

ACM Reference Format:
Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599519

1 INTRODUCTION

Sequential recommender systems model historical user interactions as temporally-ordered sequences to recommend potential items that users are interested in. Sequential recommenders [11, 14, 25, 27] can capture both short-term and long-term preferences of users and hence are widely used in different recommendation scenarios.

Various methods have been proposed to improve the performance of sequential recommendation, including Markov Chains [9, 25], RNN/CNN models [11, 17, 28, 34] and self-attentive models [14, 19, 27]. Traditional sequential recommendation models convert items into IDs and create item embedding tables for encoding. Item embeddings are learned from sequences of user interactions. To enrich item features, some approaches [4, 20, 37, 38] incorporate item contexts such as item textual information or categorical features into ID embeddings. While ID-based methods are promising, they struggle to understand cold-start items or conduct cross-domain recommendations where models are trained and then applied to different recommendation scenarios. Item-specific IDs prevent models from learning transferable knowledge from training data for cold-start items and new datasets. As a result, item IDs limit the performance of sequential recommenders on cold-start items and we have to re-train a sequential recommender for continually added new items. Therefore, transferable recommenders can benefit both cold-start items and new-domain datasets.

To develop transferable recommender systems, previous studies usually assume shared information such as overlapping users/items [13, 26, 39] and common features [29] is available and then reduce the gap between source and target domains by learning either semantic mappings [39] or transferable components [16]. Such assumptions are rarely true in real applications because items in different domains (e.g., Laptops and T-shirts) usually contain different features for recommendation. Therefore, to have effective

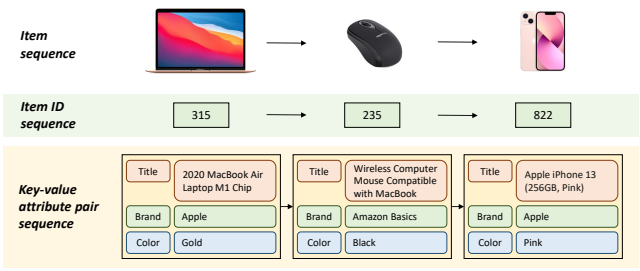

Figure 1: Input data comparison between item ID sequences for traditional sequential recommendation and key-value attribute pair sequences used in RECFORMER.

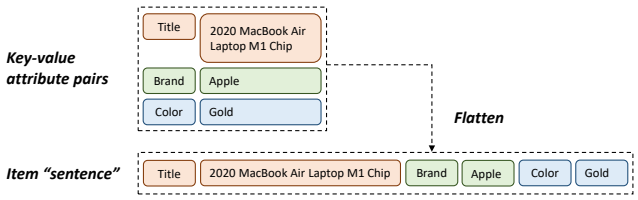

Figure 2: Model input construction. Flatten key-value attribute pairs into an item "sentence".

cross-domain transfer, recent works [7, 12] leverage the generality of natural language texts (e.g., titles, descriptions of items) for common knowledge in different domains. The basic idea is to employ pre-trained language models such as BERT [6] to obtain text representations and then learn the transformation from text representations to item representations. The knowledge of the transformation can be transferred across different domains and shows promising performance. However, such frameworks of learning transformation from language to items have several limitations: (1) Pre-trained language models are usually trained on a general language corpus (e.g., Wikipedia) serving natural language tasks that have a different language domain from item texts (e.g., concatenation of item attributes), hence text representations from pre-trained language models for items are usually sub-optimal. (2) Text representations from pre-trained language models are not able to learn the importance of different item attributes and only provide coarse-grained (sentence-level) textual features but cannot learn fine-grained (word-level) user preferences for recommendations (e.g., find the same color in recent interactions for clothing recommendations). (3) Due to the independent training of pre-trained language models (by language understanding tasks, e.g., Masked Language Modeling) and transformation models (by recommendation tasks, e.g., next item prediction), the potential ability of models to understand language for recommendations has not been fully developed (by joint training).

With the above limitations in mind, we aim to unify the frameworks of natural language understanding and recommendations in an ID-free sequential recommendation paradigm. The pre-trained language models [6, 15, 23, 24] benefit various downstream natural language processing tasks due to their transferable knowledge from pre-training. The basic idea of this paper is to use the generality of language models through joint training of language understanding and sequential recommendations. To this end, there are three major challenges to be solved. First, previous text-based methods [7, 12] usually have their specific item texts (e.g., item descriptions, concatenation of item attributes). Instead of specific data types, we need to find a universal input data format of items for language models that is flexible enough to different kinds of textual item information. Second, it is not clear how to model languages and sequential transitions of items in one framework. Existing language models are not able to incorporate sequential patterns of items and cannot learn the alignment between items and item texts. Third, a training and inference framework is necessary to bridge the gap between natural languages and recommendations like how to efficiently rank items based on language models without trained item embeddings.

To address the above problems, we propose RECFORMER, a framework that can learn language representations for sequential recommendation. Overall, our approach takes a text sequence of historical items as input and predicts the next item based on language understanding. Specifically, as shown in Figure 1, we first formulate an item as key-value attribute pairs which can include any textual information such as the title, color, brand of an item. Different items can include different attributes as item texts. Then, to encode a sequence of key-value attribute pairs, we propose a novel bi-directional Transformer [30] based on Longformer structure [2] but with different embeddings for item texts to learn item sequential patterns. Finally, to effectively learn language representations for recommendation, we design the learning framework for the model including pre-training, finetuning and inference processes. Based on the above methods, RECFORMER can effectively recommend the next items based on item text representations. Furthermore, the knowledge learned from training can be transferred to cold-start items or a new recommendation scenario.

To evaluate RECFORMER, we conduct extensive experiments on real-world datasets from different domains. Experimental results show that our method can achieve 15.83% and 39.78% (NDCG@10) performance improvements under fully-supervised and zero-shot sequential recommendation settings respectively.[1] Our contributions in this paper can be summarized as follows:

- We formulate items as key-value attribute pairs for the ID-free sequential recommendation and propose a novel bi-directional Transformer structure to encode sequences of key-value pairs.
- We design the learning framework that helps the model learn users' preferences and then recommend items based on language representations and transfer knowledge into different recommendation domains and cold-start items.
- Extensive experiments are conducted to show the effectiveness of our method. Results show that RECFORMER outperforms baselines for sequential recommendation and largely improves knowledge transfer as shown by zero-shot and cold-start settings.

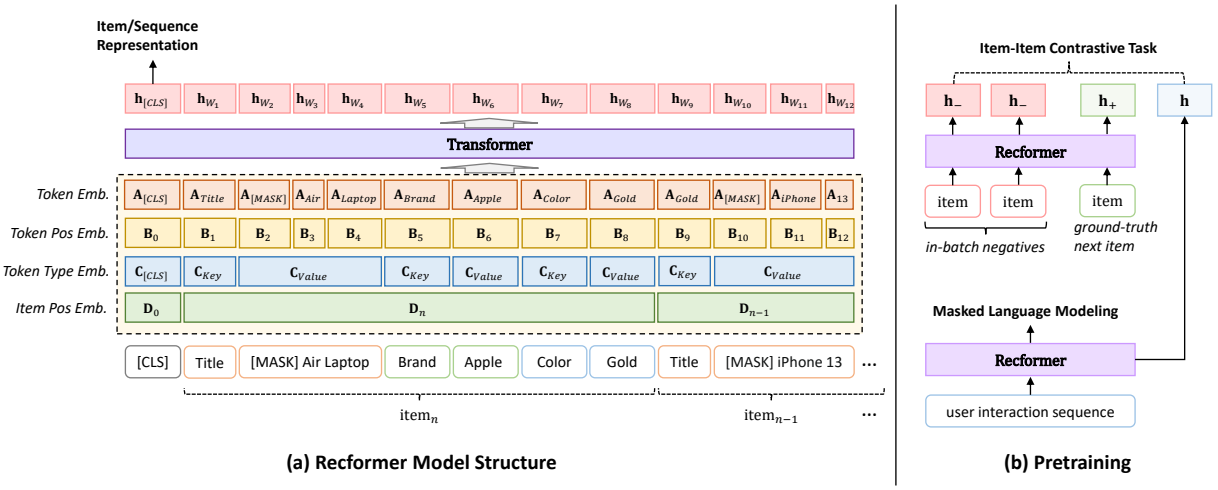

Figure 3: The overall framework of Recformer.

2 METHODOLOGY

In this section, we present Recformer which can learn language representations for sequential recommendation and effectively transfer and generalize to new recommendation scenarios.

2.1 Problem Setup and Formulation

In the setting of sequential recommendation, we are given an item set $\mathcal{I}$ and a user's interaction sequence $s = \{i_1, i_2, \ldots, i_n\}$ in temporal order where n is the length of s and $i \in \mathcal{I}$. Based on s, we seek to predict the next item. In previous sequential recommendation methods, each interacted item i is associated with a unique item ID. In this paper, each item i has a corresponding attribute dictionary D_i containing key-value attribute pairs $\{(k_1, v_1), (k_2, v_2), \ldots, (k_m, v_m)\}$ where k denotes an attribute name (e.g., *Color*) and v is the corresponding value (e.g., *Black*). k and v are both described by natural languages and contain words $(k, v) = \{w_1^k, \ldots, w_c^k, w_1^v, \ldots, w_c^v\}$, where w^k and w^v are words of k and v from a shared vocabulary in the language model and c denotes the truncated length of text. An attribute dictionary D_i can include all kinds of item textual information such as titles, descriptions, colors, etc. As shown in Figure 2, to feed the attribute dictionary D_i into a language model, we flatten key-value attribute pairs into $T_i = \{k1, v1, k2, v2, \ldots, k_m, v_m\}$ to obtain an item "sentence" as the input data. Unlike previous sequential recommenders [12, 37] using both text and item IDs, in this study, we use *only text* for the sequential recommendation.

2.2 Recformer

Figure 3 (a) shows the architecture of Recformer. The model has a similar structure as Longformer [2] which adopts a multi-layer bidirectional Transformer [30] with an attention mechanism that scales linearly with sequence length. We consider only computing efficiency for using Longformer but our method is open to other bidirectional Transformer structures such as BERT [6] and BigBird [36].

[1]Code will be released upon acceptance.

2.2.1 Model Inputs. As introduced in Section 2.1, for each item i and corresponding attribute dictionary D_i, we flatten the dictionary into an item "sentence" $T_i = \{k1, v1, k2, v2, \ldots, k_m, v_m\}$ where k and v are described by words, formally $(k, v) = \{w_1^k, \ldots, w_c^k, w_1^v, \ldots, w_c^v\}$. To encode a user's interaction sequence $s = \{i_1, i_2, \ldots, i_n\}$, we first reverse items in a sequence to $\{i_n, i_{n-1}, \ldots, i_1\}$ because intuitively recent items (i.e., $i_n, i_{n-1}, \ldots$) are important for the next item prediction and reversed sequences can make sure recent items are included in the input data. Then, we use the item "sentences" to replace items and add a special token `[CLS]` at the beginning of sequences. Hence, model inputs are denoted as:

$$X = \{\texttt{[CLS]}, T_n, T_{n-1}, \ldots, T_1\} \tag{1}$$

where X is a sequence of words containing all items and corresponding attributes the user interacted with in the historical interactions.

2.2.2 Embedding Layer. The target of Recformer is to understand the model input X from both language understanding and sequential patterns in recommendations. The key idea in our work is to combine the embedding layers from language models [6, 21] and self-attentive sequential recommenders [14, 27]. Hence, Recformer contains four embeddings as follows:

- **Token embedding** represents the corresponding tokens. We denote the word token embedding by $\mathbf{A} \in \mathbb{R}^{V_w \times d}$, where V_w is the number of words in our vocabulary and d is the embedding dimension. Recformer does not have item embeddings as previous sequential recommenders and hence Recformer understands items in interaction sequences mainly based on these token embeddings. The size of token embeddings is a constant for different recommendation scenarios; hence, our model size is irrelevant to the number of items.
- **Token position embedding** represents the position of tokens in a sequence. A word appearing at the i-th position in the sequence X is represented as $\mathbf{B}_i \in \mathbb{R}^d$. Similar to language models, token position embedding is designed to help Transformer understand the sequential patterns of words.
- **Token type embedding** represents where a token comes from. Specifically, the token type embedding totally contains

three vectors $\mathbf{C}_{\texttt{[CLS]}}, \mathbf{C}_{Key}, \mathbf{C}_{Value} \in \mathbb{R}^d$ to represent if a token comes from [CLS], attribute keys, or attribute values respectively. Different types of tokens usually have different importance for the next item prediction. For example, because most items usually have the same attribute keys in a recommendation dataset, models with token type embedding will recognize repeated words from the same attribute keys.

- **Item position embedding** represents the position of items in a sequence. A word from attributes of the k-th item in the sequence X is represented as $\mathbf{D}_k \in \mathbb{R}^d$ and $\mathbf{D} \in \mathbb{R}^{n \times d}$ where n is the maximum length of a user's interaction sequence s. Same as previous self-attentive sequential recommenders [14, 27], the item position embedding is a key component for item sequential pattern learning. In Recformer, the item position embedding can also help the model learn the alignment between word tokens and items.

Therefore, given a word w from the input sequence X, the input embedding is calculated as the summation of four different embeddings followed by layer normalization [1]:

$$\mathbf{E}_w = \text{LayerNorm}(\mathbf{A}_w + \mathbf{B}_w + \mathbf{C}_w + \mathbf{D}_w) \tag{2}$$

where $\mathbf{E}_w \in \mathbb{R}^d$.

The embedding of model inputs X is a sequence of $\mathbf{E}_w$,

$$\mathbf{E}_X = [\mathbf{E}_{\texttt{[CLS]}}, \mathbf{E}_{w_1}, \ldots, \mathbf{E}_{w_l}] \tag{3}$$

where $\mathbf{E}_X \in \mathbb{R}^{(l+1) \times d}$ and l is the maximum length of tokens in a user's interaction sequence.

2.2.3 Item or Sequence Representations. To encode $\mathbf{E}_X$, we employ the bidirectional Transformer structure Longformer [2] as our encoder. Because X is usually a long sequence, the local windowed attention in Longformer can help us efficiently encode $\mathbf{E}_X$. As the standard settings in Longformer for document understanding, the special token [CLS] has global attention but other tokens use the local windowed attention. Hence, Recformer computes d-dimensional word representations as follows:

$$[\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_{w_1}, \ldots, \mathbf{h}_{w_l}] = \text{Longformer}([\mathbf{E}_{\texttt{[CLS]}}, \mathbf{E}_{w_1}, \ldots, \mathbf{E}_{w_l}]) \tag{4}$$

where $\mathbf{h}_w \in \mathbb{R}^d$. Similar to the language models used for sentence representations, the representation of the first token $\mathbf{h}_{\texttt{[CLS]}}$ is used as the sequence representation.

In Recformer, we do not maintain an embedding table for items. Instead, we view the item as a special case of the interaction sequence with only one item. For each item i, we construct its item "sentence" T_i and use $X = \{\texttt{[CLS]}, T_i\}$ as the model input to get the sequence representation $\mathbf{h}_{\texttt{[CLS]}}$ as the item representation $\mathbf{h}_i$.

2.2.4 Prediction. We predict the next item based on the cosine similarity between a user's interaction sequence s and item i. Formally, after obtaining the sequence representation $\mathbf{h}_s$ and the item representation $\mathbf{h}_i$ as introduced in Section 2.2.3, we calculate the scores between s and i as follows:

$$r_{i,s} = \frac{\mathbf{h}_i^\top \mathbf{h}_s}{\|\mathbf{h}_i\| \cdot \|\mathbf{h}_s\|} \tag{5}$$

where $r_{i,s} \in \mathbb{R}$ is the relevance of item i being the next item given s. To predict the next item, we calculate $r_{i,s}$ for all items [2] in the item set $\mathcal{I}$ and select item with the highest $r_{i,s}$ as the next item:

$$\hat{i_s} = \text{argmax}_{i \in \mathcal{I}}(r_{i,s}) \tag{6}$$

where $\hat{i_s}$ is the predicted item given user interaction sequence s.

2.3 Learning Framework

To have an effective and efficient language model for the sequential recommendation, we propose our learning framework for Recformer including pre-training and two-stage finetuning.

2.3.1 Pre-training. The target of pre-training is to obtain a high-quality parameter initialization for downstream tasks. Different from previous sequential recommendation pre-training methods which consider only recommendations, we need to consider both language understanding and recommendations. Hence, to pre-train Recformer, we adopt two tasks: (1) Masked Language Modeling (MLM) and (2) an item-item contrastive task.

Masked Language Modeling (MLM) [6] is an effective pre-training method for language understanding and has been widely used for various NLP pre-training tasks such as sentence understanding [8], phrase understanding [18]. Adding MLM as an auxiliary task will prevent language models from forgetting the word semantics when models are jointly trained with other specific tasks. For recommendation tasks, MLM can also eliminate the language domain gap between a general language corpus and item texts. In particular, following BERT [6], the training data generator chooses 15% of the token positions at random for prediction. If the token is selected, we replace the token with (1) the [MASK] with probability 80%; (2) a random token with probability 10%; (3) the unchanged token with probability 10%. The MLM loss is calculated as:

$$\mathbf{m} = \text{LayerNorm}(\text{GELU}(\mathbf{W}_h \mathbf{h}_w + \mathbf{b}_h)) \tag{7}$$

$$p = \text{Softmax}(\mathbf{W}_0 \mathbf{m} + \mathbf{b}_0) \tag{8}$$

$$\mathcal{L}_{\text{MLM}} = -\sum_{i=0}^{|\mathcal{V}|} y_i \log(p_i) \tag{9}$$

where $\mathbf{W}_h \in \mathbb{R}^{d \times d}$, $\mathbf{b}_h \in \mathbb{R}^d$, $\mathbf{W}_0 \in \mathbb{R}^{|\mathcal{V}| \times d}$, $\mathbf{b}_0 \in \mathbb{R}^{|\mathcal{V}|}$, GELU is the GELU activation function [10] and $\mathcal{V}$ is the vocabulary used in the language model.

Another pre-training task for Recformer is the item-item contrastive (IIC) task which is widely used in the next item prediction for recommendations. We use the ground-truth next items as positive instances following previous works [12, 14, 27]. However, for negative instances, we adopt in-batch next items as negative instances instead of negative sampling [14] or fully softmax [12, 27]. Previous recommenders maintain an item embedding table, hence they can easily retrieve item embeddings for training and update embeddings. In our case, item embeddings are from Recformer, so it is infeasible to re-encode items (from sampling or full set) per batch for training. In-batch negative instances [3] are using ground truth items of other instance sequences in the same batch as negative items. Although it is possible to provide false negatives, false negatives are less likely in the pre-training dataset with a large size.

[2] For efficient calculation, we encode all items in advance for score calculation.

Algorithm 1: Two-Stage Finetuning

Input: D_{train}, D_{valid}, $\mathcal{I}$, M
Hyper-parameters: n_{epoch}
Output: M', $\mathbf{I}'$

1: $M \leftarrow$ initialized with pre-trained parameters
2: $p \leftarrow$ metrics are initialized with 0
Stage 1
3: **for** n in n_{epoch} **do**
4: $\quad \mathbf{I} \leftarrow \text{Encode}(M, \mathcal{I})$
5: $\quad M \leftarrow \text{Train}(M, \mathbf{I}, D_{\text{train}})$
6: $\quad p' \leftarrow \text{Evaluate}(M, \mathbf{I}, D_{\text{valid}})$
7: $\quad$ **if** $p' > p$ **then**
8: $\quad\quad M', \mathbf{I}' \leftarrow M, \mathbf{I}$
9: $\quad\quad p \leftarrow p'$
10: $\quad$ **end if**
11: **end for**
Stage 2
12: $M \leftarrow M'$
13: **for** n in n_{epoch} **do**
14: $\quad M \leftarrow \text{Train}(M, \mathbf{I}', D_{\text{train}})$
15: $\quad p' \leftarrow \text{Evaluate}(M, \mathbf{I}', D_{\text{valid}})$
16: $\quad$ **if** $p' > p$ **then**
17: $\quad\quad M' \leftarrow M$
18: $\quad\quad p \leftarrow p'$
19: $\quad$ **end if**
20: **end for**
21: **return** M', $\mathbf{I}'$

Furthermore, the target of pre-training is to provide high-quality initialized parameters and we have the finetuning with accurate supervision for downstream tasks. Therefore, we claim that in-batch negatives will not hurt the recommendation performance but have much higher training efficiency than accurate supervision. Formally, the item-item contrastive loss is calculated as:

$$\mathcal{L}_{\text{IIC}} = -\log \frac{e^{\text{sim}(\mathbf{h}_s, \mathbf{h}_i^+)/\tau}}{\sum_{i \in \mathcal{B}} e^{\text{sim}(\mathbf{h}_s, \mathbf{h}_i)/\tau}} \tag{10}$$

where sim is the similarity introduced in Equation (5); $\mathbf{h}_i^+$ is the representation of the ground truth next item; $\mathcal{B}$ is the ground truth item set in one batch and τ is a temperature parameter.

At the pre-training stage, we use a multi-task training strategy to jointly optimize Recformer:

$$\mathcal{L}_{\text{PT}} = \mathcal{L}_{\text{IIC}} + \lambda \cdot \mathcal{L}_{\text{MLM}} \tag{11}$$

where λ is a hyper-parameter to control the weight of MLM task loss. The pre-trained model will be fine-tuned for new scenarios.

2.3.2 Two-Stage Finetuning. Similar to pre-training, we do not maintain an independent item embedding table. Instead, we encode items by Recformer. However, in-batch negatives cannot provide accurate supervision in a small dataset because it is likely to have false negatives which undermine recommendation performance. To solve this problem, we propose two-stage finetuning as shown in Algorithm 1. The key idea is to maintain an item feature matrix $\mathbf{I} \in \mathbb{R}^{|\mathcal{I}| \times d}$. Different from the item embedding table, $\mathbf{I}$ is not learnable and all item features are encoded from Recformer. As shown in Algorithm 1, our proposed finetuning method has two stages. In stage 1, $\mathbf{I}$ is updated (line 4) per epoch,[3] whereas, in stage 2 we freeze $\mathbf{I}$ and update only parameters in model M. The basic idea is that although the model is already pre-trained, item representations from the pre-trained model can still be improved by further training on downstream datasets. It is expensive to re-encode all items in every batch hence we re-encode all items in every epoch to update $\mathbf{I}$ (line 4) and use $\mathbf{I}$ as supervision for item-item contrastive learning (line 5). After obtaining the best item representations, we re-initialize the model with the corresponding parameters (line 12) and start stage 2. Since $\mathbf{I}$ keeps updating in stage 1, the supervision for finetuning is also changing. In this case, the model is hard to be optimized to have the best performance. Therefore, we freeze $\mathbf{I}$ and continue training the model until achieving the best performance on the validation dataset.

The learning task used in finetuning is item-item contrastive learning which is the same as pre-training but with fully softmax instead of in-batch negatives. The finetuning loss is calculated as:

$$\mathcal{L}_{\text{FT}} = -\log \frac{e^{\text{sim}(\mathbf{h}_s, \mathbf{I}_i^+)/\tau}}{\sum_{i \in \mathcal{I}} e^{\text{sim}(\mathbf{h}_s, \mathbf{I}_i)/\tau}} \tag{12}$$

where $\mathbf{I}_i$ is the item feature of item i.

2.4 Discussion

In this section, we briefly compare Recformer to other sequential recommendation methods to highlight the novelty of our method.

Traditional sequential recommenders such as GRU4Rec [11], SASRec [14] and BERT4Rec [27] rely on item IDs and corresponding trainable item embeddings to train a sequential model for recommendations. These item embeddings are learned from sequential patterns of user interactions. However, as mentioned in [20], these approaches suffer from data sparsity and can not perform well with cold-start items.

To reduce the dependence on item IDs, some **context-aware sequential recommenders** such as UniSRec [12], S^3-Rec [38], ZESRec [7] are proposed to incorporate side information (e.g., categories, titles) as prior knowledge for recommendations. All of these approaches rely on a feature extractor such as BERT [6] to obtain item feature vectors and then fuse these vectors into item representations with an independent sequential model.

In this paper, we explore conducting sequential recommendations in a new paradigm that learns language representations for the next item recommendations. Instead of trainable item embeddings or fixed item features from language models, we bridge the gap between natural language understanding and sequential recommendation to directly learn representations of items and user sequences based on words. We expect the generality of natural language can improve the transferability of recommenders in order to benefit new domain adaptation and cold-start item understanding.

3 EXPERIMENTS

In this section, we empirically show the effectiveness of our proposed model Recformer and learning framework.

[3] Updating means encoding all items with Recformer.

Table 1: Statistics of the datasets after preprocessing. Avg. n denotes the average length of item sequences.

Datasets	#Users	#Items	#Inters.	Avg. n	Density
Pre-training	3,613,906	1,022,274	33,588,165	9.29	9.1e-6
-Training	3,501,527	954,672	32,291,280	9.22	9.0e-6
-Validation	112,379	67,602	1,296,885	11.54	1.7e-4
Scientific	11,041	5,327	76,896	6.96	1.3e-3
Instruments	27,530	10,611	231,312	8.40	7.9e-4
Arts	56,210	22,855	492,492	8.76	3.8e-4
Office	101,501	27,932	798,914	7.87	2.8e-4
Games	11,036	15,402	100,255	9.08	5.9e-4
Pet	47,569	37,970	420,662	8.84	2.3e-4

3.1 Experimental Setup

3.1.1 Datasets. To evaluate the performance of Recformer, we conduct pre-training and finetuning on different categories of Amazon review datasets [22]. The statistics of datasets after preprocessing are shown in Table 1.

For pre-training, seven categories are selected as training data including "*Automotive*", "*Cell Phones and Accessories*", "*Clothing Shoes and Jewelry*", "*Electronics*", "*Grocery and Gourmet Food*", "*Home and Kitchen*", "*Movies and TV*", and one category "*CDs and Vinyl*" is left out as validation data. Datasets from these categories are used as source domain datasets.

For finetuning, we select six categories including "*Industrial and Scientific*", "*Musical Instruments*", "*Arts, Crafts and Sewing*", "*Office Products*", "*Video Games*", "*Pet Supplies*", as target domain datasets to evaluate Recformer.

For pre-training and finetuning, we use the five-core datasets provided by the data source and filter items whose *title* is missing. Then we group the interactions by users and sort them by timestamp ascendingly. Following previous work [12], we select item attributes *title*, *categories* and *brand* as key-value pairs for items.

3.1.2 Baselines. We compare three groups of works as our baselines which include methods with only item IDs; methods using item IDs and treating item text as side information; and methods using only item texts as inputs.

(1) ID-Only methods:

- **GRU4Rec** [11] adopts RNNs to model user action sequences for session-based recommendations. We treat each user's interaction sequence as a session.
- **SASRec** [14] uses a directional self-attentive model to capture item correlations within a sequence.
- **BERT4Rec** [27] employs a bi-directional self-attentive model with the cloze objective for modeling user behavior sequences.
- **RecGURU** [16] proposes to pre-train sequence representations with an autoencoder in an adversarial learning paradigm. We do not consider overlapped users for this method in our setting.

(2) ID-Text methods:

- **FDSA** [37] uses a self-attentive model to capture item and feature transition patterns.
- **S^3-Rec** [38] pre-trains sequential models with mutual information maximization to learn the correlations among attributes, items, subsequences, and sequences.

(3) Text-Only methods:

- **ZESRec** [7] encodes item texts with a pre-trained language model as item features. We pre-train this method and finetune the model on six downstream datasets.
- **UniSRec** [12] uses textual item representations from a pre-trained language model and adapts to a new domain using an MoE-enhance adaptor. We initialize the model with the pre-trained parameters provided by the authors and finetune the model on target domains.

3.1.3 Evaluation Settings. To evaluate the performance of sequential recommendation, we adopt three widely used metrics NDCG@N, Recall@N and MRR, where N is set to 10. For data splitting of finetuning datasets, we apply the leave-one-out strategy [14] for evaluation: the most recent item in an interaction sequence is used for testing, the second most recent item for validation and the remaining data for training. We rank the ground-truth item of each sequence among all items for evaluation and report the average scores of all sequences in the test data.

3.1.4 Implementation Details. We build Recformer based on Longformer implemented by Huggingface [4]. For efficient computing, we set the size of the local attention windows in Longformer to 64. The maximum number of tokens is 32 for each attribute and 1,024 for each interaction sequence (i.e., X in Equation (1)). The maximum number of items in a user sequence is 50 for all baselines and Recformer. The temperature parameter τ is 0.05 and the weight of MLM loss λ is 0.1. Other than token type embedding and item position embedding in Recformer, other parameters are initialized with pre-trained parameters of Longformer [5] before pre-training. The batch size is 64 for pre-training and 16 for finetuning. We optimize Recformer with Adam optimizer with learning rate 5e-5 and adopt early stop with the patience of 5 epochs to prevent overfitting. For baselines, we use the suggested settings introduced in [12].

3.2 Overall Performance

We compare Recformer to baselines on six datasets across different recommendation domains. Results are shown in Table 2.

For baselines, ID-Text methods (i.e., FDSA and S^3-Rec) achieve better results compared to ID-Only and Text-Only methods in general. Because ID-Text methods include item IDs and content features, they can learn both content-based information and sequential patterns from finetuning. Comparing Text-Only methods and ID-Only methods, we can find that on the Scientific, Instruments, and Pet datasets, Text-Only methods perform better than ID-Only methods. A possible reason is that the item transitions in these three datasets are highly related to item texts (i.e., title, brand) hence text-only methods can recommend the next item based on content similarity.

Our proposed method Recformer, achieves the best overall performance on all datasets except the Recall@10 of Instruments. Recformer improves the NDCG@10 by 15.83% and MRR by 15.99% on average over the second best results. Different from baselines,

[4] https://huggingface.co/docs/transformers/
[5] https://huggingface.co/allenai/longformer-base-4096

Table 2: Performance comparison of different recommendation models. The best and the second-best performance is bold and underlined respectively. Improv. denotes the relative improvement of RECFORMER over the best baselines.

Dataset	Metric	ID-Only Methods				ID-Text Methods		Text-Only Methods			Improv.
		GRU4Rec	SASRec	BERT4Rec	RecGURU	FDSA	S^3-Rec	ZESRec	UniSRec	RECFORMER	
Scientific	NDCG@10	0.0826	0.0797	0.0790	0.0575	0.0716	0.0451	0.0843	0.0862	**0.1027**	19.14%
	Recall@10	0.1055	0.1305	0.1061	0.0781	0.0967	0.0804	0.1260	0.1255	**0.1448**	10.96%
	MRR	0.0702	0.0696	0.0759	0.0566	0.0692	0.0392	0.0745	0.0786	**0.0951**	20.99%
Instruments	NDCG@10	0.0633	0.0634	0.0707	0.0468	0.0731	0.0797	0.0694	0.0785	**0.0830**	4.14%
	Recall@10	0.0969	0.0995	0.0972	0.0617	0.1006	0.1110	0.1078	**0.1119**	0.1052	-
	MRR	0.0707	0.0577	0.0677	0.0460	0.0748	0.0755	0.0633	0.0740	**0.0807**	6.89%
Arts	NDCG@10	0.1075	0.0848	0.0942	0.0525	0.0994	0.1026	0.0970	0.0894	**0.1252**	16.47%
	Recall@10	0.1317	0.1342	0.1236	0.0742	0.1209	0.1399	0.1349	0.1333	**0.1614**	15.37%
	MRR	0.1041	0.0742	0.0899	0.0488	0.0941	0.1057	0.0870	0.0798	**0.1189**	12.49%
Office	NDCG@10	0.0761	0.0832	0.0972	0.0500	0.0922	0.0911	0.0865	0.0919	**0.1141**	17.39%
	Recall@10	0.1053	0.1196	0.1205	0.0647	0.1285	0.1186	0.1199	0.1262	**0.1403**	9.18%
	MRR	0.0731	0.0751	0.0932	0.0483	0.0972	0.0957	0.0797	0.0848	**0.1089**	12.04%
Games	NDCG@10	0.0586	0.0547	0.0628	0.0386	0.0600	0.0532	0.0530	0.0580	**0.0684**	8.92%
	Recall@10	0.0988	0.0953	0.1029	0.0479	0.0931	0.0879	0.0844	0.0923	**0.1039**	0.97%
	MRR	0.0539	0.0505	0.0585	0.0396	0.0546	0.0500	0.0505	0.0552	**0.0650**	11.11%
Pet	NDCG@10	0.0648	0.0569	0.0602	0.0366	0.0673	0.0742	0.0754	0.0702	**0.0972**	28.91%
	Recall@10	0.0781	0.0881	0.0765	0.0415	0.0949	0.1039	0.1018	0.0933	**0.1162**	11.84%
	MRR	0.0632	0.0507	0.0585	0.0371	0.0650	0.0710	0.0706	0.0650	**0.0940**	32.39%

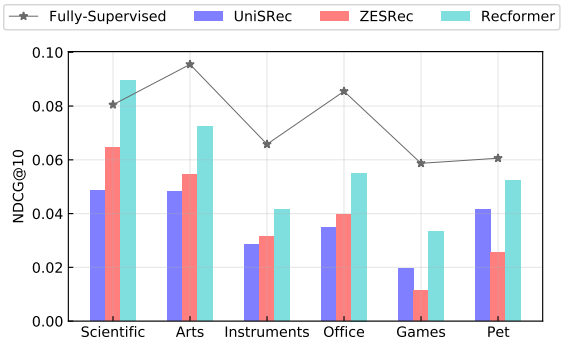

Figure 4: Performance (NDCG@10) of three Text-Only methods under the zero-shot setting. Fully-Supervised denotes the average scores of three classical ID-Only methods (i.e., SASRec, BERT4Rec, GRU4Rec) trained with all training data.

Figure 5: Performance (NDCG@10) of SASRec, UniSRec, RECFORMER over different sizes (i.e., 1%, 5%, 10%, 50%, 100%) of training data.

RECFORMER learns language representations for sequential recommendation without pre-trained language models or item IDs. With two-stage finetuning, RECFORMER can be effectively adapted to downstream domains and transferred knowledge from pre-training can consistently benefit finetuning tasks. The results illustrate the effectiveness of the proposed RECFORMER.

3.3 Low-Resource Performance

3.3.1 Zero-Shot. To show the effectiveness of pre-training, we evaluate the zero-shot recommendation performance of three Text-Only methods (i.e., UniSRec, ZESRec, RECFORMER) and compare results to the average scores of three ID-Only methods fully trained on downstream datasets. The zero-shot recommendation setting requires models to learn knowledge from pre-training datasets and directly test on downstream datasets without further training. Hence, traditional ID-based methods cannot be evaluated in this setting. We evaluate the knowledge transferability of Text-Only methods in different recommendation scenarios. All results in six downstream datasets are shown in Figure 4. Overall, RECFORMER improves the zero-shot recommendation performance compared to UniSRec and ZESRec on six datasets. On the Scientific dataset, RECFORMER performs better than the average performance of three ID-Only methods trained with full training sets. These results show that (1) natural language is promising as a general item representation across different recommendation scenarios; (2) RECFORMER can effectively learn knowledge from pre-training and transfer learned knowledge to downstream tasks based on language understanding.

3.3.2 Low-Resource. We conduct experiments with SASRec, UniSRec and RECFORMER in low-resource settings. In this setting, we train models on downstream datasets with different ratios of training data and results are shown in Figure 5. We can see that methods with item text (i.e., UniSRec and RECFORMER) outperform ID-only method SASRec especially when less training data is available. This

Table 3: Performance of models compared between in-set items and cold-start items on four datasets. N@10 and R@10 stand for NDCG@10 and Recall@10 respectively.

		SASRec		UniSRec		Recformer	
Dataset	Metric	In-Set	Cold	In-Set	Cold	In-Set	Cold
Scientific	N@10	0.0775	0.0213	0.0864	0.0441	0.1042	0.0520
	R@10	0.1206	0.0384	0.1245	0.0721	0.1417	0.0897
Instruments	N@10	0.0669	0.0142	0.0715	0.0208	0.0916	0.0315
	R@10	0.1063	0.0309	0.1094	0.0319	0.1130	0.0468
Arts	N@10	0.1039	0.0071	0.1174	0.0395	0.1568	0.0406
	R@10	0.1645	0.0129	0.1736	0.0666	0.1866	0.0689
Pet	N@10	0.0597	0.0013	0.0771	0.0101	0.0994	0.0225
	R@10	0.0934	0.0019	0.1115	0.0175	0.1192	0.0400

indicates UniSRec and Recformer can incorporate prior knowledge and do recommendations based on item texts. In low-resource settings, most items in the test set are unseen during training for SASRec. Therefore, the embeddings of unseen items are randomly initialized and cannot provide high-quality representations for recommendations. After being trained with adequate data, SASRec could rapidly improve its performance. Recformer achieves the best performance over different ratios of training data. On the Scientific dataset, Recformer outperforms other methods by a large margin with 1% and 5% of training data.

3.4 Further Analysis

3.4.1 Performance w.r.t. Cold-Start Items. In this section, we simulate this scenario by splitting a dataset into two parts, i.e., an in-set dataset and cold-start dataset. Specifically, for the in-set dataset, we make sure all test items appear in the training data and all other test items (never appearing in training data) will be sent to the cold-start dataset. We train models on in-set datasets and test on both in-set and cold-start datasets. In this case, models never see the cold-start items during training and item embedding tables do not contain cold-start items. We compare the ID-only method SASRec and the Text-only method UniSRec to Recformer. For ID-based SASRec, we substitute items appearing only once in the training set with a cold token and after training, we add this cold token embedding to cold-start item embeddings to provide prior knowledge [6]. For UniSRec, cold-start items are represented by item texts and encoded by BERT which is identical to seen items. Recformer directly encode item texts to represent cold-start items.

Experimental results are shown in Table 3. We can see that Text-Only methods significantly outperform SASRec, especially on datasets with a large size (i.e., Arts, Pet). Because of randomly initialized cold-start item representations, the performance of SASRec is largely lower on cold-start items than in-set items. Hence, ID-only methods are not able to handle cold-start items and applying text is a promising direction. For Text-only methods, Recformer greatly improves performance on both in-set and cold-start datasets compared to UniSRec which indicates learning language representations is superior to obtaining text features for recommendations.

[6]We try to provide a reasonable method for ID-based baselines with cold-start items.

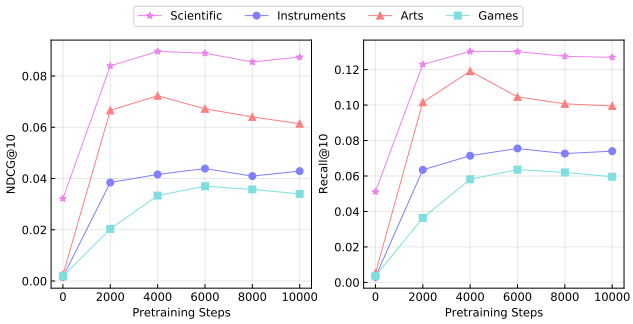

Figure 6: Recformer zero-shot recommendation performance (NDCG@10 and Recall@10) over different pre-training steps.

3.4.2 Ablation Study. We analyze how our proposed components influence the final sequential recommendation performance. The results are shown in Table 4. We introduce the variants and analyze their results respectively.

We first test the effectiveness of our proposed two-stage finetuning. In variant (1) w/o two-stage finetuning, we do not update item feature matrix **I** and only conduct finetuning based on **I** from pre-trained parameters. We find that compared to (0) Recformer, (1) has similar results on Scientific but has a large margin on Instruments since the pre-trained model has better pre-trained item representations on Scientific compared to Instruments (shown in Figure 4). Hence, our proposed two-stage finetuning can effectively improve the sub-optimal item representations from pre-training and further improve performance on downstream datasets.

Then, we investigate the effects of freezing/trainable word embeddings and item embeddings. In our default setting (1), we freeze the item feature matrix **I** and train word embeddings of Recformer. In variants (2)(3)(4), we try to train the item feature matrix or freeze word embeddings. Overall, on the Scientific dataset, the model with fixed item embeddings performs better than the model with trainable item embeddings, whereas on the Instruments dataset, our model performs well when item embeddings are trainable. The divergence can be eliminated by our two-stage finetuning strategy.

Variant (5) w/o pre-training finetunes Recformer from scratch. We can see that (0) Recformer significantly outperforms Variant (5) in both datasets because without pre-training, the item feature matrix **I** is not trained and cannot provide informative supervision during finetuning even if we update **I** by two-stage finetuning. These results show the effectiveness of pre-training.

Finally, we explore the effectiveness of our proposed model structure (i.e., item position embeddings and token type embeddings). Variant (6) removes the two embeddings and results show that the model in (6) causes performance decay on the instruments dataset which indicates the two embeddings are necessary when the gap between pre-training and finetuning is large.

3.4.3 Pre-training Steps vs. Performance. We investigate the zero-shot sequential recommendation performance on downstream tasks over different pre-training steps and results on four datasets are shown in Figure 6. The pre-training of natural language understanding usually requires a large number of training steps to achieve

Table 4: Ablation study on two downstream datasets. The best and the second-best scores are bold and underlined respectively.

Variants	Scientific			Instruments		
	NDCG@10	Recall@10	MRR	NDCG@10	Recall@10	MRR
(0) Recformer	**0.1027**	**0.1448**	**0.0951**	**0.0830**	**0.1052**	**0.0807**
(1) w/o two-stage finetuning	0.1023	<u>0.1442</u>	<u>0.0948</u>	0.0728	0.1005	0.0685
(1) + (2) freezing word emb. & item emb.	<u>0.1026</u>	0.1399	0.0942	0.0728	<u>0.1015</u>	0.0682
(1) + (3) trainable word emb. & item emb.	0.0970	0.1367	0.0873	<u>0.0802</u>	<u>0.1015</u>	0.0759
(1) + (4) trainable item emb. & freezing word emb.	0.0965	0.1383	0.0856	0.0801	0.1014	<u>0.0760</u>
(5) w/o pre-training	0.0722	0.1114	0.0650	0.0598	0.0732	0.0584
(6) w/o item position emb. & token type emb.	0.1018	0.1427	0.0945	0.0518	0.0670	0.0501

a promising result. However, we have a different situation in sequential recommendation. From Figure 6, we can see that most datasets already achieve their best performance after around 4,000 training steps and further pre-training may hurt the knowledge transferability on downstream tasks. We think there are two possible reasons: (1) We initialize most parameters from a Longformer model pre-trained by the MLM task. In this case, the model already has some essential knowledge of natural languages. The domain adaptation from a general language understanding to the item text understanding for recommendations should be fast. (2) Even if we include seven categories in the training data, there is still a language domain difference between pre-training data and downstream data since different item categories have their own specific vocabularies. For instance, the category *Electronics* has quite different words in item text compared to the *Pets* category.

4 RELATED WORK

4.1 Sequential Recommendation

Sequential recommendation [11, 14, 27] aims to predict the next item based on historical user interactions. Proposed methods model user interactions as a sequence ordered by their timestamps. Due to the ability to capture the long-term preferences and short-term dynamics of users, sequential recommendation methods show their effectiveness for personalization and attract a lot of studies. Early works [9, 25] apply the Markov Chain to model item-item transition relations based on matrix factorization. For deep learning methods, Convolutional Sequence Embedding (Caser) [28] views the embedding matrix of previous items as an "image" and applies convolutional operations to extract transitions. GRU4Rec [11] introduces Gated Recurrent Units (GRU) [5] to model user sequential patterns. With the development of the Transformer [30], recent studies [14, 27] widely use self-attention model for sequential recommendation. Although these approaches achieve promising performance, they struggle to learn transferable knowledge or understand cold-start items due to the dependence on IDs and item embeddings which are specific to items and datasets. Recently, researchers attempt to employ textual features as transferable item representations [7, 12]. These methods first obtain item features by encoding item texts with language models and then learn transferable item representations with an independent sequential model. Independent language understanding and sequential pattern learning still limit the capacity of the model to learn user interactions based on languages. In this paper, we explore unifying the language understanding and sequential recommendations into one Transformer framework. We aim to have a sequential recommendation method that can effectively model cold-start items and learn transferable sequential patterns for different recommendation scenarios.

4.2 Transfer Learning for Recommendation

Data sparsity and cold-start item understanding issues are challenging in recommender systems and recent studies [33, 39, 40] explore transferring knowledge across different domains to improve the recommendation at the target domain. Previous methods for knowledge transfer mainly rely on shared information between the source and target domains including common users [13, 31, 32, 35], items [26, 39] or attributes [29]. To learn common item features from different domains, pre-trained language models [6, 21] provide high-quality item features by encoding item texts (e.g., title, brand). Based on pre-trained item features, several methods [7, 12] are proposed to learn universal item representations by applying additional layers. In this work, we have the same target as previous transfer learning for recommendation (i.e., alleviate data sparsity and cold-start item issues). However, instead of relying on common users, items and attributes or encoding items with pre-trained language models, we directly learn language representations for sequential recommendation and hence transfer knowledge based on the generality of natural languages.

5 CONCLUSION

In this paper, we propose Recformer, a framework that can effectively learn language representations for sequential recommendation. To recommend the next item based on languages, we first formulate items as key-value attribute pairs instead of item IDs. Then, we propose a novel bi-directional Transformer model for sequence and item representations. The proposed structure can learn both natural languages and sequential patterns for recommendations. Furthermore, we design a learning framework including pre-training and finetuning that helps the model learn to recommend based on languages and transfer knowledge into different recommendation scenarios. Finally, extensive experiments are conducted to evaluate the effectiveness of Recformer under full-supervised and low-resource settings. Results show that Recformer largely outperforms existing methods in different settings, especially for the zero-shot and cold-start items recommendation which indicates Recformer can effectively transfer knowledge from training. An ablation study is conducted to show the effectiveness of our proposed components.

REFERENCES

[1] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *ArXiv* abs/1607.06450 (2016).

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *ArXiv* abs/2004.05150 (2020).

[3] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On Sampling Strategies for Neural Network-based Collaborative Filtering. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017).

[4] Yong-Guang Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. *Proceedings of the ACM Web Conference 2022* (2022).

[5] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *ArXiv* abs/1412.3555 (2014).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv* abs/1810.04805 (2019).

[7] Hao Ding, Yifei Ma, Anoop Deoras, Bernie Wang, and Hao Wang. 2021. Zero-Shot Recommender Systems. *ArXiv* abs/2105.08318 (2021).

[8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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

[10] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv: Learning* (2016).

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015).

[12] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2022).

[13] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018).

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. *2018 IEEE International Conference on Data Mining (ICDM)* (2018), 197–206.

[15] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Annual Meeting of the Association for Computational Linguistics*.

[16] Chenglin Li, Mingjun Zhao, Huanming Zhang, Chenyun Yu, Lei Cheng, Guoqiang Shu, Beibei Kong, and Di Niu. 2021. RecGURU: Adversarial Learning of Generalized User Representations for Cross-Domain Recommendation. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (2021).

[17] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017).

[18] Jiacheng Li, Jingbo Shang, and Julian McAuley. 2022. UCTopic: Unsupervised Contrastive Learning for Phrase Representations and Topic Mining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 6159–6169. https://doi.org/10.18653/v1/2022.acl-long.426

[19] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020).

[20] Jiacheng Li, Tong Zhao, Jin Li, Jim Chan, Christos Faloutsos, George Karypis, Soo-Min Pantel, and Julian McAuley. 2022. Coarse-to-Fine Sparse Sequential Recommendation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022).

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* abs/1907.11692 (2019).

[22] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Conference on Empirical Methods in Natural Language Processing*.

[23] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training.

[24] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv* abs/1910.10683 (2019).

[25] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *The Web Conference*.

[26] Ajit Paul Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Knowledge Discovery and Data Mining*.

[27] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019).

[28] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018).

[29] Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. 2012. Cross-domain collaboration recommendation. In *Knowledge Discovery and Data Mining*.

[30] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. *ArXiv* abs/1706.03762 (2017).

[31] Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. 2020. PTUM: Pre-training User Model from Unlabeled User Behaviors via Self-supervision. *ArXiv* abs/2010.01494 (2020).

[32] Chaojun Xiao, Ruobing Xie, Yuan Yao, Zhiyuan Liu, Maosong Sun, Xu Zhang, and Leyu Lin. 2021. UPRec: User-Aware Pre-training for Recommender Systems. *ArXiv* abs/2102.10989 (2021).

[33] Ruobing Xie, Qi Liu, Liangdong Wang, Shukai Liu, Bo Zhang, and Leyu Lin. 2021. Contrastive Cross-domain Recommendation in Matching. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2021).

[34] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2018. A Simple Convolutional Generative Network for Next Item Recommendation. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (2018).

[35] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Xiangnan He, Joemon M. Jose, Beibei Kong, and Yudong Li. 2020. One Person, One Model, One World: Learning Continual User Representation without Forgetting. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[36] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. *ArXiv* abs/2007.14062 (2020).

[37] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation. In *International Joint Conference on Artificial Intelligence*.

[38] Kun Zhou, Haibo Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji rong Wen. 2020. S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

[39] Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. 2019. DTCDR: A Framework for Dual-Target Cross-Domain Recommendation. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019).

[40] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. 2021. Cross-Domain Recommendation: Challenges, Progress, and Prospects. *ArXiv* abs/2103.01696 (2021).