Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5)

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, Yongfeng Zhang
Department of Computer Science, Rutgers University, NJ 08854, US
{sg1309,shuchang.syt.liu,zuohui.fu,yingqiang.ge,yongfeng.zhang}@rutgers.edu

ABSTRACT
For a long time, different recommendation tasks typically require designing task-specific architectures and training objectives. As a result, it is hard to transfer the learned knowledge and representations from one task to another, thus restricting the generalization ability of existing recommendation approaches, e.g., a sequential recommendation model can hardly be applied or transferred to a review generation method. To deal with such issues, considering that language can describe almost anything and language grounding is a powerful medium to represent various problems or tasks, we present a flexible and unified text-to-text paradigm called “Pretrain, Personalized Prompt, and Predict Paradigm” (P5) for recommendation, which unifies various recommendation tasks in a shared framework. In P5, all data such as user-item interactions, user descriptions, item metadata, and user reviews are converted to a common format — natural language sequences. The rich information from natural language assists P5 to capture deeper semantics for personalization and recommendation. Specifically, P5 learns different tasks with the same language modeling objective during pretraining. Thus, it serves as the foundation model for various downstream recommendation tasks, allows easy integration with other modalities, and enables instruction-based recommendation based on prompts. P5 advances recommender systems from shallow model to deep model to big model, and will revolutionize the technical form of recommender systems towards universal recommendation engine. With adaptive personalized prompt for different users, P5 is able to make predictions in a zero-shot or few-shot manner and largely reduces the necessity for extensive fine-tuning. On several recommendation benchmarks, we conduct experiments to show the effectiveness of P5. To help advance future research on Recommendation as Language Processing (RLP), Personalized Foundation Models (PFM), and Universal Recommendation Engine (URE), we release the source code, dataset, prompts, and pretrained P5 model at https://github.com/jeykigung/P5.

KEYWORDS
Recommender Systems; Natural Language Processing; Multitask Learning; Personalized Prompt; Language Modeling; Unified Model

1 INTRODUCTION
For the past decades, recommender systems have witnessed significant advancements and played an essential role in people’s daily life, helping their micro decisions and fulfilling their demands with outstanding accuracy. In retrospect, we can summarize the development trend of modern recommender systems towards a more comprehensive system that accommodates diverse features and a wide spectrum of application scenarios.

On one hand, feature engineering and learning in recommender systems has evolved greatly from simple to complex. In early ages, recommender systems typically adopt logistic regression or collaborative filtering [25, 35, 50, 52] which utilize user-item interaction records to model users’ behavioral patterns. Later on, the contextual features such as user profile and item metadata are further integrated into the system through more sophisticated models such as factorization machines [48] and GBDT [20]. Recently, deep neural network models [3, 5, 19, 74] facilitate crossing and combination among even more diverse and sophisticated features. As a result, these models gain better representation ability compared with traditional feature engineering based approaches.

On the other hand, more recommendation tasks have emerged. Except for classical rating prediction and direct user-item matching-based recommendation tasks, recent works are broadening the spectrum to new tasks and scenarios such as sequential recommendation [21, 60, 63, 80], conversational recommendation [8, 61, 76], explainable recommendation [17, 31, 62, 70, 75, 77] and so on. While the approaches to the aforementioned recommendation tasks are often proposed separately, there is an evident trend of utilizing multiple recommendation tasks to jointly learn the transferable representations [31, 56, 57, 72]. Although existing recommender systems achieved great success, there is still a considerable gap between current solutions and the foreseeable intersection of the aforementioned trends — a comprehensive recommender system that can accommodate diverse features and different types of tasks. Since recommendation tasks usually share a common user–item pool and have overlapping contextual features, we believe it is promising to merge even more recommendation tasks into a unified framework so that they can implicitly transfer knowledge to benefit each other and enable generalization to other unseen tasks.

Inspired by the recent progress in multitask prompt-based training [1, 51, 67], in this work, we propose a unified “Pretrain, Personalized Prompt & Predict Paradigm” (denoted as P5). We show that P5 is possible to learn multiple recommendation related tasks together through a unified sequence-to-sequence framework by formulating these problems as prompt-based natural language tasks, where user–item information and corresponding features are integrated with personalized prompt templates as model inputs. P5 sheds light...
on a promising technical route for unified and instruction-based recommendation. It has three main advantages:

1) P5 deeply immerses recommendation models into a full language environment, where all recommendation tasks are reformulated to NLP tasks with the help of personalized prompts. Since language grounding is sufficiently flexible and powerful to express various kinds of features in text templates, so there is no need to design feature-specific encoders. As a result, P5 can exploit the abundant semantics and knowledge inside the training corpora;

2) P5 integrates multiple recommendation tasks into a shared text-to-text encoder-decoder architecture and trains them with the same language modeling loss rather than designing task-specific architectures and objective functions. In other words, P5 treats all personalized tasks as a conditional text generation problem;

3) Trained with instruction-based prompts, P5 attains sufficient zero-shot performance when generalizing to novel personalized prompts or unseen items in other domains.

In our experiments, we study how P5 performs compared with task-specific approaches on all five task families as well as evaluating P5’s zero-shot generalization ability. We also conduct several ablation studies to justify the design details of P5 framework. Overall, our main contributions can be outlined as follows:

- To the best of our knowledge, this is the first work to propose a unified “Pretrain, Personalized Prompt & Predict Paradigm” which integrates various recommendation related tasks into a shared conditional language generation framework.
- We create a collection of personalized prompts that cover five different recommendation task families.
- According to the experimental results, P5 achieves promising performances on the five task families when taking seen prompt templates as model inputs.
- P5 shows sufficient zero-shot generalization ability for novel personalized prompts and new items in unseen domains.

2 RELATED WORK

Unified Frameworks. Many prior works have pursued to solve various tasks in a unified model. As early pioneers, T5 [47] and GPT-3 [2] unifies NLP downstream tasks through text-to-text encoder-decoder framework and autoregressive language modeling, respectively. They both allow effective knowledge sharing among different tasks based on a common pretrained language model. Following this trend, recent advances started to focus on unifying large-scale language tasks [1, 51, 67] or cross-modality applications [6, 66, 71] through a shared sequence-to-sequence framework, where different types of tasks and modalities are all expressed in the format of natural language. However, aforementioned methods never consider personalization in their sequence-to-sequence models. Recently, a line of work [56, 57, 72] attempt to learn universal user representations which are easily transferable to downstream tasks. One limitation of these methods is that they still require additional finetuning on downstream datasets. In contrast, our P5 first takes personalization into an encoder-decoder Transformer model that can generalize to a wide spectrum of recommendation related application scenarios – tasks that naturally require personalization. Moreover, with the help of prompt-based pretraining, P5 acquires zero-shot generalization ability when transferring to unseen prompts and items.
Prompt Learning. The success of GPT series especially GPT-3 [2] marked the beginning of prompt’s popularization on NLP tasks. Trained with huge language data from the Web, GPT-3 exhibited the capability of solving NLP tasks when provided a number of input-output examples as exemplar prompts. Besides exemplar prompts, many prompt design methods have proliferated following the "pre-train, prompt, and predict" paradigm [37]. One type of the methods [16, 23, 36, 40, 58] explored prompt search for proper discrete prompts. Meanwhile, another line of work [18, 28, 33, 38, 45, 81] exploited continuous vector embeddings as prompts. Compared with the aforementioned prompt types, instruction-based prompts contain detailed task descriptions and adhere more to the natural language format. Since instruction-based prompts are flexible and close to how humans communicate with each other, several pioneer works [11, 68] claim that learning from crowd-sourced NLP datasets is a promising route for general purpose NLP systems. Recent works such as FLAN [67] and T0 [51] finetuned pretrained language models on large-scale NLP datasets verbalized via human-readable prompts. As a result, such multitask prompt-based tuning brings powerful models that exhibit strong zero-shot ability on unseen tasks. Inspired by the success of these approaches, we create a collection of personalized prompts and then train a sequence-to-sequence model on a variety of recommendation related tasks verbalized according to the constructed personalized prompts.

NLP for Recommendation. Recommendation has been interacting with NLP techniques for a long time. The main work mostly address four lines of research: 1) explainable recommendation [4, 10, 30–32, 75, 77] where NLP models help generating text explanations for a given recommendation; 2) sequential recommendation as language modeling [9, 60, 80] which considers user interaction histories as word token sequences; 3) text feature extraction [69, 74, 79] which aims to extract informative text encodings that can improve the performance of recommendation; and 4) conversational recommendation [8, 12–14, 22, 61, 76] that reasons the intent of users and the performance of recommendation; and additionally offers insights on how addressing four lines of research: 1) explainable recommendation [4, 10, 30–32, 75, 77] where NLP models help generating text explanations for a given recommendation; 2) sequential recommendation as language modeling [9, 60, 80] which considers user interaction histories as word token sequences; 3) text feature extraction [69, 74, 79] which aims to extract informative text encodings that can improve the performance of recommendation; and 4) conversational recommendation [8, 12–14, 22, 61, 76] that reasons the intent of users and gives recommendation in an interactive dialog format. In our work, we explicitly covers the tasks of sequential recommendation and explanation generation, and additionally offers insights on how addressing four lines of research: 1) explainable recommendation [4, 10, 30–32, 75, 77] where NLP models help generating text explanations for a given recommendation; 2) sequential recommendation as language modeling [9, 60, 80] which considers user interaction histories as word token sequences; 3) text feature extraction [69, 74, 79] which aims to extract informative text encodings that can improve the performance of recommendation; and 4) conversational recommendation [8, 12–14, 22, 61, 76] that reasons the intent of users and gives recommendation in an interactive dialog format.

Zero-shot and Cold Start Recommendation. Recommender systems’ performances heavily rely on the available training data, but there are always zero-shot cases where the history records are limited. The evidences of performing well on such startup cases signal a good generalization ability of recommendation models.
As mentioned in [51], a prompt is considered as consisting of an input–target pair to serve as the training data. During pretraining, we mix the input–target pairs from different task families together to serve as the training data. As illustrated in Figure 2, we can simply substitute the fields in braces with the corresponding information in the raw data and thus create training input–target pairs or zero-shot testing personalized prompts. The training data and pre-training tasks will distill the rich semantics from diverse modalities into the user and item tokens for preference understanding and personalization. Note that we divide the raw data into three parts—rating/review/explanation—share the same raw data, while sequential and direct recommendation differ in terms of whether to use interaction history as input information. During pretraining, we mix the input–target pairs from different task families together to serve as the training data. To enhance P5’s robustness and zero-shot generalization, for each raw datum, we only sample a portion of rather than all of the personalized prompts to help P5 discover various aspects about users and items. For example, a user’s preference can be indicated through either an ID number or a description of the user such as name, gender, age, etc. Moreover, the expected model output of a personalized prompt should also vary according to its item field. This implies the change of user’s preferences towards different items. Such item fields can be represented by either item ID numbers or item metadata that contains detailed descriptions.

We designed basic P5 personalized prompt collection for each task family. For rating prediction task family, we divide the prompts into three categories: 1) Given the information about a user and an item, directly predict the rating score ranging from 1 to 5; 2) Predict whether a user will rate an item given a score. The expected output is yes or no; 3) Predict if a user likes or dislikes an item. Here we consider a star rating equal to or greater than 4 to be a like preference of the user, whereas lower scores indicate a dislike preference. For sequential recommendation task family, we create three types of prompts: 1) Directly predict the next item based on user interaction history; 2) Given user interaction history, choose the possible next item from a candidate list, where only one item is positive; 3) Based on user interaction history, predict whether a given item will be interacted next by the user. For explanation task family, we ask P5 model to generate a textual explanation to justify a user’s preference towards a given item. There are two prompt categories in this task family: 1) Directly generate an explanation sentence with user/item information; 2) Generate explanation based on a feature word as hint [31]. For each category, there could be other auxiliary information included such as the review headline and the star rating. For review related task family, we create two types of prompts: 1) Summarize review comment to a shorter review title; 2) Predict the corresponding rating score based on the given review comment. For direct recommendation, we also create two types of prompts: 1) Predict whether to recommend an item to a user, the answer should be yes or no; 2) Select the most suitable item from a list of candidate items to recommend to the user. We provide some example prompts in Figure 2, and the complete collection of personalized prompts are provided in the Appendix.

With the prompts, we can directly build input–target pairs from raw data. As illustrated in Figure 2, we can simply substitute the fields in braces with the corresponding information in the raw data and thus create training input–target pairs or zero-shot testing personalized prompts. The training data and pre-training tasks will distill the rich semantics from diverse modalities into the user and item tokens for preference understanding and personalization. Note that we divide the raw data into three parts—rating/review/explanation—share the same raw data, while sequential and direct recommendation differ in terms of whether to use interaction history as input information. During pretraining, we mix the input–target pairs from different task families together to serve as the training data. To enhance P5’s robustness and zero-shot generalization, for each raw datum, we only sample a portion of rather than all of the personalized prompts in each task family. In sequential and direct recommendation task families, we also randomly select a group of negative items for those prompts that require a candidate list.
4 THE P5 PARADIGM AND MODEL

4.1 The P5 Architecture

The collection of personalized prompts introduced in the previous section makes it convenient to create a large amount of available pretraining data that covers a wide range of recommendation related tasks. Thanks to the prompt templates, all pretraining data shares a unified format of input–target token sequences, which breaks the boundaries among different tasks. We claim that pretraining multiple recommendation tasks under a unified framework of conditional generation can facilitate all involving tasks together. By immersing P5 in the full language environment throughout the pretraining stage, we also expect its zero-shot generalization capability of understanding unseen personalized prompts with detailed item descriptions. That is the reason why P5 is called a unified “Pretrain, Personalized Prompt, and Predict Paradigm”.

In terms of the model architecture, our P5 is established upon a basic encoder–decoder framework. We employ Transformer [65] blocks to build both the encoder and decoder. Suppose the embeddings of an input token sequence is \( x = [x_1, \ldots, x_n] \). As depicted in Figure 3, before feeding the embedding sequence into the bidirectional text encoder \( E(\cdot) \), we add positional encodings \( P \) to the raw embeddings to capture their position information in the sequence. Furthermore, to make P5 aware of the personalized information contained in the input sequence, we also apply whole-word embeddings \( W \) to indicate whether consecutive sub-word tokens are from the same original word. For instance, if we directly represent the item with ID number 7391 as “item_7391”, then the word will be split into 4 separate tokens (i.e., “item”, “_”, “73”, “91”) by SentencePiece tokenizer [54]. With the assistance of the shared whole-word embedding “(w10)”, e.g., in Figure 3, P5 can better recognize the important field with personalized information. Another alternative is to represent each user/item by an independent extra token (e.g., “(item_7391)”). However, this may incur huge amounts of additional tokens when there is a large pool of users and items. Hence, in this paper, we adopt multiple sub-word units to represent a user or item.

Afterwards, the text encoder takes the sum of the aforementioned three embeddings \( e = [e_1, \ldots, e_n] \) and outputs their contextualized representations \( t = [t_1, \ldots, t_n] = E(e) \). The decoder \( D(\cdot) \) then attends to both the previously generated tokens \( y_{<j} \) and the encoder output \( t \) and predicts the probability distribution of future tokens: \( P_\theta(y_j \mid y_{<j}, x) = D(y_{<j}, t) \). During the pretraining stage, P5 learns the model parameters \( \theta \) by minimizing the negative log-likelihood of label tokens \( y \) conditioned on input text \( x \) in an end-to-end manner:

\[
\mathcal{L}^P_\theta = -\sum_{j=1}^{\lceil y \rceil} \log P_\theta(y_j \mid y_{<j}, x)
\]  

(1)

This same objective function is shared by all recommendation tasks under P5. As a result, we unify recommendation tasks with one model, one loss, and one data format.

4.2 Recommendation with Pretrained P5

After pretraining, P5 can directly perform different tasks with either seen or unseen personalized prompts. For rating, explanation, and review tasks, we simply use greedy decoding to generate answers. In contrast, sequential and direct recommendation tasks usually require an item list as target output. In view of this, for sequential recommendation, we apply beam search to generate a list of potential next items and evaluate it under the all-item setting. For direct recommendation, we predict the recommended items from a candidate set \( S = \{s_1, \ldots, s_m\} \), where only one of the \( m \) candidates is positive. Here, we also use beam search to decode a list of potential target items with the highest scores and then conduct evaluations. Both of the above decoding processes can be written as:

\[
C = [C_1, \ldots, C_B] = \text{Beam Search}(D, t, B)
\]

(2)

where \( B \) denotes the beam size and \( C \) is the output item list.

5 EXPERIMENTS

In this section, we evaluate the performance of the proposed P5 approach on real-world data and compare it with various representative methods targeting at different task families. Through the performance comparison and ablation studies, we aim to answer the following research questions regarding our unified “Pretrain, Personalized Prompt, and Predict Paradigm” (P5):

- **RQ1:** How does our unified P5 framework perform compared with task-specific methods on all five task families?
- **RQ2:** Does P5 have enough zero-shot generalization ability when transferring to unseen personalized prompts for either existing or new items?
- **RQ3:** How do scaling factors such as model size, number of task families, and number of prompts affect the performance of P5?
- **RQ4:** Which is a better way to implement personalization in P5: adopting an independent extra token for each user or item (e.g., “(user,23)”) or the default setting, i.e., tokenizing each user or item into multiple sub-word units (e.g., “user”, “_”, “23”)?
- **RQ5:** How long does it take for P5 to conduct pretraining? Is it efficient to make inference with the pretrained P5 model? We provide statistics on training and inference time in the Appendix.

5.1 Experimental Setup

**Datasets.** We conduct extensive experiments over four real-world datasets. The Amazon1 datasets are collected from Amazon.com platform with user ratings and reviews on 29 categories of products. In this paper, we adopt three of them to evaluate our method, namely Sports & Outdoors, Beauty, as well as Toys & Games. Besides, Yelp2 dataset contains a large number of user ratings and reviews for business recommendation. We follow [80] and use transaction records between January 1, 2019 to December 31, 2019. Due to space limit and that the results on Yelp show similar trends with other datasets, we put the experimental results on Yelp dataset in the Appendix. The detailed statistics of these datasets are presented in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sports</th>
<th>Beauty</th>
<th>Toys</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>35,598</td>
<td>22,363</td>
<td>19,412</td>
<td>30,431</td>
</tr>
<tr>
<td>#Items</td>
<td>18,357</td>
<td>12,101</td>
<td>11,924</td>
<td>20,033</td>
</tr>
<tr>
<td>#Reviews</td>
<td>296,337</td>
<td>198,502</td>
<td>167,597</td>
<td>316,354</td>
</tr>
<tr>
<td>#Sparsity (%)</td>
<td>0.0453</td>
<td>0.0734</td>
<td>0.0724</td>
<td>0.0519</td>
</tr>
</tbody>
</table>

1https://nijiaom.github.io/amazon/  
2https://www.yelp.com/dataset
Task splits. For rating, explanation, and review task families, we randomly split each dataset into training (80%), validation (10%) and testing (10%) sets, and ensure that there is at least one instance included in the training set for each user and item. To obtain the ground-truth explanations, following the natural language explanation works [30, 31], we first extract item feature words from the reviews with the help of the Sentires toolkit\(^3\) [77, 78], and then extract the sentences from reviews that comment on one or more item feature words as users’ explanation about their preference. In terms of sequential recommendation task family, for each user interaction sequence, the last item is used as the test data, the item before the last one is used as the validation data, and the remaining data is used for training. To avoid data leakage during pretraining, we follow the training split of sequential recommendation to build the training set for direct recommendation task family.

Implementation Details. Our P5 model utilizes the pretrained T5 checkpoints [47] as backbone. According to the size of T5 backbone, we create two versions of P5, namely P5-small (P5-S) and P5-base (P5-B). For P5-small, there are 6 layers for both encoder and decoder, the model dimensionality is 512 with 8-headed attention, and the number of parameters is 60.75 million. For P5-base, encoder and decoder both have 12 Transformer blocks. The model has an embedding dimensionality of 768 and a 12-headed attention, and the number of parameters is 223.28 million. For tokenization, we use the SentencePiece [54] tokenizer with a vocabulary size of 32,128 for parsing sub-word units. We pretrain P5 for 10 epochs with AdamW optimization [39] on four NVIDIA RTX A5000 GPUs. The batch size is set to 16 for P5-base and 32 for P5-small. We choose \(1 \times 10^{-3}\) as the peak learning rate and set the maximum length of input tokens to 512. The warmup strategy is used to adjust the learning rate during training, the warmup stage is set to be the first 5% of all iterations. When negative sampling is needed for training, we use 1:1 positive vs. negative sampling for both P5 and baselines.

Our default pretrained-predict combination adopts the last prompt in each task family for zero-shot evaluation while all remaining prompts are utilized for multitask prompted pretraining. For rating prediction, we use Gaussian sampling to convert the original integer scores to float numbers rounded to 1 decimal place. In this way, we can avoid overfitting the limited score types. After this change, we increase the number of score classes from 5 to 41. For sequential recommendation, we set the beam size \(B\) to 20. For direct recommendation, the beam size is also 20 and the candidate pool contains 100 items, which consist of one ground-truth item and 99 sampled negative ones that the user has not interacted with.

Metrics. For rating prediction, we adopt Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). For sequential recommendation and direct recommendation tasks, we employ top-k Hit Ratio (HR@k) and Normalized Discounted Cumulative Gain (NDCG@k) to evaluate the performance and report HR@1, 5, 10 and NGCG@5, 10. For explanation generation and review summarization, we evaluate different methods with BLEU-4, as well as ROUGE-1, ROUGE-2, and ROUGE-L. RMSE and MAE are “the lower, the better”, while all other metrics are “the higher, the better”. For all tables in the following, **bold** numbers refer to the best performance, while *underlined* numbers indicate the second best performance.

### 5.2 Baselines for Multiple Tasks
To demonstrate P5’s competence on a wide range of recommendation related tasks, we gather a collection of representative approaches for difference task families.

**Rating Prediction and Direct Recommendation.** These tasks take the user–item rating/interaction data, but no content or side information is provided. We aim to justify whether the models are able to provide accurate rating prediction or recommendation lists that align with the user preferences. We use MF [25] and MLP [5] under mean square root loss as rating prediction baselines. For direct recommendation, we use BPR-MF [49], BPR-MLP [5], and a state-of-the-art contrastive learning-based collaborative filtering model SimpleX [43] as baselines.

**Sequential Recommendation.** We adopt several representative sequential recommendation approaches as our baselines. Caser [63] treats sequential recommendation as a Markov Chain and employs convolutional neural networks to model user interests. HGN [41] adopts a hierarchical gating networks to learn user behaviors from the perspectives of both long and short terms. GRU4Rec [21] is originally proposed for session-based recommendation. It utilizes GRU [7] to model the user click history sequence. BERT4Rec [60] mimics the BERT-style masked language modeling and learns a bidirectional representation for sequential recommendation. FDSA [73] focuses on the feature transition patterns by modeling feature sequence with a self-attention module. SASRec [24] adopts self-attention mechanism in a sequential recommendation model, which reconciles the properties of Markov Chains and RNN-based approaches. S\(^5\)-Rec [80] leverages self-supervised objectives to help sequential recommendation model better discover the correlations among different items and their attributes. We use the implementation of S\(^5\)-Rec and its baselines for comparison\(^4\).

**Explanation Generation.** For performance comparison, we consider several baselines with regard to the task of explanation generation. Attn2Seq [10] learns to encode attributes into vectors, and then invokes an attention mechanism to generate reviews conditioned on the attribute vector. NRT [32] utilizes GRU [7] to generate explanations based on user and item IDs. PETER [31] is a simple and effective framework that attempts to utilize user and item IDs to generate explanations. It is built upon a modified attention mask of the Transformer architecture. There is also a variant PETER+, which takes a hint feature word to assist the explanation generation.

**Review Related.** For review summarization, we adopt pretrained T0 [51] and GPT-2 [46] checkpoints hosted by Hugging Face\(^5\) as baselines. For review preference prediction, we only use T0 to make comparisons because GPT-2 cannot perform this task.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Sports</th>
<th>Beauty</th>
<th>Toys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>MF</td>
<td>1.0234</td>
<td>0.7935</td>
<td>1.1973</td>
</tr>
<tr>
<td>MLP</td>
<td>1.1277</td>
<td>0.7626</td>
<td>1.3078</td>
</tr>
<tr>
<td>P5-S (1-6)</td>
<td>1.0594</td>
<td>0.6639</td>
<td>1.3128</td>
</tr>
<tr>
<td>P5-B (1-6)</td>
<td>1.0357</td>
<td>0.6813</td>
<td>1.2843</td>
</tr>
<tr>
<td>P5-S (1-10)</td>
<td>1.0522</td>
<td>0.6698</td>
<td>1.2789</td>
</tr>
<tr>
<td>P5-B (1-10)</td>
<td>1.0292</td>
<td>0.6664</td>
<td>1.2870</td>
</tr>
</tbody>
</table>

---

3https://github.com/evison/Sentires

4https://github.com/RUCAIBox/CIRM2020-S5Rec

5https://huggingface.co/
5.3 Performance Comparison on Different Task Families (RQ1)

In this section, we pretrain P5 with prompts from all five task families to verify its multitask learning ability. According to the default pretrain–predict task combination, we leave Prompt 1-10, Prompt 2-13, Prompt 3-12, Prompt 4-4, and Prompt 5-8 for zero-shot evaluation and pretrain P5 with the remaining personalized prompts. The performances of P5 and relevant baselines on the five task families are presented in Table 2 to Table 7. For each task family, we choose one or more seen prompts as supplement to the aforementioned zero-shot unseen prompts to perform evaluations.

5.3.1 Rating Prediction. Prompt 1-6 and Prompt 1-10 are used for evaluating P5’s performance on rating prediction. The performance comparison is presented in Table 2. We can see that when testing with seen Prompt 1-6, P5-B gets better MAE and slightly higher RMSE on all three datasets compared with MF. When testing with unseen Prompt 1-10, P5-B can achieve similar performance as Prompt 1-6. Moreover, P5-S usually has better MAE but higher RMSE. It seems that P5 is overfitting these data since the task complexity of rating prediction is relatively lower than other recommendation tasks. Overall, these results show that it is feasible to perform rating prediction on a conditional text generation framework.

5.3.2 Sequential Recommendation. As illustrated in Table 3, Prompt 2-3 and Prompt 2-13 are employed for the evaluation of sequential recommendation under all-item setting, i.e., using all items as candidates rather than sampling 100 or 1,000 items for ranking. From the table, we can see that P5-B surpasses all competitive baselines with a relatively large gap on both seen (Prompt 2-3) and unseen (Prompt 2-13) prompts. On Toys, P5-S can get even better performance than P5-B. While on Beauty and Sports, P5-B achieves the advantage over P5-S. The results show that the P5 architecture is effective in modeling the user interaction history and conducting next item prediction with the help of beam search.

5.3.3 Explanation Generation. In Table 4, Prompt 3-9 and Prompt 3-12 are used to evaluate P5’s performance on explanation generation. In most cases, Prompt 3-9 & 3-12 are used to evaluate P5’s performance on explanation generation under feature-based setup, while Prompt 3-3 is used for direct explanation generation without providing a hint word. We can see that for both seen (Prompts 3-3 & 3-9) and unseen (Prompt 3-12) prompts, P5 achieves both the best and the second best performance on most cases. For feature-based prompts (Prompts 3-9 & 3-12), P5-B outperforms PETER+ by a large margin on all domains.

5.3.4 Review Related. We take Prompts 4-2 and 4-4 to compare P5’s performance with T0 on review preference prediction, as shown in Table 5. We can see that P5-B surpasses all competitive baselines with a relatively large gap on both seen and unseen prompts. On Toys, P5-S can get even better performance than P5-B. While on Beauty and Sports, P5-B achieves the advantage over P5-S. The results show that the P5 architecture is effective in modeling the user interaction history and conducting next item prediction with the help of beam search.
MAE on \textit{Beauty} and \textit{Toys}, while \textit{P5-B} shows better performance on \textit{Sports}. Additionally, we take Prompt 4-1 to evaluate \textit{P5}'s ability on review summarization, as shown in Table 6. For this task, \textit{P5-S} clearly outperforms \textit{T0} and \textit{GPT-2} except for \textit{BLUE4} score on \textit{Sports}. It is worth noting that \textit{GPT-2} and \textit{T0} has 1.5B and 11B parameters, respectively. This shows that \textit{P5} can achieve better performance than these competitive baselines with a much smaller model size.

5.3.5 \textbf{Direct Recommendation}. Finally, Prompts 5-1, 5-4, 5-5 and 5-8 are applied to evaluate the direct recommendation task under the 1-out-of-100 evaluation setting. For binary question prompts (5-1 & 5-4), which are discriminative prompts, we use the softmax generation probability of “yes” to rank the candidate items. For open question prompts (5-5 & 5-8), which are generative prompts, we use beam-search (Eq.(2)) to generate the top-\(k\) list. The results are presented in Table 7. From the table, we can see that \textit{P5-B} and \textit{P5-S} have great advantages over \textit{BPR-MF} and \textit{BPR-MLP} on all three datasets. Comparing with \textit{SimpleX}, we can see that \textit{P5} works especially well on top-1 item ranking, which is more than two times better than \textit{SimpleX} on \textit{HR}@1. Besides, \textit{P5} also achieves the best result on most of the other metrics. The success of \textit{P5} on direct recommendation shows the competence of the sequence-to-sequence generation framework in recommendation domain.

5.4 \textbf{Zero-shot Generalization to Unseen Prompts and Items in New Domain (RQ2)}

5.4.1 \textbf{Transfer to Unseen Personalized Prompts}. In this section, we transfer the pretrained \textit{P5} models to the previously held-out prompts during pretraining. These unseen prompts are from the same task families, and the testing items have been seen by \textit{P5} during pretraining at least once. The experimental results are also reported in Table 2 to Table 7. As previously discussed in Section 5.3, \textit{P5} achieves surprisingly good performances on various task families when being challenged by unseen prompts. On some specific datasets, the performances of \textit{P5} on unseen prompts even surpass seen prompts, e.g., \textit{P5-B} gets the best performances under Prompt 2-13 on \textit{Sports} and under Prompt 3-12 on \textit{Beauty}. These results show that multitask prompted pretraining empowers \textit{P5} enough robustness to understand unseen prompts with wording variations.

5.4.2 \textbf{Transfer to Items in New Domain}. Next, we increase the difficulty level of zero-shot transfer. We collect a group of 741 users that exist in all the three domains with their interaction and review histories in other domains. The detailed statistics of these domain transfer evaluation sets are illustrated in Table 8. We then challenge \textit{P5-B} pretrained on one domain with unseen prompts from the Task Family Z, whose item fields are filled with the information from a new product domain. For example, we ask the \textit{P5} model pretrained on the \textit{Toys} domain about an existing user’s preference towards an item in the \textit{Beauty} domain. The full results on all six directions are reported in Table 9. From the table, we notice \textit{P5} still maintains sufficient performances for rating prediction (Prompts Z-2 & Z-3), like/dislike prediction (Prompts Z-1 & Z-4), as well as explanation generation with feature word (Prompt Z-6). In contrast, direct explanation generation without feature word (Prompts Z-5 & Z-7) is very difficult for \textit{P5} because it lacks awareness of relevant knowledge in the new domain. In Figure 4, we provide some example explanations generated by \textit{P5-B} under the setup of zero-shot domain transfer (Prompt Z-6). We can see that \textit{P5} is able to catch different users’ rating preferences and hint feature words, then integrate them with the knowledge learned from previous domain to generate plausible explanations.

Table 6: Performance comparison on review summarization.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Sports</th>
<th>Beauty</th>
<th>Toys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLUE4</td>
<td>ROUGE1</td>
<td>ROUGE2</td>
</tr>
<tr>
<td>T0 (4-1)</td>
<td>8.0881</td>
<td>43.4226</td>
<td>13.9629</td>
</tr>
<tr>
<td>GPT-2 (4-1)</td>
<td>2.6586</td>
<td>43.6313</td>
<td>15.2021</td>
</tr>
<tr>
<td>P5-S (4-1)</td>
<td>7.1781</td>
<td>44.6943</td>
<td>20.0491</td>
</tr>
<tr>
<td>P5-B (4-1)</td>
<td>7.5727</td>
<td>47.7859</td>
<td>21.6872</td>
</tr>
</tbody>
</table>

Table 7: Performance comparison on direct recommendation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Sports</th>
<th>Beauty</th>
<th>Toys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HR@1</td>
<td>HR@5</td>
<td>NDCG@5</td>
</tr>
<tr>
<td>BPR-MF</td>
<td>0.0314</td>
<td>0.1404</td>
<td>0.0848</td>
</tr>
<tr>
<td>BPR-MLP</td>
<td>0.0351</td>
<td>0.1520</td>
<td>0.0927</td>
</tr>
<tr>
<td>SimpleX</td>
<td>0.0331</td>
<td>0.2362</td>
<td>0.1505</td>
</tr>
<tr>
<td>P5-S (5-1)</td>
<td>0.0638</td>
<td>0.2096</td>
<td>0.1375</td>
</tr>
<tr>
<td>P5-B (5-1)</td>
<td>0.0245</td>
<td>0.0816</td>
<td>0.0529</td>
</tr>
<tr>
<td>P5-S (5-4)</td>
<td>0.0791</td>
<td>0.2244</td>
<td>0.1483</td>
</tr>
<tr>
<td>P5-B (5-4)</td>
<td>0.0299</td>
<td>0.1026</td>
<td>0.0665</td>
</tr>
</tbody>
</table>

Table 8: Statistics on domain transfer evaluation sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sports</th>
<th>Beauty</th>
<th>Toys</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>290</td>
<td>439</td>
<td>487</td>
</tr>
<tr>
<td>#Items</td>
<td>381</td>
<td>586</td>
<td>886</td>
</tr>
<tr>
<td>#Reviews</td>
<td>478</td>
<td>1,237</td>
<td>1,183</td>
</tr>
</tbody>
</table>
5.5 Ablation on Model Size (RQ3)

In this section, we will discuss the influence of model size on the performance of P5 on different recommendation tasks. Here, we train two size variants of P5, namely P5-small and P5-base. The parameter numbers of these two P5 models are 60.75M and 223.28M, respectively. From Table 2 to Table 7, we can see that although P5-S is only 1/4 of the size of P5-B, P5-S can beats P5-B on a series of tasks and datasets. For example, P5-S achieves better sequential recommendation, review preference prediction, and direct recommendation (Prompts 5-5 & 5-8) performances than P5-B on Toys. In contrast, P5-B shows advantages on sequential recommendation and review preference prediction tasks for Sports. Since Sports contains more users, items and reviews and has a lower sparsity, it requires a model with higher capacity to discover latent correlation among different personalized factors. The findings indicate that larger P5 models may be needed when the dataset is large, while for smaller datasets, smaller P5 models could be enough. As a result, we should decide an appropriate model size that matches the scale of the training data.

5.6 Ablation on Task Scaling (RQ3)

Moreover, we explore whether multitask prompted pretraining is superior than pretraining on each task family alone. We pretrain P5-small on Beauty dataset with prompts from every single task family, resulting in five models – P5-S1, P5-S2, P5-S3, P5-S4, and P5-S5. We then compare P5-S on various recommendation tasks with the corresponding single task P5 model. The performance comparison between P5-S and P5-SN (N ∈ [1, 2, 3, 4, 5]) is illustrated in Figure 5. As shown in the figure, P5-S achieves comparable or better performance than P5-SN on rating prediction, sequential recommendation and direct recommendation tasks, while on text generation tasks such as explanation generation (Prompts 3-9 & 3-12) and review summarization (Prompt 4-1), P5-SN is better than P5-S. This indicates that multitask modeling (P5-S) seeks a good balance among tasks and improves recommendation performance by leveraging the power of language understanding. Besides, both P5-S and P5-SN perform better than or comparable with state-of-the-art baselines on all tasks, as shown in Table 2 through Table 7, which demonstrates the power of P5 for recommendation.
we need so many prompts during pretraining to enable P5’s zero-shot generalization ability? In this section, we explore to reduce the number of pretraining prompts and then make comparisons with the P5 model pretrained under default setup. To this end, we choose a collection of pretraining prompts that has the minimum number of prompts to cover all important personalized fields. Specifically, this combination contains the following 18 personalized prompts: [1-3, 1-4, 1-8, 1-9, 2-1, 2-3, 2-8, 2-11, 3-2, 3-3, 3-6, 3-9, 4-1, 4-2, 4-3, 5-2, 5-5, 5-7]. Similar to the default pretrain–predict combination, the last prompt in each task family is for zero-shot evaluation. We name this prompt scaling variant of P5-small as P5-PS and then pretrain P5-PS on Beauty dataset. The performance comparison between P5-S and P5-PS is also presented in Figure 5. From the figure, we can observe that P5-S beats P5-PS on most tasks except for some generation tasks (i.e., Prompts 3-3, 3-9 & 4-1). Interestingly, P5-S outperforms P5-PS on Prompt 3-12 – a zero-shot explanation generation task. In fact, P5-S also shows its superiority on other zero-shot tasks such as Prompts 1-10, 2-13, and 5-8. Overall, we can find that larger number of high quality personalized prompts can generally help P5 achieve better performances on various recommendation tasks especially zero-shot tasks with unseen prompts.

5.8 How to Implement Personalization (RQ4)

In this section, we discuss different strategies to implement personalization in P5. The default practice is using SentencePiece tokenizer to split personalized fields into multiple sub-word units and meanwhile using whole-word embedding to preserve the field information (Figure 3). A straightforward alternative is creating an independent extra token for each user and item. Here we name this P5-small variant as P5-I and also pretrain it on Beauty dataset. While the former utilizes collaborative learning to implicitly optimize the latent correlations among different sub-word tokens, the latter learns a unique personalized representation for every extra token. The performance comparison between P5-S and P5-I is shown in Figure 6. We can see that P5-I achieves similar performances as P5-S on regression tasks (Prompts 1-6 & 1-10 for rating prediction, Prompts 4-2 & 4-4 for review-based rating regression) and review summarization tasks (Prompt 4-1). Also, P5-I is slightly better than P5-S on explanation generation tasks (Prompts 3-3, 3-9 & 3-12). However, P5-I significantly underperforms P5-S by a large margin on both sequential and direct recommendation tasks (all prompts in Figure 6 (c) & (d)). The reason behind P5-I’s lower performance lies in that the newly introduced huge number of extra tokens and embeddings cannot be well trained compared with the original sub-word units initialized from T5. This shows that our default setting can achieve better recommendation and overall performances with the help of collaborative learning while keeping a small and constant amount of learnable tokens.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we present P5 which unifies different recommendation tasks into a shared language modeling and natural language generation framework. By designing a collection of personalized prompts covering five recommendation task families, we transfer raw data such as the user-item interactions, user descriptions, item metadata, and user reviews to the same format – input-target text pairs. We then pretrain P5 in a full language environment to help it discover deeper semantics for various recommendation tasks. According to our experiments, P5 can beat or achieve similar performance with several representative approaches on all five task families. Moreover, P5 shows the generalization ability on performing zero-shot transfer to new items, new domains, and new personalized prompts. In the future, we will continue exploring to further enlarge the model size of P5 and employ more powerful base models such as GPT-3, OPT, and BLOOM. Besides, P5 is a very flexible paradigm and it is promising to further extend P5 to diverse modalities and more tasks such as conversational recommendation, comparative recommendation, cross-platform recommendation, or even various search tasks by incorporating user queries into P5. Finally, in this work, we designed explicit prompts since they are intuitive, flexible, and close to the natural way of how humans communicate with each other, which enables instruction-based recommendation, while in the future, we will also investigate prompt search and/or latent prompt techniques to achieve instruction prompts or leverage retrieval-enhanced generation to further boost P5’s performance on downstream tasks.
ACKNOWLEDGMENT

This work was supported in part by NSF IIS 1910154, 2007907, and 2046457. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES


[10] Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In EACL.


[27] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In EMNLP.


APPENDIX

In this appendix, we first provide additional experimental results on Yelp dataset in Section A. Then we collect and report statistics on the training and inference time of P5 variants in Section B. Since personalized prompts constitute a very important part of our work, we thus offer the full list of the personalized prompts for Amazon datasets in Section C and Yelp dataset in Section D, respectively.

A EXPERIMENTAL RESULTS ON YELP DATASET

For Yelp dataset, we also follow the default pretrain–predict task combination setup. Based on the personalized prompts presented in Section D, we leave Prompt 1-10, Prompt 2-13, Prompt 3-10, Prompt 4-3, and Prompt 5-8 for zero-shot evaluation and pretrain a P5-S model with the remaining prompts. We adopt the same baselines as described in Section 5.2 for performance comparison. Again, for each task family, we choose one or more seen prompts as supplement to the aforementioned zero-shot unseen prompts to perform evaluations. The performances of P5-S and relevant baseline models are shown in Table 10 to Table 14. As indicated in these experimental results, P5 also shows great capability on Yelp, especially for the sequential recommendation and explanation generation tasks.

B STATISTICS ON TRAINING & INFERENCE TIME (RQ5)

In this section, we provide statistics on the training and inference time of P5 models, we collect the running time on the A5000 GPUs. From our records, P5-S spent 6.7 hours to finish training, but the inference is very fast. It is also the larger the beam size $\alpha$ in Eq. (2). Since we need to generate a list of recommended items, other tasks. This can be ascribed to the beam search step described in Eq. (2). Since we need to generate a list of recommended items, the larger the beam size $\beta$ is, the longer the decoding will take. Besides, we can also observe that direct recommendation typically takes longer than sequential recommendation. The reason is that 100 candidates are included in the input of direct recommendation prompts, which usually have more tokens than that of sequential recommendation prompts. Overall, even though the pretraining of P5 takes hours to finish, but the inference is very fast. It is also promising to further reduce the training and inference time with the help of efficient Transformer techniques.

Table 10: Performance comparison on rating prediction.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
</tr>
<tr>
<td>MF</td>
<td>1.2645</td>
</tr>
<tr>
<td>MLP</td>
<td>1.2951</td>
</tr>
<tr>
<td>P5-S (1-6)</td>
<td>1.4868</td>
</tr>
<tr>
<td>P5-S (1-10)</td>
<td>1.4685</td>
</tr>
</tbody>
</table>

Table 11: Performance on sequential recommendation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HR@5</td>
</tr>
<tr>
<td>Caser</td>
<td>0.0151</td>
</tr>
<tr>
<td>HGN</td>
<td>0.0186</td>
</tr>
<tr>
<td>GRU4Rec</td>
<td>0.0152</td>
</tr>
<tr>
<td>BERT4Rec</td>
<td>0.0051</td>
</tr>
<tr>
<td>FDSA</td>
<td>0.0158</td>
</tr>
<tr>
<td>SASRec</td>
<td>0.0162</td>
</tr>
<tr>
<td>S$^3$Rec</td>
<td>0.0201</td>
</tr>
<tr>
<td>P5-S (2-3)</td>
<td>0.0568</td>
</tr>
<tr>
<td>P5-S (2-13)</td>
<td><strong>0.0574</strong></td>
</tr>
</tbody>
</table>

Table 12: Performance on explanation generation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLUE4</td>
</tr>
<tr>
<td>Attn2Seq</td>
<td>0.8031</td>
</tr>
<tr>
<td>NRT</td>
<td>0.8128</td>
</tr>
<tr>
<td>PETER</td>
<td>0.5938</td>
</tr>
<tr>
<td>P5-S (3-2)</td>
<td><strong>3.4515</strong></td>
</tr>
<tr>
<td>PETER+</td>
<td>3.2827</td>
</tr>
<tr>
<td>P5-S (3-7)</td>
<td>9.8952</td>
</tr>
<tr>
<td>P5-S (3-10)</td>
<td>9.9617</td>
</tr>
</tbody>
</table>

Table 13: Performance on review preference prediction.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
</tr>
<tr>
<td>T0 (4-2)</td>
<td>0.5383</td>
</tr>
<tr>
<td>T0 (4-3)</td>
<td><strong>0.5359</strong></td>
</tr>
<tr>
<td>P5-S (4-2)</td>
<td>0.6301</td>
</tr>
<tr>
<td>P5-S (4-3)</td>
<td>0.6350</td>
</tr>
</tbody>
</table>

Table 14: Performance on direct recommendation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HR@1</td>
</tr>
<tr>
<td>BPR-MF</td>
<td><strong>0.0813</strong></td>
</tr>
<tr>
<td>BPR-MLP</td>
<td>0.0489</td>
</tr>
<tr>
<td>SimpleX</td>
<td>0.0569</td>
</tr>
<tr>
<td>P5-S (5-1)</td>
<td>0.0603</td>
</tr>
<tr>
<td>P5-S (5-4)</td>
<td>0.0567</td>
</tr>
<tr>
<td>P5-S (5-5)</td>
<td>0.0696</td>
</tr>
<tr>
<td>P5-S (5-8)</td>
<td><strong>0.0705</strong></td>
</tr>
</tbody>
</table>

Table 15: Average inference time (in milliseconds) of P5 variants on different tasks on the Beauty dataset.

<table>
<thead>
<tr>
<th>Models</th>
<th>Sequential Direct Rating</th>
<th>Explanation</th>
<th>Summarization Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>per user</td>
<td>per user–item pair</td>
<td>per review</td>
</tr>
<tr>
<td>(2-13)</td>
<td>(5-8)</td>
<td>(1-10)</td>
<td>(3-12)</td>
</tr>
<tr>
<td>(1-10)</td>
<td>(4-1)</td>
<td>(4-4)</td>
<td></td>
</tr>
<tr>
<td>P5-S</td>
<td>53.96</td>
<td>57.40</td>
<td>4.93</td>
</tr>
<tr>
<td>P5-B</td>
<td>78.19</td>
<td>98.80</td>
<td>13.40</td>
</tr>
</tbody>
</table>
C FULL LIST OF PERSONALIZED PROMPTS FOR AMAZON DATASETS

C.1 Task Family 1: Rating Prediction

Prompt ID: 1-1
Input template: Which star rating will user_((user_id)) give item_((item_id))? (1 being lowest and 5 being highest)
Target template: ((star_rating))

Prompt ID: 1-2
Input template: How will user_((user_id)) rate this product: ((item_title))? (1 being lowest and 5 being highest)
Target template: ((star_rating))

Prompt ID: 1-3
Input template: Will user_((user_id)) give item_((item_id)) a ((star_rating))-star rating? (1 being lowest and 5 being highest)
Target template: ((answer_choices[label])) (yes/no)

Prompt ID: 1-4
Input template: Does user_((user_id)) like or dislike item_((item_id))?
Target template: ((answer_choices[label])) (like/dislike) – like (4,5) / dislike (1,2,3)

Prompt ID: 1-5
Input template: Predict the user_((user_id)) 's preference on item_((item_id)) ((item_title)) -1 \n -2 \n -3 \n -4 \n -5
Target template: ((star_rating))

Prompt ID: 1-6
Input template: What star rating do you think ((user_desc)) will give item_((item_id))? (1 being lowest and 5 being highest)
Target template: ((star_rating))

Prompt ID: 1-7
Input template: How will ((user_desc)) rate this product: ((item_title))? (1 being lowest and 5 being highest)
Target template: ((star_rating))

Prompt ID: 1-8
Input template: Will ((user_desc)) give a ((star_rating))-star rating for ((item_title))? (1 being lowest and 5 being highest)
Target template: ((answer_choices[label])) (yes/no)

Prompt ID: 1-9
Input template: Does ((user_desc)) like or dislike ((item_title))?
Target template: ((answer_choices[label])) (like/dislike) – like (4,5) / dislike (1,2,3)

Prompt ID: 1-10
Input template: Predict ((user_desc)) 's preference towards ((item_title)) (1 being lowest and 5 being highest)
Target template: ((star_rating))

C.2 Task Family 2: Sequential Recommendation

Prompt ID: 2-1
Input template: Given the following purchase history of user_((user_id)): ((purchase_history)) predict next possible item to be purchased by the user?
Target template: ((next_item))

Prompt ID: 2-2
Input template: I find the purchase history list of user_((user_id)): ((purchase_history)) I wonder which is the next item to recommend to the user. Can you help me decide?
Target template: ((next_item))

Prompt ID: 2-3
Input template: Here is the purchase history list of user_((user_id)): ((purchase_history)) try to recommend next item to the user
Target template: ((next_item))

Prompt ID: 2-4
Input template: Given the following purchase history of ((user_desc)): ((purchase_history)) predict next possible item for the user
Target template: ((next_item))

Prompt ID: 2-5
Input template: Based on the purchase history of ((user_desc)): ((purchase_history)) Can you decide the next item likely to be purchased by the user?
Target template: ((next_item))

Prompt ID: 2-6
Input template: Here is the purchase history of ((user_desc)): ((purchase_history)) What to recommend next for the user?
Target template: ((next_item))

Prompt ID: 2-7
Input template: Here is the purchase history of user_((user_id)): ((purchase_history)) Select the next possible item likely to be purchased by the user from the following candidates: ((candidate_items))
Target template: ((next_item))

Prompt ID: 2-8
Input template: Given the following purchase history of ((user_desc)): ((purchase_history)) What to recommend next for the user? Select one from the following items: ((candidate_items))
Target template: ((next_item))

Prompt ID: 2-9
Input template: Based on the purchase history of user_((user_id)): ((purchase_history)) Choose the next possible purchased item from the following candidates: ((candidate_items))
Target template: ((next_item))

Prompt ID: 2-10
Input template: I find the purchase history list of ((user_desc)): ((purchase_history)) I wonder which is the next item to recommend to the user. Try to select one from the following candidates: ((candidate_items))
Target template: ((next_item))
Prompt ID: 2-11
Input template: User_{((user_id))} has the following purchase history: 
((purchase_history))
Does the user likely to buy ((candidate_item)) next?
Target template: [((answer_choices[label]))] (yes/no)

Prompt ID: 2-12
Input template: According to ((user_desc)) 's purchase history list: 
((purchase_history))
Predict whether the user will purchase ((candidate_item)) next?
Target template: [((answer_choices[label]))] (yes/no)

Prompt ID: 2-13
Input template: According to the purchase history of ((user_desc)):
((purchase_history))
Can you recommend the next possible item to the user?
Target template: [((next_item))]

C.3 Task Family 3: Explanation Generation

Prompt ID: 3-1
Input template: Generate an explanation for user_{((user_id))} about this product:
((item_title))
Target template: [((explanation))]

Prompt ID: 3-2
Input template: Given the following review headline
((review_headline))
can you help generate an explanation of user_{((user_id))} for item_{((item_id))}?
Target template: [((explanation))]

Prompt ID: 3-3
Input template: Help user_{((user_id))} generate a ((star_rating))-star explanation about this product:
((item_title))
Target template: [((explanation))]

Prompt ID: 3-4
Input template: Generate an explanation for ((user_desc)) about this product:
((item_title))
Target template: [((explanation))]

Prompt ID: 3-5
Input template: Based on the following review headline:
((review_headline))
Generate ((user_desc)) 's purchase explanation about ((item_title))
Target template: [((explanation))]

Prompt ID: 3-6
Input template: Help ((user_desc)) generate a ((star_rating))-star explanation for item_{((item_id))}
Target template: [((explanation))]

Prompt ID: 3-7
Input template: Predict the star rating, then use ((feature_word)) as feature word to generate user_{((user_id))} 's purchase explanation for item_{((item_id))}
Target template: [((star_rating)), [((explanation))]

Prompt ID: 3-8
Input template: What score will ((user_desc)) rate item_{((item_id))}?
Then give an explanation for the rating score. (1 being lowest and 5 being highest)
Target template: [((star_rating)), [((explanation))]

C.4 Task Family 4: Review Related

Prompt ID: 4-1
Input template: Write a short sentence to summarize the following product review from user_{((user_id))}:
((review))
Target template: [((summary))]

Prompt ID: 4-2
Input template: Given the following review written by user_{((user_id))}:
((review))
Can you predict the associated star rating (1 being lowest and 5 being highest)?
Target template: [((star_rating))]

Prompt ID: 4-3
Input template: Give a short sentence describing the following product review from ((user_desc)):
((review))
Target template: [((summary))]

Prompt ID: 4-4
Input template: According to the following review written by ((user_desc)):
((review))
Predict the associated star rating (1 being lowest and 5 being highest)
Target template: [((star_rating))]

C.5 Task Family 5: Direct Recommendation

Prompt ID: 5-1
Input template: Will user_{((user_id))} likely to interact with item_{((item_id))}?
Target template: [((answer_choices[label]))] (yes/no)

Prompt ID: 5-2
Input template: Shall we recommend item_{((item_id))} to ((user_desc))?
Target template: [((answer_choices[label]))] (yes/no)
C.6 Task Family Z: Zero-Shot Generalization

Prompt ID: Z-1
Input template: Given the facts about the new product, do you think user_((user_id)) will like or dislike it? title: ((item_title)) brand: ((brand)) price: ((price))
Target template: ((answer_choices[0] \{like/dislike\} – like \{4,5\} / dislike \{1,2,3\})) (yes/no)

Prompt ID: Z-2
Input template: Here are the details about a new product: title: ((item_title)) brand: ((brand)) price: ((price)). What star will user_((user_id)) probably rate the product?
-1 \-2 \-3 \-4 \-5
Target template: ((star_rating))

Prompt ID: Z-3
Input template: Predict user_((user_id))’s preference about the new product \{1 being lowest and 5 being highest\}: title: ((item_title)) price: ((price)) brand: ((brand))
Target template: ((star_rating))

Prompt ID: Z-4
Input template: Will ((user_desc)) like or dislike the following product? title: ((item_title)) price: ((price)) brand: ((brand))
Target template: ((answer_choices[0] \{like/dislike\} – like \{4,5\} / dislike \{1,2,3\})) (yes/no)

Prompt ID: Z-5
Input template: Generate a possible explanation for ((user_desc))’s preference about the following product: title: ((item_title)) brand: ((brand)) price: ((price))
Target template: ((explanation))

Prompt ID: Z-6
Input template: Based on the word ((feature_word)), help user_((user_id)) write a ((star_rating))-star explanation for this new product: title: ((item_title)) price: ((price)) brand: ((brand))
Target template: ((explanation))

Prompt ID: Z-7
Input template: For the new product ((item_title)), we would like to know whether ((user_desc)) will love it. If you think the user will love it, please help explain why.
Target template: ((explanation))

D FULL LIST OF PERSONALIZED PROMPTS FOR YELP DATASET

D.1 Task Family 1: Rating Prediction

Prompt ID: 1-1
Input template: Which star rating will user_((user_id)) give item_((item_id))? \{1 being lowest and 5 being highest\}
Target template: ((star_rating))

Prompt ID: 1-2
Input template: How will user_((user_id)) rate this business: ((item_title))? \{1 being lowest and 5 being highest\}
Target template: ((star_rating))

Prompt ID: 1-3
Input template: Will user_((user_id)) give item_((item_id)) a ((star_rating))-star rating? \{1 being lowest and 5 being highest\}
Target template: ((answer_choices[0] \{like/dislike\} – like \{4,5\} / dislike \{1,2,3\})) (yes/no)

Prompt ID: 1-4
Input template: Does user_((user_id)) like or dislike item_((item_id))?
Target template: ((answer_choices[0] \{like/dislike\} – like \{4,5\} / dislike \{1,2,3\})) (yes/no)

Prompt ID: 1-5
Input template: Predict the user_((user_id))’s preference on item_((item_id)) ((item_title))
-1 \-2 \-3 \-4 \-5
Target template: ((star_rating))

Prompt ID: 1-6
Input template: What star rating do you think ((user_desc)) will give item_((item_id))? \{1 being lowest and 5 being highest\}
Target template: ((star_rating))

Prompt ID: 1-7
Input template: How will ((user_desc)) rate this business: ((item_title))? \{1 being lowest and 5 being highest\}
Target template: ((star_rating))

Prompt ID: 1-8
Input template: Will ((user_desc)) give a ((star_rating))-star rating for ((item_title))? \{1 being lowest and 5 being highest\}
Target template: ((answer_choices[0] \{like/dislike\} – like \{4,5\} / dislike \{1,2,3\})) (yes/no)

Prompt ID: 1-9
D.2 Task Family 2: Sequential Recommendation

Prompt ID: 2-1
Input template: Given the following visit history of user, {{visit_history}}
predict next possible business to be visited by the user?
Target template: {{next_item}}

Prompt ID: 2-2
Input template: I wonder which is the next item to recommend to the user. Can you help me decide?
Target template: {{next_item}}

Prompt ID: 2-3
Input template: Try to recommend next item to the user
Target template: {{next_item}}

Prompt ID: 2-4
Input template: Given the following visit history of user, {{visit_history}}
predict next possible business for the user
Target template: {{next_item}}

Prompt ID: 2-5
Input template: Can you decide the next business likely to be visited by the user?
Target template: {{next_item}}

Prompt ID: 2-6
Input template: What to recommend next for the user?
Target template: {{next_item}}

Prompt ID: 2-7
Input template: Select the next possible business likely to be visited by the user from the following candidates:
Target template: {{next_item}}

Prompt ID: 2-8
Input template: What to recommend next for the user? Select one from the following items:
Target template: {{next_item}}

Prompt ID: 2-9
Input template: Based on the visit history of user, {{visit_history}}
Choose the next possible visited business from the following candidates:
Target template: {{next_item}}

D.3 Task Family 3: Explanation Generation

Prompt ID: 3-1
Input template: Generate an explanation for user, {{user_id}} about this business: {{item_title}}
Target template: {{explanation}}

Prompt ID: 3-2
Input template: Help user, {{user_id}} generate a {{rating}}-star explanation about this business: {{item_title}}
Target template: {{explanation}}

Prompt ID: 3-3
Input template: Generate an explanation for user, {{user_id}} about this business: {{item_title}}
Target template: {{explanation}}

Prompt ID: 3-4
Input template: Help generate a {{rating}}-star explanation for item, {{item_id}}
Target template: {{explanation}}

Prompt ID: 3-5
Input template: Predict the star rating, then use as feature word to generate user, {{user_id}} 's visitation for item, {{item_id}}
Target template: {{rating}}, {{explanation}}

Prompt ID: 3-6
Input template: What score will user, {{user_id}} rate item, {{item_id}}? Then give an explanation for the rating score. (1 being lowest and 5 being highest)
Target template: {{rating}}, {{explanation}}
Prompt ID: 3-7
Input template: Based on the feature word \{feature_word\}, generate an explanation for user_\{user_id\} about this business: \{item_title\}
Target template: \{explanation\}

Prompt ID: 3-8
Input template: Given the word \{feature_word\}, can you help generate an explanation for \{user_desc\} about the business: \{item_title\}
Target template: \{explanation\}

Prompt ID: 3-9
Input template: Using the word \{feature_word\}, write a \{star_rating\}-star explanation for user_\{user_id\} about item_\{item_id\}
Target template: \{explanation\}

Prompt ID: 3-10
Input template: According to the feature word \{feature_word\}, generate a \{star_rating\}-star explanation for \{user_desc\} about item_\{item_id\}
Target template: \{explanation\}

D.4 Task Family 4: Review Related
Prompt ID: 4-1
Input template: Predict the associated rating score of the review written by user_\{user_id\} (1 being lowest and 5 being highest):
\{review\}
Target template: \{star_rating\}

Prompt ID: 4-2
Input template: Given the following review written by user_\{user_id\}:
\{review\}
Can you predict the associated star rating (1 being lowest and 5 being highest)?
Target template: \{star_rating\}

Prompt ID: 4-3
Input template: According to the following review written by \{user_desc\}:
\{review\}
Predict the associated star rating (1 being lowest and 5 being highest)
Target template: \{star_rating\}

D.5 Task Family 5: Direct Recommendation
Prompt ID: 5-1
Input template: Will user_\{user_id\} likely to interact with item_\{item_id\}?
Target template: \{answer_choices\} (yes/no)

Prompt ID: 5-2
Input template: Shall we recommend item_\{item_id\} to \{user_desc\}?
Target template: \{answer_choices\} (yes/no)

Prompt ID: 5-3
Input template: For \{user_desc\}, do you think it is good to recommend \{item_title\}?
Target template: \{answer_choices\} (yes/no)

Prompt ID: 5-4