OpenP5: Benchmarking Foundation Models for Recommendation

Shuyuan Xu
Rutgers University
New Brunswick, NJ, US
shuyuan.xu@rutgers.edu

Wenyue Hua
Rutgers University
New Brunswick, NJ, US
wenyue.hua@rutgers.edu

Yongfeng Zhang
Rutgers University
New Brunswick, NJ, US
yongfeng.zhang@rutgers.edu

ABSTRACT

This paper presents OpenP5, an open-source library for benchmarking foundation models for recommendation under the Pre-train, Personalized Prompt and Predict Paradigm (P5). We consider the implementation of P5 on three dimensions: 1) downstream task, 2) recommendation dataset, and 3) item indexing method. For 1), we provide implementation over two downstream tasks: sequential recommendation and straightforward recommendation. For 2), we surveyed frequently used datasets in recommender system research in recent years and provide implementation on ten datasets. In particular, we provide both single-dataset implementation and the corresponding checkpoints (P5) and another Super P5 (SP5) implementation that is pre-trained on all of the datasets, which supports recommendation across various domains with one model. For 3), we provide implementation of three item indexing methods to create item IDs: random indexing, sequential indexing, and collaborative indexing. We also provide comprehensive evaluation results of the library over the two downstream tasks, the ten datasets, and the three item indexing methods to facilitate reproducibility and future research. We open-source the code and the pre-trained checkpoints of the OpenP5 library at https://github.com/agiresearch/OpenP5.

CCS CONCEPTS

• **Information systems** → *Recommender systems*; • **Computing methodologies** → *Machine learning*; *Natural language processing*.

KEYWORDS

Foundational Model; Recommender System; Generative Recommendation; Open Source

1 INTRODUCTION

The recent surge in interest around foundation models, including Large Language Models (LLM), within both academic and industrial domains has been largely attributed to their significant contributions across various research fields, including natural language processing (NLP) [1, 4, 14] and computer vision (CV) [22, 27]. In the sphere of recommender systems, practitioners and researchers are progressively incorporating these models into recommendation tasks. Certain recent studies, such as P5 [7] and M6 [5], have efficaciously harnessed the advantages of large language models to facilitate generative recommendation by transforming recommendation tasks into natural language formats. Nevertheless, despite the intensifying focus on the utilization of foundation models within recommendation systems, the field remains relatively nascent, and the absence of standardized benchmarks might impede the rapid evolution of this budding area.

This paper endeavors to address the gap concerning the absence of standardized benchmarks in the realm of recommendation foundation models by introducing OpenP5. OpenP5 is an open-source library for benchmarking foundation models for recommendations, built upon the principles of the P5 model [7]. It incorporates three dimensions of the P5 model [7]: downstream task, recommendation dataset, and item indexing method.

In recommendation foundation models, language serves as an efficacious medium to integrate various recommendation downstream tasks into a singular model. Hence, OpenP5 considers the two most prevalent tasks in recommender systems: sequential recommendation and straightforward recommendation. The former requires the model to generate recommended items based on user history, while the latter mandates the model to generate recommendations solely based on user ID.

To facilitate researchers and practitioners, the OpenP5 library includes a diverse range of commonly employed public recommendation datasets. We provide a comprehensive survey of the popular datasets used in recent years, and incorporates the top 10 datasets into the library. The library also includes the implementation of Super P5 (SP5), a versatile model with the ability to recommend items from all selected datasets using a singular model.

The cruciality of assigning a unique ID to each item in recommendation foundation models is underscored, ensuring that each item is represented by a minimal number of tokens and can be differentiated from other items, and to avoid hallucination problems in generative recommendation [9]. Moreover, the item indexing method can greatly impact the performance of the recommendation foundation models. Existing studies have adapted several item representation methods while transforming recommendation tasks into language generation tasks. For instance, P5 [7] uses number tokens, M6 [5] leverages rich metadata to generate metadata-based embeddings to represent items, and LMRecSys [25] utilizes item titles as representation. However, considering many public datasets may not include rich metadata or textual information, OpenP5 library includes only three item indexing methods based solely on user-item interactions: random indexing, sequential indexing, and collaborative indexing.

In conclusion, the OpenP5 library offers an implementation of the recommendation foundation model based on P5 principles [7], encompassing two downstream tasks. It also provides checkpoints for the top ten popular public datasets and an implementation of SP5 pre-trained on all datasets over three item indexing methods.

2 PERSONALIZED PROMPT COLLECTION

Recommendation foundation models possess the capability to integrate various downstream tasks of recommendation into a singular generative model [5, 7]. Acknowledging that some public datasets may not encompass certain information such as reviews, metadata, explicit feedback, and so forth, the OpenP5 library focuses solely on the two most commonly employed downstream tasks in recommender systems: the **sequential recommendation** task and

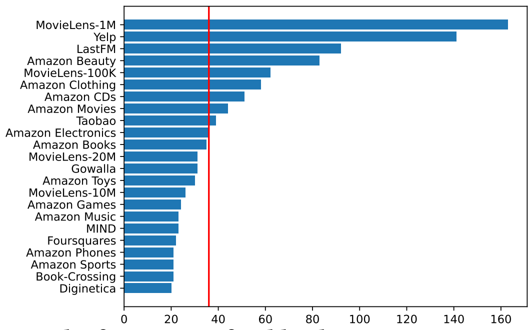

Figure 1: The frequency of public datasets in recent publications. We only show the datasets with more than 20 occurrences in recent three years at RecSys, CIKM, WSDM, KDD, WWW, and SIGIR. We select top 10 datasets in OpenP5 library. The vertical red line represents the threshold.

the **straightforward recommendation** task. Both of the tasks encompass several personalized prompts tailored to individual users. More specifically, various prompt templates have been designed for both tasks, which are filled with personalized information such as user ID and item ID. In addition, to circumvent the issue of recommending items from divergent datasets in SP5 (for instance, recommending a Yelp restaurant to an Amazon user), the dataset name has been included within our designed prompts.

The sequential recommendation task requires generating recommended items based on the user's history, and thus the personalized prompts incorporate the dataset name, user ID, user history, and target item ID. The straightforward recommendation task requires the model to generate recommended items given only the user ID, hence the prompts for this task exclude user history.

The OpenP5 library offers 11 distinct prompt templates for each downstream task. From each task, a single prompt template is selected as the *unseen* prompt, which serves to evaluate the model's zero-shot generalization capabilities. Notably, the OpenP5 library is designed with flexibility in mind, enabling users to modify the prompt templates according to their specific needs or objectives. This adaptability enhances the utility of the library, allowing it to better align with a range of research contexts and requirements.

3 DATASET COLLECTION

Dataset Selection. To identify popular public datasets suitable for recommendations, we conduct a frequency analysis of their occurrence in recent publications. More specifically, we examine papers accepted in the preceding three years at related conferences, including RecSys, CIKM, WSDM, KDD, WWW, and SIGIR. Using the ACM Digital Library[1], we filter relevant publications with the keywords "recommend", "recommender", "recommendation" and "collaborative". Due to the large number of public datasets, we only show the frequency of datasets with more than 20 occurrences in Figure 1. We include the top 10 popular public datasets in the OpenP5 library, including *Movielens-1M*, *Yelp*, *LastFM*, *Amazon Beauty*, *Movielens-100K*, *Amazon Clothing*, *Amazon CDs*, *Amazon Movies*, *Taobao*, and *Amazon Electronics*. This approach ensures that the datasets selected are not only popular but also align with current research trends in recommender systems.

[1]https://dl.acm.org/

Dataset Preprocessing. The ten datasets selected for this study span various recommendation scenarios and platforms. The Amazon[2] dataset originates from the *amazon.com* e-commerce platform, encompassing a broad spectrum of user interactions with products from different categories. The Yelp[3] dataset consists of extensive user ratings and reviews for business recommendations. The MovieLens[4] dataset offers a collection of movie ratings made by MovieLens users. The LastFM[5] dataset provides music artist listening records sourced from users of the Last.fm online music system. Finally, the Taobao[6] dataset features a multitude of user behaviors from the Taobao e-commerce platform.

We subjected all datasets to a common 5-core setting filter, which effectively removes inactive users and items that have recorded fewer than 5 interactions. For the Taobao dataset, we specifically included only the purchase behavior. As for the LastFM dataset, we employed the filtered data[7] provided by [26]. We provide the statistical overview of all datasets in our GitHub repository[8].

Following [26], we categorized the interactions by users and arranged them in ascending order based on their interaction timestamps. We then partitioned the interactions into training, validation, and testing data following the widely adopted leave-one-out policy.

4 ITEM INDEXING METHODS

In order to transform recommendation tasks into language generation tasks, user and item identifiers need to be compatible with natural language. This compatibility ensures that these identifiers can be seamlessly incorporated into natural language instructions used for the pre-training, fine-tuning, and prompting stages of Large Language Models (LLMs). In this section, we detail three item indexing methods implemented in the OpenP5 library: random indexing, sequential indexing, and collaborative indexing.

4.1 Random Indexing

Random indexing represents a straightforward approach to item indexing. This method assigns each item a unique random number that serves as the item ID. Within the model, the SentencePiece tokenizer [17] is employed to further tokenize this random number ID into a sequence of tokens. For instance, an item with the randomly assigned unique ID of "2048" would be tokenized into the tokens "20" and "48" within the recommendation foundation model.

While random indexing is frequently employed in traditional recommendation systems, it may not be optimally suited for foundation models: the potential drawback stems from the fact that the randomly assigned IDs are further tokenized, which can inadvertently cause unrelated items to share identical tokens. To illustrate, the items "2048" and "2049", despite being completely unrelated and not even interacted with by the same user, share the token "20". Consequently, the model could mistakenly establish a semantic relationship between these items, thereby affecting the accuracy of the recommendations [9].

[2]https://cseweb.ucsd.edu/ jmcauley/datasets/amazon/links.html
[3]https://www.yelp.com/dataset
[4]https://grouplens.org/datasets/movielens/
[5]https://grouplens.org/datasets/hetrec-2011/
[6]https://tianchi.aliyun.com/dataset/649
[7]https://github.com/RUCAIBox/CIKM2020-S3Rec
[8]https://github.com/agiresearch/OpenP5

Algorithm 1 Method for Collaborative Indexing

```
Require: Training data user sequence $D$, number of clusters $N$ to
    be created, number of items $k$ in the largest allowed cluster
 1: Instantiate a queue and enqueue all items as one set
 2: while queue is not empty do
 3:     Dequeue the first item set $S$
 4:     if The size of $S < k$ then
 5:         Assign a unique token to all items within $S$
 6:     else
 7:         Compute the co-occurrence matrix $M$ for items in $S$
            based on $D$
 8:         Apply spectral clustering on $M$ into $N$ clusters
 9:         Generate unique tokens for each cluster and assign corresponding tokens to all items within $S$ based on the clustering
10:         Enqueue all resultant clusters
11:     end if
12: end while
```

4.2 Sequential Indexing

To mitigate the issues associated with random indexing, one viable strategy involves integrating collaborative information into item IDs. A basic implementation of this approach can be observed in the sequential indexing method [9]. This method assigns consecutive number IDs to users' consecutive interactions, commencing with the first user and progressing through to the last user. It iterates across all interactions, assigning a new, incrementally increasing ID to any item that has not yet been assigned. Importantly, we apply the sequential indexing method solely to the training data to circumvent potential data leakage during the evaluation phase.

For items subjected to sequential indexing, the sharing of an identical token at the same position following tokenization between two items suggests that these two items may have been interacted with by the same user. Consequently, the sequential information embedded within the item IDs could potentially augment the effectiveness of the foundational model's recommendations.

4.3 Collaborative Indexing

To integrate further collaborative information into item indexing, we have incorporated a collaborative indexing method within the OpenP5 library. The underlying intuition of the collaborative indexing method is predicated on the idea that the frequency of co-occurrence of items should influence the degree to which they share the same token at the same position [9]. This concept is represented as a graph, with nodes signifying items and edge weights denoting co-occurrence frequency. To engender collaborative indexing, we employ the spectral clustering method [13, 21]. Given that the collaborative indexing method necessitates the introduction of Out-of-vocabulary (OOV) tokens to construct item indices, we denote these OOV tokens with angle brackets "$\langle\rangle$" (e.g., "$\langle CI1\rangle$"). A detailed exposition of this method can be found in Algorithm 1.

5 EXPERIMENTS

5.1 Datasets and Baselines

We have introduced the dataset collection in Section 3. Due to the space limitation, we present experimental results for only three datasets: Movielens-1M, Amazon Beauty, LastFM. The remaining results can be accessed via our GitHub repository[9]. To demonstrate the superior performance of the OpenP5 library, we gather a collection of representative approaches for different downstream tasks.

Sequential Recommendation. Since our OpenP5 only uses user interaction information for prediction, for fair comparision, we adopt several prominent sequential recommendation baselines that also use user interaction information only. **Caser** [19] views sequential recommendation as a Markov Chain and employ Convolutional Neural Networks (CNNs) to model users. **HGN** [11] uses hierarchical gating networks to learn user behaviors from both long-term and short-term perspectives. **GRU4Rec** [8] leverages Gated Recurrent Units (GRU) [3] to model user interaction sequences. **Bert4Rec** [18] utilizes BERT-style masked language modeling [6] to learn a bidirectional representation for sequential recommendation. **FDSA** [24] models feature sequences with a self-attention module. **SASRec** [10] deploys a self-attention mechanism within a sequential recommendation model.

Straightforward Recommendation. We utilize three existing methods as our baselines for straightforward recommendation task. **BPR-MF** [15] leverages matrix factorization with the pairwise Bayesian Personalized Ranking (BPR) loss. **BPR-MLP** [2] utilize MLP to model users and items. **SimpleX** [12] leverages cosine contrastive loss (CCL) in collaborative filtering for recommendation.

5.2 Implementation Details

Following the P5 framework [7], our implementation is grounded in the T5 model [14], specifically leveraging the T5-small pre-trained checkpoint. Notably, we randomly initialize the embedding of number-related tokens in the pre-trained checkpoint. This is predicated on the fact that while these embeddings encapsulate semantic similarity amongst tokens in the pre-training phase, such semantic patterns may not carry over to recommendation tasks when items are indexed with numerical identifiers. For prompt during the training, we select 10 prompts for each task, reserving one to evaluate zero-shot generalization. To improve the efficiency, we sample a set of prompts in each epoch. To alleviate the potential forgetting problem, we employ a training regimen wherein batches from different tasks are alternated. For SP5, addressing the data imbalance and potential forgetting problem is paramount. Hence, we alternate batches derived from different datasets and tasks. For smaller datasets, we repeat iterations until the completion of the largest dataset to ensure a balanced training process.

5.3 Results Analysis

The performance metrics for the sequential recommendation task and the straightforward recommendation task are presented in Table 1 and Table 2 respectively. Specifically, we use the top-k Hit Ratio (HR@k) and Normalized Discounted Cumulative Gain (NDCG@k) to evaluate performance, providing the results for HR@5,10, and NDCG@5,10. The best result for each metric is highlighted in bold, while the second-best result is underlined.

From the recommendation performance, we can observe that OpenP5 achieves the best performance in most cases. A comparison of three item indexing methods reveals the anticipated lower performance of the random indexing method, with the sequential

[9]https://github.com/agiresearch/OpenP5

Methods	ML1M				Beauty				LastFM			
	HR@5	NCDG@5	HR@10	NCDG@10	HR@5	NCDG@5	HR@10	NCDG@10	HR@5	NCDG@5	HR@10	NCDG@10
Caser	0.0912	0.0565	0.1442	0.0734	0.0205	0.0131	0.0347	0.0176	0.0303	0.0178	0.0413	0.0214
HGN	0.1430	0.0874	0.2404	0.1231	0.0325	0.0206	0.0512	0.0266	0.0321	0.0175	0.0505	0.0233
GRU4Rec	0.0806	0.0475	0.1344	0.0649	0.0164	0.0099	0.0283	0.0137	0.0275	0.0158	0.0367	0.0187
BERT4Rec	0.1308	0.0804	0.2219	0.1097	0.0203	0.0124	0.0347	0.0170	0.0422	0.0269	0.0633	0.0337
FDSA	0.1167	0.0762	0.1868	0.0987	0.0267	0.0163	0.0407	0.0208	0.0303	0.0219	0.0413	0.0254
SASRec	0.1078	0.0681	0.1810	0.0918	0.0387	0.0249	0.0605	0.0318	**0.0505**	0.0331	0.0688	0.0390
OpenP5-R (seen)	0.1098	0.0734	0.1575	0.0888	0.0318	0.0226	0.0464	0.0273	0.0156	0.0104	0.0312	0.0153
OpenP5-S (seen)	0.1901	0.1229	0.2849	0.1535	**0.0457**	**0.0336**	**0.0622**	**0.0389**	0.0394	0.0262	0.0578	0.0321
OpenP5-C (seen)	**0.2066**	**0.1400**	**0.2945**	0.1683	0.0421	0.0285	0.0601	0.0346	0.0453	0.0301	0.0674	0.0370
SP5-R (seen)	0.0305	0.0185	0.0558	0.0267	0.0073	0.0050	0.0111	0.0062	0.0037	0.0022	0.0064	0.0031
SP5-S (seen)	0.1058	0.0696	0.1589	0.0866	0.0130	0.0067	0.0224	0.0097	0.0101	0.0061	0.0156	0.0079
SP5-C (seen)	0.1490	0.0984	0.2225	0.1221	0.0276	0.0192	0.0391	0.0229	0.0192	0.0130	0.0284	0.0160
OpenP5-R (unseen)	0.1058	0.0693	0.1533	0.0847	0.0313	0.0222	0.0456	0.0267	0.0128	0.0072	0.0248	0.0110
OpenP5-S (unseen)	0.1916	0.1236	0.2854	**0.1737**	0.0452	0.0332	0.0613	0.0384	0.0404	0.0265	0.0606	0.0331
OpenP5-C (unseen)	0.2055	0.1386	0.2940	0.1672	0.0412	0.0286	0.0600	0.0346	0.0504	**0.0332**	**0.0724**	**0.0420**
SP5-R (unseen)	0.0306	0.0190	0.0541	0.0264	0.0076	0.0051	0.0116	0.0064	0.0046	0.0025	0.0092	0.0039
SP5-S (unseen)	0.1046	0.0688	0.1586	0.0862	0.0183	0.0122	0.0266	0.0149	0.0083	0.0049	0.0147	0.0070
SP5-C (unseen)	0.1064	0.0702	0.1685	0.0901	0.0240	0.0162	0.0339	0.0193	0.0101	0.0080	0.0211	0.0117

Table 1: Performance results on sequential recommendation task. R, S, C represent three item indexing.

Methods	ML1M				Beauty				LastFM			
	HR@5	NCDG@5	HR@10	NCDG@10	HR@5	NCDG@5	HR@10	NCDG@10	HR@5	NCDG@5	HR@10	NCDG@10
BPR-MF	0.0141	0.0081	0.0301	0.0133	0.0224	0.0149	0.0363	0.0204	0.0218	0.0147	0.0253	0.0162
BPR-MLP	0.0123	0.0068	0.0270	0.0116	0.0193	0.0127	0.0305	0.0176	0.0211	0.0150	0.0321	0.0185
SimpleX	0.0301	0.0133	0.0596	0.0206	0.0300	0.0189	**0.0471**	0.0245	0.0312	0.0211	0.0523	0.0277
OpenP5-R (seen)	0.0215	0.0133	0.0348	0.0176	0.0233	0.0169	0.0317	0.0196	0.0239	0.0151	0.0294	0.0169
OpenP5-S (seen)	0.0310	0.0192	0.0571	0.0275	**0.0317**	**0.0239**	0.0437	**0.0277**	0.0376	0.0259	**0.0661**	**0.0350**
OpenP5-C (seen)	**0.0347**	**0.0224**	**0.0618**	**0.0309**	0.0294	0.0206	0.0444	0.0254	**0.0404**	**0.0270**	0.0615	0.0336
SP5-R (seen)	0.0114	0.0074	0.0243	0.0115	0.0039	0.0021	0.0087	0.0036	0.0012	0.0009	0.0073	0.0022
SP5-S (seen)	0.0121	0.0082	0.0224	0.0115	0.0130	0.0067	0.0224	0.0097	0.0027	0.0012	0.0073	0.0027
SP5-C (seen)	0.0214	0.0135	0.0344	0.0177	0.0176	0.0121	0.0278	0.0154	0.0183	0.0118	0.0321	0.0161
OpenP5-R (unseen)	0.0177	0.0114	0.0301	0.0154	0.0078	0.0051	0.0127	0.0066	0.0183	0.0117	0.0284	0.0150
OpenP5-S (unseen)	0.0190	0.0122	0.0368	0.0178	0.0122	0.0089	0.0182	0.0109	0.0128	0.0076	0.0202	0.0099
OpenP5-C (unseen)	0.0210	0.0134	0.0303	0.0164	0.0139	0.0089	0.0226	0.0117	0.0174	0.0117	0.0257	0.0144
SP5-R (unseen)	0.0086	0.0056	0.0209	0.0095	0.0017	0.0009	0.0042	0.0017	0.0018	0.0009	0.0046	0.0017
SP5-S (unseen)	0.0105	0.0066	0.0177	0.0088	0.0044	0.0024	0.0093	0.0040	0.0073	0.0039	0.0147	0.0062
SP5-C (unseen)	0.0126	0.0082	0.0238	0.0117	0.0068	0.0034	0.0113	0.0049	0.0028	0.0011	0.0092	0.0032

Table 2: Performance results on straighforward recommendation task. R, S, C represent three item indexing.

indexing method trailing slightly behind the collaborative indexing method. Remarkably, OpenP5 delivers strong results in the sequential recommendation task with unseen prompts. Certain datasets even demonstrate better performances with unseen prompts than seen ones, as evidenced in OpenP5-S on ML-1M and LastFM. However, for the straightforward recommendation task, performance decreases with unseen prompts, possibly because the word tokens of unseen prompt template dominate over the tokens of user ID, affecting the accuracy of recommendations. As for SP5, it outperforms some baselines on the ML1M dataset. However, the performance on Beauty and LastFM is suboptimal, potentially due to overfitting of these smaller datasets while addressing dataset imbalance during SP5 training, as discussed in Section 5.2.

6 CONCLUSION AND FUTURE WORK

In this paper, we provide the OpenP5 library for benchmarking foundation models in recommendation. We consider the implementation on three perspectives: downstream tasks, recommendation datasets, and item indexing methods. The library serves as a continous effort to develop and evaluate foundation models for recommendation and helps the community to advance further on this direction with future innovations. In the future, we will consider incorporating more item indexing methods, more foundation model traning and inference paradigms, more data modalities, and more backbone LLMs into the library, such as OPT [23], LLaMA [20], BLOOM [16] and others.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[4] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[5] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. *arXiv preprint arXiv:2205.08084* (2022).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

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

[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

[9] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to Index Item IDs for Recommendation Foundation Models. *arXiv preprint arXiv:2305.06569* (2023).

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

[11] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 825–833.

[12] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A simple and strong baseline for collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1243–1252.

[13] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001).

[14] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[16] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100* (2022).

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1715–1725.

[18] 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. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

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

[20] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[21] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[22] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. 2021. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432* (2021).

[23] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[24] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation.. In *IJCAI*. 4320–4326.

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

[26] Kun Zhou, Hui 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. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1893–1902.

[27] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to prompt for vision-language models. *International Journal of Computer Vision* 130, 9 (2022), 2337–2348.

APPENDIX

In this appendix, we provide the full list of the personalized prompts for both downstream tasks.

A Sequential Recommendation

Prompt Seen: A1
Input Template: Considering {dataset} user_{user_id} has interacted with {dataset} items {history} . What is the next recommendation for the user ?
Target Template: {dataset} {target}

Prompt Seen: A2
Input Template: Here is the purchase history of {dataset} user_{user_id} : {dataset} item {history} . I wonder what is the next recommended item for the user .
Target Template: {dataset} {target}

Prompt Seen: A3
Input Template: {dataset} user_{user_id} has purchased {dataset} items {history} , predict next possible item to be bought by the user ?
Target Template: {dataset} {target}

Prompt Seen: A4
Input Template: I find the purchase list of {dataset} user_{user_id} : {dataset} items {history} , I wonder what other itmes does the user need . Can you help me decide ?
Target Template: {dataset} {target}

Prompt Seen: A5
Input Template: According to what items {dataset} user_{user_id} has purchased : {dataset} items {history} , Can you recommend another item to the user ?
Target Template: {dataset} {target}

Prompt Seen: A6
Input Template: What would {dataset} user_{user_id} be likely to purchase next after buying {dataset} items {history} ?
Target Template: {dataset} {target}

Prompt Seen: A7
Input Template: By analyzing the {dataset} user_{user_id} 's purchase of {dataset} items {history} , what is the next item expected to be bought ?
Target Template: {dataset} {target}

Prompt Seen: A8
Input Template: Can you recommend the next item for {dataset} user_{user_id} , given the user 's purchase of {dataset} items {history} ?
Target Template: {dataset} {target}

Prompt Seen: A9
Input Template: After buying {dataset} items {history} , what is the next item that could be recommended for {dataset} user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: A10
Input Template: The {dataset} user_{user_id} has bought items : {dataset} items {history} , What else do you think is necessary for the user ?
Target Template: {dataset} {target}

Prompt Unseen: A11
Input Template: What is the top recommended item for {dataset} user_{user_id} who interacted with {dataset} item {history} ?
Target Template: {dataset} {target}

B Straightforward Recommendation

Prompt Seen: B1
Input Template: What should we recommend for {dataset} user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: B2
Input Template: {dataset} user_{user_id} is looking for some items . Do you have any recommendations ?
Target Template: {dataset} {target}

Prompt Seen: B3
Input Template: Do you have any suggested items for dataset user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: B4
Input Template: Which recommendation should we provide to {dataset} user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: B5
Input Template: How can we assist {dataset} user_{user_id} with a recommendation ?
Target Template: {dataset} {target}

Prompt Seen: B6
Input Template: What would be a suitable recommendation for {dataset} user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: B7
Input Template: What would be a helpful recommendation for {dataset} user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: B8
Input Template: Can you recommend an item for {dataset} user_{user_id} ?
Target Template: {dataset} {target}

Prompt Seen: B9
Input Template: Based on {dataset} user_{user_id} 's interests and requirements , what item would you suggest to try ?
Target Template: {dataset} {target}

Prompt Seen: B10
Input Template: For {dataset} user_{user_id} , what item stands out as a top recommendation that they should consider ?
Target Template: {dataset} {target}

Prompt Unseen: B11
Input Template: What is the top recommendation for {dataset} user_{user_id} ?
Target Template: {dataset} {target}