OpenP5: An Open-Source Platform for Developing, Training, and Evaluating LLM-based Recommender Systems

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

In recent years, the integration of Large Language Models (LLMs) into recommender systems has garnered interest among both practitioners and researchers. Despite this interest, the field is still emerging, and the lack of open-source R&D platforms may impede the exploration of LLM-based recommendations. This paper introduces OpenP5, an open-source platform designed as a resource to facilitate the development, training, and evaluation of LLM-based generative recommender systems for research purposes. The platform is implemented using the encoder-decoder LLMs (e.g., T5) and the decoder-only LLMs (e.g., LLaMA-2) across 10 widely recognized public datasets, catering to two fundamental recommendation tasks: sequential and straightforward recommendations. Recognizing the crucial role of item IDs in LLM-based recommendations, we have also incorporated three item indexing methods within the OpenP5 platform: random indexing, sequential indexing and collaborative indexing. Built on the Transformers library, the platform facilitates easy customization of LLM-based recommendations for users. OpenP5 boasts a range of features including extensible data processing, task-centric optimization, comprehensive datasets and checkpoints, efficient acceleration, and standardized evaluations, making it a valuable tool for the implementation and evaluation of LLM-based recommender systems. The open-source code and pretrained checkpoints for the OpenP5 library are publicly available at https://github.com/agiresearch/OpenP5.

CCS CONCEPTS

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

KEYWORDS

Large Language Model; Recommender System; Generative Recommendation; Open Source

ACM Reference Format:

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2024. OpenP5: An Open-Source Platform for Developing, Training, and Evaluating LLM-based Recommender Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3626772.3657883

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657883

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, 35] and computer vision (CV) [50, 54]. In the sphere of recommender systems, practitioners and researchers are progressively incorporating these models into recommendation tasks. Certain recent studies, such as P5 [12] 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 a standardized development platform might impede the rapid evolution of this budding area.

This paper endeavors to address the gap concerning the absence of a standardized development platform in the realm of recommendation foundation models by introducing OpenP5. OpenP5 is an open-source platform for developing, training, and evaluating LLM-based models for generative recommendation, built upon the principles of the P5 model [12]. It incorporates four dimensions of the P5 model [12]: backbone models, downstream task, recommendation dataset, and item indexing method.

In recommender systems utilizing Large Language Models (LLMs), the generative prowess is derived from the foundational LLMs. Contemporary LLM architectures are principally classified into three types: encoder-only, encoder-decoder, and decoder-only. The burgeoning LLMs predominantly adopt either the encoder-decoder or the decoder-only architecture. To this end, the OpenP5 platform incorporates a quintessential LLM representative for both the encoder-decoder and decoder-only architectures. Specifically, the T5 [35] model is included as a representative of the encoder-decoder architecture, while the LLaMA-2 [42] model epitomizes the decoder-only architecture.

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 ID and user history, while the latter mandates the model to generate recommendations solely based on user ID.

To facilitate researchers and practitioners, the OpenP5 platform 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. We also design a Super P5 (SP5) model to have a preliminary exploration of the potential of recommendation foundation models that have the ability to recommend items across various datasets using a singular model.

In OpenP5, we also include various methods to represent items in language. 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 [17]. 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 [12] uses number tokens, M6 [5] leverages rich metadata to generate metadata-based embeddings to represent items, and LMRecSys [53] utilizes item titles as representation. However, considering many public datasets may not include rich metadata or textual information, OpenP5 platform includes only three item indexing methods based solely on user-item interactions: random indexing, sequential indexing, and collaborative indexing [17].

In summary, OpenP5 offers a platform for developing, training and evaluating LLM-based recommendation systems based on the P5 principles [12], encompassing two downstream tasks, ten datasets, three ID creation methods, and supporting both encoder-decoder and decoder-only LLM architectures. It also provides checkpoints based on two backbone models for each of the ten popular public datasets and an implementation of SP5 pre-trained on all datasets over three item indexing methods. Furthermore, the platform also supports users to develop their customized methods based on our provided APIs, such as new ID creation methods, backbones, datasets, tasks, or evaluation methods, facilitating future research on LLM-based generative recommendation.

The remainder of this paper is organized as follows. In Section 2, we provide the necessary background and related work. In Section 3 to Section 7, we introduce how to process data in OpenP5, including raw data preprocessing, item indexing methods, personalized prompt collection and data preparation, explain the pre-training and fine-tuning details of OpenP5, provide the evaluation methods, and show the experimental results, which can help users of the platform to easily adapt OpenP5 platform to other data or tasks. Finally, we conclude the work and discuss the future directions in Section 8.

2 RELATED WORK

Recently, there have been several attempts to leverage the power of large language models into recommender systems. Following [16, 27], we introduce existing work in three dimensions: the role of LLM, how to adapt LLM, and the recommendation tasks.

With the powerful ability, LLM can participate in several components of recommender systems. LLM can be used for feature engineering, which takes the original data as input and generates rich textual features as data augmentations [32, 47]. LLM can also be used as feature representation extractor, which formulates features as embeddings. Using LLM to obtain feature representation can

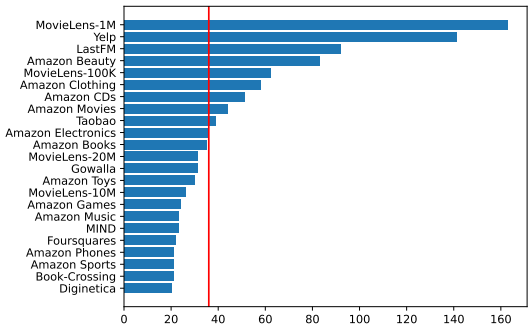

Figure 1: The frequency of public datasets in recent publications. We only show the datasets with more than 20 occurrences in recent three years at SIGIR, RecSys, WSDM, KDD, WWW, and CIKM. We select top 10 datasets in OpenP5 library. The vertical red line represents the threshold.

provide item or user representations with rich semantic information [23, 46, 48] and may be helpful for cross-domain or cold-start recommendation with natural language as connections [8, 10, 44]. Some works [5, 12, 24, 29] directly use LLM as recommender system, which is able to complete recommendation related tasks by LLM. In addition to being part of the recommendation system, LLM can also be used as a controller, possibly leading to a more interactive and explainable recommendation [9, 11].

Regarding how to adapt LLM, the LLM in recommender systems can be either tuned or not tuned, depending on whether the model will tune LLM during the training section or not. This includes both full fine-tuning and other parameter-efficient fine-tuning methods, such as LoRA [15]. With the emergence of large foundation models, researchers intend to analyze the zero-shot or few-shot performance of LLM in recommendation scenarios. Many existing work [6, 25, 28, 38, 40, 45] investigate zero-shot recommendation based on LLM without fine-tuning, which constructs prompts to instruct LLM on various recommendation tasks, such as rating prediction, pairwise comparison, reranking, etc. Although LLM may provide good language understanding performance, the recommendation performance without fine-tuning still needs future improvement, which indicates the importance of domain knowledge such as user-item interaction information from recommender systems [6, 28].

With the development of the field of recommender systems, the tasks of recommender systems are not limited to rating prediction or item recommendation. Traditional recommender systems are usually designed for a specific task, due to the difference on data format and model architecture for different tasks. With the help of LLM, language can serve as a bridge to integrate various downstream tasks into a singular model. Existing work on LLM-based recommender systems can be divided into multi-task recommender and task-specific recommender from the perspective of downstream tasks. Some work [18, 22, 34] incorporates LLM to improve the performance on specific tasks, while some other work [5, 12, 51] uses LLM to handle multi-task through a unified language format.

Based on the aspects introduced above, we define the OpenP5 platform as a tool for researchers and practitioners who are interested in developing fine-tuned LLM as recommender systems to perform multiple recommendation tasks.

User	Interaction History
A1YJEY40YUW4SE	B004756YJA B004ZT0SSG B0020YLEYK 7806397051 B002WLWX82
A60XNB876KYML	B0009P4PZC B009HULFLW B00BZ1QN2C B00G2TQNZ4 B00812ZWOS 7806397051 B0000YUX4O
A3G6XNM240RMWA	7806397051 B003H8180I B00538TSMU B002S8TOYU B001MP471K B00011JI88 B00C1F13CQ B003ATNYJC B003ZS6ONQ
A1PQFP6SAJ6D80	B0030HKJ8I B00027D8IC B002PMLGOU B00BN1MPPS 7806397051 B004Z40048
A38FVHZTNQ271F	7806397051 B009PZVOF6 B008LQX8J0 B007EHWDTS B009DDGHFC B002BGDLDO B003VWZCMK B00DQ2ILQY B00DAYGJVW

Table 1: This table presents a few instances from *Amazon Beauty* data. The data is stored a text file format, with each line encapsulating the information pertaining to an individual user. Within every line, the initial element represents the user's raw ID, followed by item raw IDs, listed in chronological order based on the user's interaction history.

3 DATA PROCESSING

In this section, we will discuss the data processing module of our platform, which allows users to integrate new datasets and create customized extensions.

3.1 Raw Data Preprocessing

The OpenP5 platform provides 10 popular preprocessed public datasets. 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 SIGIR, RecSys, WSDM, KDD, WWW, and CIKM. 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. We provide the statistical overview of all datasets in our GitHub repository[2].

The preprocessed data is saved as a txt file, with an illustrative example from the *Amazon Beauty* dataset shown in Table 1. Our platform principally requires user-item interaction data, as additional information may not be available for most public datasets. More precisely, we segregate the information of different users into separate lines. Within each line, elements are divided by a space, where the first element denotes the user raw ID, and the subsequent elements – item raw IDs – chronologically delineate the user's interaction history. Users of the platform can effortlessly train models on new datasets by converting the raw data into the specified data format. The platform will automatically partition the data into training, validation, and testing sets.

3.2 Item Indexing

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). Our platform provides implementation of three item indexing methods: random indexing, sequential indexing, and collaborative indexing. After applying indexing methods, the results are saved as a txt file consisting of two values in each line, where the first value represents the raw ID, and the second value represents the reindexed ID. The user-item interaction data will be formulated in the same format as preprocessed data (i.e., Table 1) after indexing. We will introduce more details about the provided indexing methods in the platform.

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 [37] 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 [17]: 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. As the relationship stems from the index structure, they are impossible to eliminate no matter how the model learns from data, thereby affecting the accuracy of the recommendations [17]. Consequently, RID is considered an unfavorable method. However, we still include this simple indexing method in this platform in case researchers would like to use it as a baseline for comparison and exploration.

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, as illustrated in [17]. 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

[1] https://dl.acm.org/
[2] 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

Instantiate a queue and enqueue all items as one set
while queue is not empty **do**
  Dequeue the first item set S
  if The size of $S < k$ **then**
    Assign a unique token to all items within S
  else
    Compute the co-occurrence matrix M for items in S based on D
    Apply spectral clustering on M into N clusters
    Generate unique tokens for each cluster and assign corresponding tokens to all items within S based on the clustering
    Enqueue all resultant clusters
  end if
end while

by the same user. Consequently, the sequential information embedded within the item IDs could potentially augment the effectiveness of the foundational model's recommendations.

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 [17]. 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 [33, 43]. 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 "⟨⟩" (e.g., "⟨$CI1$⟩"). A detailed exposition of this method can be found in Algorithm 1.

3.3 Personalized Prompt Collection

Recommendation foundation models possess the capability to integrate various downstream tasks of recommendation into a singular generative model [5, 12]. 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 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 following examples illustrate the prompt templates for both downstream tasks.

Sequential Recommendation

```
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}
```

Straightforward Recommendation

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

The OpenP5 platform 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 platform is designed with flexibility in mind, enabling users to modify the prompt templates according to their specific needs or objectives. More specifically, the prompt templates are saved in a txt file, with a representative example depicted in Table 2. Each line delineates a distinct prompt template, encompassing four pieces of information separated by semicolons: the first specifies the task to which the prompt pertains; the second denotes whether this prompt template is exposed to the model during training; the third outlines the instruction for input; and the fourth signifies the recommended item as output. The personalized information within the prompt template is enclosed in curly brace, and are substituted with specific data during data processing.

4 MULTI-TASK LEARNING

In our previous discussion, we highlighted that the OpenP5 platform supports the two predominant recommendation tasks: sequential and straightforward recommendations. It worth mentioning that the platform is also architecturally equipped to incorporate additional tasks into its training paradigm. This extensibility is a cornerstone of the platform's design, allowing for a broader scope of learning and adaptability. This section is devoted to elucidating the multi-task learning framework provided by the platform.

When tasks are learned in a sequence rather than concurrently, the model is susceptible to the "forgetting problem" [20, 21, 26, 49], which predominantly enhances its performance on the latest task to the detriment of prior tasks. To counteract this, simultaneous task learning is imperative. A common, albeit intuitive, solution is to mix training data from various tasks. This method, however, is not without its pitfalls. In the case of our Super P5 (SP5) model, indiscriminately blending data from disparate datasets can still lead to the "forgetting problem" when the datasets are uneven in size. Additionally, tasks differ in their textual length requirements; sequential recommendations necessitate a history of interactions, resulting in longer input sequences compared to direct recommendations. This discrepancy can lead to excessive padding when batching data from multiple tasks. To avoid these issues, our platform ensures that each batch is task-homogeneous—data from the same task. This

Task	Type	Input	Output
sequential	seen	What would {dataset} user_{user_id} be likely to purchase next after buying {dataset} items {history} ?	{dataset} {target}
sequential	unseen	What is the top recommended item for {dataset} user_{user_id} who interacted with {dataset} item {history} ?	{dataset} {target}
straightforward	seen	What should we recommend for {dataset} user_{user_id} ?	{dataset} {target}
straightforward	unseen	What is the top recommendation for {dataset} user_{user_id} ?	{dataset} {target}

Table 2: This table displays a few instances of prompt templates. The prompt templates are stored in a text file, with each line representing a unique prompt template. Every prompt encompasses four types of information, delineated by semicolons: the recommendation task, indication of whether seen or unseen during training, input template, and output template.

strategy effectively mitigates forgetting and maintains efficiency across varying task demands.

5 PRE-TRAINING AND FINE-TUNING

Given the personalized prompt for multiple recommendation tasks, we then introduce the pre-training and fine-tuning of the OpenP5 platform.

To improve the efficiency of pre-training and fine-tuning, the OpenP5 platform incorporates two techniques. One is to enable distributed learning in Multi-GPU environment. Distributed learning enables the model to be learned within a shorter time to improve the efficiency. Apart from distributed learning, the OpenP5 platform incorporates efficient training methods, such as LoRA [15], which freezes the pre-trained model weights and injects trainable rank decomposition matrices to reduce the trainable parameters and improve the efficiency.

6 HOW TO CUSTOMIZE OPENP5

The platform facilitates the development of customized LLM-based recommendation models by users. This section illustrates how OpenP5 can be adapted from various angles through illustrative examples.

- **Incorporating New Datasets:** Integrating new datasets into the recommendation model is straightforward with our platform, provided that the datasets are properly formatted. This ease of integration supports the seamless training of LLM-based recommendation models with new data sources.
- **User/Item ID:** Our platform supports three distinct methods for indexing items, with the flexibility to introduce additional indexing strategies. For instance, adopting item titles as unique identifiers can be achieved by generating preprocessed data that utilizes these new IDs.
- **Adopting New Backbone Models:** Given that the platform's architecture is predicated on Transformers library[3], users can conveniently replace backbone models with alternative ones from the Transformers library.
- **Customizing Personalized Prompts:** Personalized prompt templates on the platform can be readily replaced, allowing for the inclusion of novel tasks into the training process. Moreover, the platform's capability to manage out-of-vocabulary (OOV) tokens enhances its utility for LLM-based recommendation models that require such functionality. For example, Geng et al. [13] introduced a multimodal foundation model that integrates item images into prompts, utilizing a visual encoder to transform images into image tokens, which are treated as OOV by the tokenizer.

In summary, OpenP5 exhibits remarkable flexibility in accommodating new datasets, indexing methodologies, backbone models, and tasks. This adaptability underscores its potential as a foundational tool for pioneering research in the domain of LLM-based generative recommendation systems.

7 EXPERIMENTS

7.1 Datasets and Baselines

We have introduced the dataset collection in Section 3. Due to the space limitation, we present experimental results on three datasets: Movielens-1M, Amazon Beauty, LastFM. The remaining results can be accessed via our GitHub repository[4]. To demonstrate the superior performance of the OpenP5 platform, 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 comparison, we adopt several prominent sequential recommendation baselines that also use user interaction information only. We introduce the baseline models for sequential recommendation as follows.

- **Caser** [41] views sequential recommendation as a Markov Chain and employ Convolutional Neural Networks (CNNs) to model users.
- **HGN** [30] uses hierarchical gating networks to learn user behaviors from both long-term and short-term perspectives.
- **GRU4Rec** [14] leverages Gated Recurrent Units (GRU) [3] to model user interaction sequences.
- **Bert4Rec** [39] utilizes BERT-style masked language modeling [7] to learn a bidirectional representation for sequential recommendation.
- **FDSA** [52] models feature sequences with a self-attention module.
- **SASRec** [19] 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** [36] leverages matrix factorization with the pairwise Bayesian Personalized Ranking (BPR) loss.
- **BPR-MLP** [2] utilize MLP to model users and items.
- **SimpleX** [31] leverages cosine contrastive loss (CCL) in collaborative filtering for recommendation, which is a very strong baseline that beats many graph-based recommendation models.

[3]https://huggingface.co/docs/transformers/en/index

[4]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-T5-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-T5-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-T5-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-T5-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-T5-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-T5-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-LLaMA-R (seen)	0.0300	0.0197	0.0470	0.0252	0.0018	0.0013	0.0024	0.0015	0.0193	0.0120	0.0284	0.0149
OpenP5-LLaMA-S (seen)	0.0714	0.0466	0.1094	0.0587	0.0022	0.0036	0.0013	0.0017	0.0101	0.0059	0.0202	0.0092
OpenP5-LLaMA-C (seen)	0.0012	0.0006	0.0026	0.0011	0.0002	0.0001	0.0007	0.0003	0.0018	0.0013	0.0018	0.0013
SP5-LLaMA-R (seen)	0.0008	0.0004	0.0033	0.0012	0.0007	0.0004	0.0014	0.0006	0.0028	0.0017	0.0055	0.0026
SP5-LLaMA-S (seen)	0.0045	0.0026	0.0118	0.0049	0.0009	0.0004	0.0017	0.0007	0.0009	0.0005	0.0037	0.0014
SP5-LLaMA-C (seen)	0.0026	0.0015	0.0041	0.0019	0.0004	0.0002	0.0009	0.0004	0.0009	0.0004	0.0018	0.0007
OpenP5-T5-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-T5-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-T5-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-T5-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-T5-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-T5-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
OpenP5-LLaMA-R (unseen)	0.0296	0.0200	0.0444	0.0247	0.0017	0.0011	0.0022	0.0013	0.0183	0.0108	0.0202	0.0113
OpenP5-LLaMA-S (unseen)	0.0556	0.0364	0.0877	0.0467	0.0029	0.0017	0.0045	0.0022	0.0128	0.0078	0.0202	0.0103
OpenP5-LLaMA-C (unseen)	0.0010	0.0006	0.0018	0.0009	0.0004	0.0002	0.0007	0.0003	0.0009	0.0004	0.0046	0.0015
SP5-LLaMA-R (unseen)	0.0015	0.0009	0.0028	0.0013	0.0013	0.0007	0.0019	0.0010	0.0009	0.0006	0.0028	0.0012
SP5-LLaMA-S (unseen)	0.0048	0.0030	0.0106	0.0048	0.0008	0.0005	0.0013	0.0006	0.0028	0.0018	0.0046	0.0024
SP5-LLaMA-C (unseen)	0.0026	0.0016	0.0043	0.0021	0.0004	0.0002	0.0019	0.0004	0.0009	0.0005	0.0009	0.0005

Table 3: Performance results on sequential recommendation task. R, S, C represent three item indexing.

7.2 Implementation Details

Following the P5 framework [12], our implementation is grounded in the T5 model [35] and LLaMA-2 [42] model. The T5 backbone is trained on full parameters, while LLaMA-2 is trained using LoRA [15]. 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 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 important. 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.

7.3 Results Analysis

The performance metrics for the sequential recommendation task and the straightforward recommendation task are presented in Table 3 and Table 4 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 shown in Table 3 and Table 4, we can observe that generative recommendation is capable of achieving the best performance in most cases compared with baselines but highly depends on the pre-trained backbone models and item indexing methods. Comparing T5-based OpenP5 and LLaMA-based OpenP5, the performance with T5 backbone is better than LLaMA backbone in most cases. This is potentially due to the large number of parameters in LLaMA backbone, which leads to underfitting with sparse recommendation data. For T5-based OpenP5, a comparison of the three item indexing methods reveals the anticipated lower performance of the random indexing method, with the sequential indexing method trailing slightly behind the collaborative indexing method. Conversely, the LLaMA-based OpenP5 model does not yield a clear preference for any indexing method, potentially attributable to the effects of sparse data on its larger parameter space. This shows future avenues for developing LLM-based generative recommendation models that are both effective and parameter-efficient.

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-T5-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-T5-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-T5-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-T5-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-T5-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-T5-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-LLaMA-R (seen)	0.0106	0.0061	0.0205	0.0093	0.0017	0.0012	0.0023	0.0014	0.0202	0.0122	0.0275	0.0146
OpenP5-LLaMA-S (seen)	0.0103	0.0066	0.0210	0.0104	0.0050	0.0035	0.0065	0.0040	0.0147	0.0112	0.0220	0.0134
OpenP5-LLaMA-C (seen)	0.0012	0.0007	0.0022	0.0011	0.0002	0.0001	0.0003	0.0002	0.0028	0.0046	0.0022	0.0028
SP5-LLaMA-R (seen)	0.0018	0.0008	0.0041	0.0016	0.0007	0.0004	0.0014	0.0006	0.0018	0.0018	0.0028	0.0021
SP5-LLaMA-S (seen)	0.0063	0.0034	0.0119	0.0052	0.0007	0.0004	0.0016	0.0007	0.0009	0.0004	0.0028	0.0009
SP5-LLaMA-C (seen)	0.0022	0.0012	0.0041	0.0018	0.0002	0.0001	0.0008	0.0003	0.0001	0.0001	0.0018	0.0006
OpenP5-T5-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-T5-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-T5-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-T5-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-T5-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-T5-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
OpenP5-LLaMA-R (unseen)	0.0094	0.0063	0.0190	0.0094	0.0014	0.0021	0.0011	0.0013	0.0202	0.0139	0.0202	0.0139
OpenP5-LLaMA-S (unseen)	0.0098	0.0066	0.0195	0.0097	0.0047	0.0032	0.0062	0.0038	0.0147	0.0108	0.0202	0.0126
OpenP5-LLaMA-C (unseen)	0.0005	0.0003	0.0015	0.0006	0.0003	0.0001	0.0004	0.0002	0.0037	0.0028	0.0037	0.0028
SP5-LLaMA-R (unseen)	0.0018	0.0009	0.0030	0.0013	0.0006	0.0003	0.0010	0.0004	0.0018	0.0009	0.0046	0.0019
SP5-LLaMA-S (unseen)	0.0048	0.0030	0.0098	0.0046	0.0011	0.0006	0.0014	0.0007	0.0018	0.0010	0.0037	0.0015
SP5-LLaMA-C (unseen)	0.0022	0.0011	0.0043	0.0018	0.0001	0.0001	0.0006	0.0002	0.0001	0.0001	0.0009	0.0003

Table 4: Performance results on straightforward recommendation task. R, S, C represent three item indexing.

8 CONCLUSION AND FUTURE WORK

In this paper, we provide the OpenP5 platform as a resource to facilitate the developing, training, and evaluating large language model based recommender systems. We consider the implementation on four perspectives: backbone models, downstream tasks, recommendation datasets, and item indexing methods. The platform serves as a continuous 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 training and inference paradigms, more data modalities, and more backbone LLMs into the platform.

ACKNOWLEDGEMENT

The work was supported in part by NSF IIS2046457 and IIS-2007907. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of the sponsors.

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}

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] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering ChatGPT's Capabilities in Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*.

[7] 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.

[8] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318* (2021).

[9] Luke Friedman, Sameer Ahuja, David Allen, Terry Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. 2023. Leveraging Large Language Models in Conversational Recommender Systems. *arXiv preprint arXiv:2305.07961* (2023).

[10] Junchen Fu, Fajie Yuan, Yu Song, Zheng Yuan, Mingyue Cheng, Shenghui Cheng, Jiaqi Zhang, Jie Wang, and Yunzhu Pan. 2023. Exploring Adapter-based Transfer Learning for Recommender Systems: Empirical Studies and Practical Insights. *arXiv preprint arXiv:2305.15036* (2023).

[11] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524* (2023).

[12] 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.

[13] Shijie Geng, Juntao Tan, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. 2023. VIP5: Towards Multimodal Foundation Models for Recommendation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

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

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[16] Wenyue Hua, Lei Li, Shuyuan Xu, Li Chen, and Yongfeng Zhang. 2023. Tutorial on Large Language Models for Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1281–1283.

[17] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to Index Item IDs for Recommendation Foundation Models. *SIGIR-AP* (2023).

[18] Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2023. Genrec: Large language model for generative recommendation. *arXiv e-prints* (2023), arXiv–2307.

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

[20] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[21] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

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

[23] Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. 2023. Exploring the Upper Limits of Text-Based Collaborative Filtering Using Large Language Models: Discoveries and Insights. *arXiv preprint arXiv:2305.11700* (2023).

[24] Xinyi Li, Yongfeng Zhang, and Edward C Malthouse. 2023. PBNR: Prompt-based News Recommender System. *arXiv preprint arXiv:2304.07862* (2023).

[25] Xinyi Li, Yongfeng Zhang, and Edward C Malthouse. 2023. A Preliminary Study of ChatGPT on News Recommendation: Personalization, Provider Fairness, Fake News. *arXiv preprint arXiv:2306.10702* (2023).

[26] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[27] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2023. How Can Recommender Systems Benefit from Large Language Models: A Survey. *arXiv preprint arXiv:2306.05817* (2023).

[28] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149* (2023).

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

[30] 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.

[31] 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.

[32] Sheshera Mysore, Andrew McCallum, and Hamed Zamani. 2023. Large Language Model Augmented Narrative Driven Recommendations. *arXiv preprint arXiv:2306.02250* (2023).

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

[34] Aleksandr V Petrov and Craig Macdonald. 2023. Generative Sequential Recommendation with GPTRec. *arXiv preprint arXiv:2306.11114* (2023).

[35] 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.

[36] 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.

[37] 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.

[38] Damien Sileo, Wout Vossen, and Robbe Raymaekers. 2022. Zero-shot recommendation as language modeling. In *European Conference on Information Retrieval*. Springer, 223–230.

[39] 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.

[40] Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *arXiv preprint arXiv:2304.09542* (2023).

[41] 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.

[42] 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).

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

[44] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Xiangnan He, Zhijin Wang, Bo Hu, and Zang Li. 2022. TransRec: Learning Transferable Recommendation from Mixture-of-Modality Feedback. *arXiv preprint arXiv:2206.06190* (2022).

[45] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. *arXiv preprint arXiv:2304.03153* (2023).

[46] Chuhan Wu, Fangzhao Wu, Tao Qi, Chao Zhang, Yongfeng Huang, and Tong Xu. 2022. Mm-rec: Visiolinguistic model empowered multimodal news recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2560–2564.

[47] Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. *arXiv preprint arXiv:2306.10933* (2023).

[48] Yang Yu, Fangzhao Wu, Chuhan Wu, Jingwei Yi, and Qi Liu. 2021. Tiny-newsrec: Effective and efficient plm-based news recommendation. *arXiv preprint arXiv:2112.00944* (2021).

[49] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Joemon Jose, Beibei Kong, and Yudong Li. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 696–705.

[50] 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).

[51] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001* (2023).

[52] 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.

[53] 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*.

[54] 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.