Universiteit Leiden

Master Computer Science

[Exploring Influencing Factors and Application Scenarios in Recommendation Models: A Comparative Study from Traditional to LLMs-based Recommendation Models]

Name: [Xiaoran Rong]
Student ID: [s3805425]

Date: [29/08/2015]

Specialisation: [Data Science]

1st supervisor: [Zhaochun Ren]
2nd supervisor: [Suzan Verberne]
External supervisor: [Jujia Zhao]

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University

Abstract

Recommendation systems play an important role in a wide range of domains, such as e-commerce, social media, and streaming media. It contains the well-established traditional recommendation models – sequential and non-sequential setting – and LLMs(Large Language Models)-based recommendation models.

Non-sequential setting of traditional recommendation models mainly include collaborative filtering (CF) and content-based methods resorting to similarity matrices. The most widely used technique of CF is matrix factorization (MF). MF projects users and items into one shared latent space and represents user and item with latent matrix. Then the interactions are modeled as the inner product of the latent vectors of user and item. Sequential recommendation systems aim to capture the context of user's actions, often using methods such as Markov Chains (MCs), Recurrent Neural Networks (RNNS), and Transformers.

Recently, the integration of Large Language Models(LLMs) with recommendation systems – LLMs-based recommendation models – have also gained significant attention. Broadly, LLMs-based recommendation models can be categorized into two main approaches, by leveraging the in-context learning ability of LLMs and fine-tuning LLMs. However, LLMs aren't designed specifically for recommendation tasks. As a result, although LLMs-based recommendation models have unique advantages, in natural language understanding and text generation, they are still in the early stage and many open challenges remain.

Besides, although the traditional models are well-established and quite mature, when the traditional models are proposed, the performance comparison of models mainly focus on the superiority of the proposed model over baselines on specific benchmarks datasets. When analyzing, they tend to pay more attention with the strengths of the proposed model, but neglecting its possible poor performance on other scenarios. At the same time, when exploring the underlying reason why different models exhibit varying performances on the benchmark datasets, they tend to neglect the perspective of baselines, only emphasizing the perspective of the proposed models.

As a result, the influencing factors and scenarios that different recommendation models show advantage are still unknown. In this thesis, we aim to bridge these gaps by conducting a comprehensive and decentralized comparison from traditional recommendation models, both sequential and non-sequential settings, to LLMs-based recommendation models. We select representative recommendation models, treat each selected model equally, and use datasets with fundamentally different characteristics.

In conclusion, we evaluate five recommendation models – NCF, SASRec, BERT4Rec, BIGRec, and LETTER – on four datasets: MovieLens 1M, Amazon 2018 Video Games category, Amazon 2018 Luxury Beauty category, and Amazon 2014 Beauty category.

We investigate how key factors such as data sparsity, user-item interaction patterns, the presence of sequence information, and domain specific features affect the performance of different models, and identify the suitable application scenarios for different models.

Based on our analysis and conclusion, we also provide the valuable reference for LLMs-based recommendation study and propose a few of potential directions for future research. It includes: 1. How LLMs-based recommendation models can handle cases where the order of items in data doesn't reflect the actual sequence of user behavior in the real world. 2. Exploring how user-centric information can be effectively incorporated to improve the recommendation quality of LLMs-based recommendation models.

Keywords: Collaborative Filtering, Sequential Recommendation System, LLMs-based Recommendation

Contents

1 Introduction **7**

2 Related work **9**
2.1 CF-based recommendations 9
2.2 Sequential recommendation 10
2.3 LLMs-based recommendation 10
2.4 Research Gaps 11

3 Preliminaries **11**
3.1 Problem Formulation 11
3.1.1 Traditional recommendation models 11
3.1.2 LLMs-based recommendation models 12
3.2 Model Architecture 13
3.2.1 Neural collaborative filtering 14
3.2.2 Self-attentive sequential recommendation 15
3.2.3 Sequential recommendation with bidirectional encoder representations from Transformer 16
3.2.4 Bi-step grounding paradigm for recommendation 18
3.2.5 Transformer index for generative recommenders 20
3.2.6 Learnable item tokenization for generative recommendation 22

4 Data **23**
4.1 Data sources and dataset scale 23
4.2 Structure, schema and examples 23
4.2.1 MovieLens 1M 23
4.2.2 Amazon Review Data 24
4.3 Descriptive statistics 25
4.3.1 MovieLens 1M 26
4.3.2 Amazon Review Data 27

5 Experimental Methodology **29**
5.1 Model-specific input requirements 29
5.2 Data consistency and evaluation protocol 29

6 Results and analysis **31**
6.1 Overall performance of five recommendation models across four datasets . 31
6.1.1 Case study: performance of recommendation models on Amazon 2018 Video Games dataset 32
6.1.2 Case study: performance of LETTER across datasets 35
6.1.3 Case study: performance of BERT4Rec across datasets 36
6.1.4 Case study: performance of SASRec across datasets 38
6.1.5 Case study: performance of NCF across datasets 40
6.2 Performance of traditional and LLMs-based recommendation model for sequential recommendation tasks across four datasets 42
6.2.1 Case study: in-depth analysis and comparative evaluation of BIGRec 42
6.2.2 Case study: the impact of SASRec on LETTER 46

7 Discussions **47**
7.1 Limitations 47

8 Conclusions **48**
8.1 Research questions 48
8.2 Future work 50

A Descriptive figures of datasets **51**

1 Introduction

Recommendation systems play an important role in a wide range of domains such as e-commerce, social media, and streaming media. Broadly, they can be categorized into traditional models – both sequential and non-sequential settings – and the emerging LLMs-based recommendation models.

Non-sequential settings from traditional recommendation models mainly include collaborative filtering and content-based methods. Collaborative filtering method models interactions between user and item features based on historical interaction data, while content-based method predicts based on the item-to-item similarity. Sequential recommendation settings take the context of user's activities into account by capturing the patterns from historical interactions, often using methods such as Markov Chains (MCs), Recurrent Neural Networks (RNNs), and Transformers.

Traditional recommendation models are well-established and widely used. And LLMs-based recommendation models, as the emerging approaches, have the powerful capacities to generate content directly, engage in conversations with users, and make recommendations based on natural language prompts. These capabilities provide great potential for developing more adaptive, personalized, and interactive recommendation systems.

But for the gap between LLMs and recommendation tasks, the factors influencing the performance of recommendation models, specifically LLMs-based recommendation models, are still uncover. Besides, even there are already many well-established traditional models and these models have been analyzed when published, this kind of analysis is insufficient.

All the published modes are compared with a range of strong baselines on multiple benchmark datasets, which makes such comparisons are model-centric. It focuses on positioning the proposed model as center and other models are treated as baselines to emphasize the better performance of the proposed model.

In such situation, even though the comparison is also conducted among different models' performances on several datasets, the analysis is insufficient. On one hand, they mainly emphasizes the strengths of the proposed model while overlooking its weakness on certain scenarios. On the other hand, the interpretation of results is generally centered on the proposed model, neglecting the perspectives on other models and not exploring the underlying reasons why different models exhibit varying performance on the given datasets.

As a result, even the traditional recommendation models are quite mature, the comprehensive and neutral perspective is still missing.

To bridge the gaps, in this thesis, we conduct the decentralized and comprehensive comparison. Each model is considered equally, rather than selecting one specific model as the center and others as baselines.

Moreover, we use datasets with fundamentally different characteristics. Beyond difference in statistics such density and interaction patterns, the datasets we use cover three application domains: movie, beauty, and video games.

There exists MovieLens 1M dataset collected from rating platform, which means that the order of user-item interactions doesn't reflect the actual sequence of user's behaviors accurately. There also are Amazon Review datasets – Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amaozn 2014 Beauty – from shopping platform, where the order of data aligns more accurately with actual behaviors.

For the selection of recommendation models, we focus on representative work from traditional – both sequential and non-sequential settings – and LLMs-based recommendation models. Each

one of these approaches has its own effective working mechanisms and distinct characteristics. For sequential tasks of traditional recommendation models, collaborative filtering (CF) is the most classic approach. It makes recommendations based on users' historical interactions. The most commonly used technique is matrix factorization (MF). MF projects users and items into one shared latent space, representing user and item with latent matrix. Then the interactions are modeled as the inner product of the latent vectors of the users and items. However, MF has limitations when dealing with implicit feedback, as MF relies on fixed inner product. On contrast, neural collaborative filtering (NCF) resorts to neural networks to model the interaction between user and item features based on implicit feedback, which addresses the limitations and is an improvement of traditional collaborative filtering. Therefore we select Neural collaborative filtering (NCF) as the representative work for non-sequential setting of traditional recommendation models.

Another line of work in this area is the sequential recommendation system to capture the context of user activities. One of the most widely-used approaches is Markov Chains (MCs). However, Markov Chains only considers the last click or selection of the users, ignoring the information of past clicks. These problems are addressed by using RNNs. When the user enters a website, the model is queried to do recommendations based on the the first item a user clicks as the initial input of the RNN. After that, each consecutive click of the user will be used to produce the output based on all the previous clicks. Markov Chains (MCs) and Recurrent Neural Networks (RNNs) come with different requirements and advantages, which has been combined by SASRec. SASRec (self-attentive sequential recommendation) is based on the self-attention mechanism. It aims to identify the relevant items from users' action history at each time step to predict the next item for users. This approach is able to capture long-term semantics with the efficiency of MCs, which base on relatively few actions. The limitation of SASRec is that it can only learn the users' historical behaviors from left to right. To improve the limitations, the BERT4Rec is proposed, which employs the deep bidirectional self-attention to model user behavior sequences. Thus SASRec and BERT4Rec are the selected representative work for sequential setting of traditional recommendation models.

Generally speaking, these recommendation models use two kinds of working mechanisms to do recommendations. The first one is mainly rely on user-item interactions, like NCF. The second kind of mechanism resort to sequential modeling, like SASRec and BERT4Rec. The LLMs-based recommendation systems, like BIGRec (Bi-step Grounding Paradigm for Recommendation) and LETTER (LEarnable Tokenizer for generaTivE Recommendation) introduce new mechanism by utilizing pre-trained knowledge and the powerful generative capabilities of LLMs. We select BIGRec and LETTER as the representative work for LLMs-based recommendation models, as LLMs-based discriminative recommendation and generative recommendation respectively.

Since no model is treated as baseline, all models are equally important. We analyze each model's performance on different datasets equally, rather than only focusing on reasons why the proposed model performs better. Through treating all models equally and using fundamentally different datasets, the findings are more generalizable and transferable. As a result, our work also provides a valuable reference for LLMs-based recommendation methods.

Overall, in this study, we conduct a comprehensive comparison and analysis among five recommendation models – NCF, SASRec, BERT4Rec, BIGRec, and LETTER – across four datasets: MovieLens 1M, Amazon 2018 Video Games category, Amazon 2018 Luxury Beauty Category, and Amazon 2014 Beauty category, using widely-used metrics, Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG).

Our analysis includes both dataset centric and model centric perspectives. We conduct the case study based on the overall performance, from conventional approaches, to sequential methods, and to generative recommendation methods. We also carry out a comparative analysis focusing on sequential recommendation models – SASRec, BERT4Rec, BIGRec, and LETTER – to examine their effectiveness and features.
In this thesis, the following research question are addressed:

- **RQ1:** What are the potential factors influencing the performance of recommendation models and how these factors make impact?
- **RQ2:** In what scenarios are different recommendation models are suitable?

Based on these findings, we also identify two potential directions for future research: 1. For the data where the order of items doesn't reflect the user's actual interaction sequence, how LLMs-based recommendation models address the limitation. 2. Future work of generative recommendation models could explore the injection of richer descriptive information, specifically the user-centric descriptive information.

2 Related work

2.1 CF-based recommendations

The early work from recommendation systems usually resorts to item-based neighborhood method [7], which is based on the similarity matrix of items. This method learns the item-to-item similarity and then predict based on the similarity between the item and the users' historical interacted items.
Another line of work is collaborative filter(CF) [22], modeling the interaction between user and item features based on their historical interactions. Among all the CF methods, the matrix factorization (MF) [15] is the most commonly used. It projects users and items into a shared latent space and uses vectors of latent features to represent the user or item. Then the interactions between users and items are modeled as the inner product of their latent vectors. However, the feedback is not only explicit but also implicit [29]. The implicit feedback doesn't demonstrate the users preference directly. For example, the user grades one item one point, which is a negative feedback, meaning that the user doesn't like the item directly. But lacking of grades from user doesn't mean the user doesn't like the ungraded items. It is possible that the users doesn't view the item yet. When dealing with such implicit feedback, the linearity feature of MF limits its capacity.
To solve the challenge of implicit data, various approaches have been proposed. Hu [11] considers all unobserved data as negative feedback. S. Rendle [25] samples negative instances from missing data. Liang [20] proposes to weight missing data. NeuCF (Neural network-based Collaborative Filtering) [9] uses multi-layer perception (MLP) to learn the interaction from data to address the implicit feedback limitation. Through the learning separate embeddings and the concatenation of the their last hidden layer, the fusion model of GMF and MLP, NeuMF (Neural Matrix Factorization), is achieved . Based on this fusion, NeuMF is more flexible compared with Neural Tensor Network (NTN) [30].
The corporation with neural networks can be traced back to 2007, firstly introduced by Salakhutdinov, the two-layer Restricted Boltzman Machines (RBM) [27]. Then autoencoders [28] and denoising autoencoders (DAEs) [19, 40] have also been applied into recommendation

system. In addition using DNNs to model the interactions as NeuMF, the DNNs have also been applied to learn from the auxiliary information. These auxiliary information can include text [18], images [8], and acoustic features [36].

2.2 Sequential recommendation

The early work in recommendation system does not take the context of users' activities into account. For sequential recommendation tasks, capturing the patterns based on historical interactions is very important. The most commonly used methods include Markov Chains (MCs) [26, 33] and Recurrent Neural Networks (RNNs) [10]. Besides, graph neural networks [6] and CNNs [33] have also been applied. However, MC-based methods often face the limitation of the MC order and RNNs-based methods has limited capacity for parallel computation and long-range dependencies learning [1].
To balance the long-term captures from RNNs and predictions based on recent actions from MC, the Self-Attention based Sequential Recommendation model (SASRec) [14] is proposed. This model applies self-attention mechanisms, suitable for parallel, for sequential recommendation tasks.
SASRec applies the left-to-right sequential model, but this unidirectional model only learn from the previous data and assumes the data has the strict order which is not always the same in reality. The sequential model BERT4Rec (Bidirectional Encoder Representations from Transformers) [32] is developed to address the challenge, learning users' interactions from both directions.
Similar with SASRec, BERT4Rec predicts the next item based on previous interactions. Compared with the unidirectional mechanism from SASRec and RNN based methods, the bidirectional feature makes BERT4Rec capture long-distance dependencies better.
Besides the usual sequential recommendations, there is also group sequential scenarios [31, 37]. In scenarios such as each member of the family share access to a TV or a Netflix account, individual preferences may vary significantly. Group sequential recommendations require to consider the groups' historical activities, rather than a single user. Effectively identifying and modeling individual preferences is a crucial challenge for group sequential recommendation tasks.

2.3 LLMs-based recommendation

Applying LLMs into the recommendation systems has giant potential [2] and current LLMs-based recommendation models can be broadly categorized into two main approaches. The first one focuses on the in-context learning (ICL) ability of LLMs [13, 39], but it is limited by the gap between LLMs and recommendation tasks. To address the limitation, the second category has been proposed by fine-tuning LLMs [21]. Fine-tuning can enhance LLMs more capable of specific types of tasks, such as recommendation tasks. To better use the giant generative ability of LLMs, the special cases TALLRec [4] and InstructRec [42] has been proposed through injecting recommendation data into instruction-tuning phase. However, all the models ignore the LLM's ability of ranking all items.
BIGRec (Bi-step Grounding Paradigm for Recommendation) [3] is one of the recommendation systems based on LLMs (Large Language Models) with the ability to rank all items, rather than negative sampling.

Considering the emphasis on the ability of ranking all items of BIGRec, it can also be categorized as LLMs-based discriminative recommendations [34]. Another line of work is the generative recommendation based on LLMs [17, 41]. The challenge caused by the gap between LLMs and recommendation tasks is how to item tokenization to transform recommendation data into LLMs language space. The common methods for item tokenization usually uses ID identifiers [23, 12], textual identifiers [3], and codebook-based methods [24]. However, these existing methods have some limitations including not using item semantic information or semantic information not hierarchically distributed, lacking of collaborative signals, and item generation bias. To solve the challenge, LETTER (a LEarnable Tokenizer for generaTivE Recommendation) [38] is proposed, which is an enhancement of TIGER (Transformer Index for GEnerative Recommenders) [24]. This model is developed based on RQ-VAE [16] to generate hierarchical semantic identifier, including semantic embedding extraction and semantic embedding quantization.

2.4 Research Gaps

Although traditional recommendation models – both sequential and non-sequential setting – have been extensively studied and quite mature. Most existing research for traditional models are model-centric.

Prior study mainly focus on the superiority of the proposed model over baselines on specific benchmark datasets. In most cases, they tend to emphasize the strengths of the proposed model, but neglecting its possible poor performance on some datasets which represent specific scenarios. In addition, the interpretation of why the proposed model performs better than baselines is focused on the proposed model, but overlooking the perspective of the baselines. At the same time, with the emerging of LLMs-based recommendation models, a comprehensive and balanced analysis about the recommendation models is still missing.

This thesis bridges the gap, as we conduct the comprehensive and decentralized comparison, by selecting representative recommendation models, treating each model equally and using datasets with fundamentally different characteristics.

As a result, this thesis uncover the potential factors influencing the performance of recommendation models and explore the scenarios that different recommendation models show advantages. Based on these findings, our thesis also provides transferable insights and valuable reference with which LLMs-based recommendation study could be inspired.

3 Preliminaries

This section describes the problem formulation and model architecture. Section 3.1 formalizes the problem for traditional and LLMs-based recommendation models. Section 3.2 introduces core mechanisms of five recommendation models, including TIGER.

3.1 Problem Formulation

3.1.1 Traditional recommendation models

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ denote a set of users. $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$ denotes a set of items. M and N denote the number of users and items respectively. Then the user-item interaction

matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ is defined as Equation 1, where y_{ui} denotes whether the interaction between user u and item i is observed or not.

$$y_{ui} = \begin{cases} 1, & \text{if interaction } (u,\, i) \text{ is observed;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Neural collaborative filtering (NCF) resorts to neural networks to model the interaction between user and item features based on implicit feedback. In that case, the NCF model is formulated as Equation 2, where $\hat{y}_{ui}$ denotes the predicted scores for user u and item i, Θ denotes the model parameters, and f represents the interaction function.

$$\hat{y}_{ui} = f(u, i \mid \Theta) \tag{2}$$

For sequential recommendation models, SASRec and BERT4Rec, the goal is to predict the next item for user u based on the historical behaviors of users. For each user $u \in \mathcal{U}$, the historical sequence is denoted as S^u, defined as Equation 3, where $|S^u|$ represents the sequence length of user u.

$$\left(S_1^u, S_2^u, \ldots, S_{|S^u|}^u\right), \quad S_t^u \in \mathcal{I} \tag{3}$$

At time t, sequential recommendation models aim to predict the next item, $t+1$ depending on the previous interacted items. Then the model is formulated as modeling denoted as Equation 4.

$$p\left(S_{t+1}^u = i \mid \mathbf{S}^u\right) \tag{4}$$

The main difference between SASRec and BERT4Rec lies how they learn from historical interactions. SASRec learns in unidirectional, from left to right, while BERT4REc adopts bidirectional approach. This fundamental difference leads to different training process.
For SASRec, the input consists of observed sequence up to a certain point, and the model is trained to predict the next item. In contrast, BERT4Rec learns by masking random items and the last item of the sequence and predicting the masked items using information from both directions.
Therefore, the formulation of SASRec and BERT4Rec are defined as from Equation 4 to Equation 5, where t^* denotes the target position model predict, $S_{1:t}^u$ denotes the interaction sequence of user u from time 1 to time t, M_u denotes the masked items of historical sequence of user u, and $S_{\setminus M_u}^u$ denotes the historical sequence of user u without the maksed items.

$$p\left(S_{t^*}^u = i \mid S^u\right) = \begin{cases} p\left(S_{t+1}^u = i \mid S_{1:t}^u\right), & SASRec : t^* = t+1,\ \text{context} = S_{1:t}^u, \\ p\left(S_t^u = i \mid S_{\setminus M_u}^u\right), & BERT4Rec : t^* = t \in M_u,\ \text{context} = S_{\setminus M_u}^u \end{cases} \tag{5}$$

3.1.2 LLMs-based recommendation models

BIGRec requires the historical sequence of user-item interactions to generate the next item by recommendation-specific instruction-tuning. For each user $u \in \mathcal{U}$, the historical sequence is denoted as S^u, defined as Equation 3, where $|S^u|$ represents the sequence length of user u.
Based on the historical sequence S^u and designed prompt $Pompt$, BIGRec generates the new item for user, denoted as Equation 6, where $S_{1:t}^u$ denotes the interaction sequence of user u from time 1 to time t, $Prompt$ denotes the specifically designed prompt for recommendation

instruction-tuning, ϕ denotes the parameters, g denotes the generation function, and $oracle$ denotes the embeddings of the outputs generated by the LLMs.

$$oracle = g_\phi(Prompt,\, S^u_{1:t}) \in \mathbb{R}^d \tag{6}$$

The $Prompt$ consists of three parts: an $instruction$ defining the recommendation task, an $input$ providing the historical sequence of user, and an $output$ representing the ground-truth item which is the last item of the historical interaction sequence. The example is demonstrated in Table 1.

The global ranking of BIGRec is achieved by calculating the L2 distance, defined as Equation 7, between actual items and the generated item. emb_i denotes the embedding of the i-th item and D_i denotes the L2 distance between emb_i and $oracle$.

$$D_i = \|emb_i - oracle\|_2 \tag{7}$$

LETTER uses LLM to generate the next item in the format of an identifier, based on the historical interaction sequence S^u of user u. It applies semantic regularization to make identifiers with hierarchical semantics, collaborative regularization to inject collaborative signals, and diversity regularization to reduce the code assignment bias.

The training loss of LETTER is defined as Equation 8, where L_{Sem} denotes the loss for semantic regularization, L_{CF} denotes the loss for collaborative regularization, L_{Div} denotes the loss for diversity regularization, and α and β are set to regulate the influence of collaborative and diversity regularization.

$$\mathcal{L}_{\mathsf{LETTER}} = \mathcal{L}_{\mathsf{Sem}} + \alpha\mathcal{L}_{\mathsf{CF}} + \beta\mathcal{L}_{\mathsf{Div}} \tag{8}$$

Given the historical interaction sequence S^u of user u, each item i is projected into the identifier $\tilde{i}$ of length L, shown as Equation 9. The user interaction sequence is projected into x, described as Equation 10. Then training data $\mathcal{D}$ is defined as Equation 11. y denotes the identifier of the next item to be predicted defined as Equation 12. Then LETTER is formulated as Equation 13, where θ denotes the parameters.

$$\tilde{i} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_L] \tag{9}$$

$$\mathbf{x} = \left[\tilde{i}_1, \tilde{i}_2, \ldots, \tilde{i}_M\right] \tag{10}$$

$$\mathcal{D} = \{(x, y)\} \tag{11}$$

$$y = \tilde{i}_{M+1} \tag{12}$$

$$p_\theta(y \mid x) \tag{13}$$

3.2 Model Architecture

In this research, we evaluate five different recommendation models on four datasets, ranging from NCF, a neural network–based model, to SASRec, which uses self-attention for sequential recommendation, to BERT4Rec with deep bidirectional self-attention. We also included BIGRec, a LLMs-based recommendation model emphasizing global ranking ability, and LETTER, a generative recommendation model which improves item tokenization from TIGER.

These five recommendation models are selected as the representative work from traditional recommendation models – both non-sequential and sequential settings – and LLMs-based recommendation systems. NCF resorts to neural networks to model interactions between user

and item features based on implicit feedback, which is an improvement of classic collaborative filtering. Sequential recommendation systems aim to capture the context of user actions. Traditional MCs-based approaches require sparse datasets. and RNNs-based approaches require dense datasets. SASRec leverages self-attention mechanism from Transformer to capture long-term semantics using relatively few recent actions, reducing the limitation of traditional MCs-based and RNNs-based methods. The limitation of SASRec is that it can only learn from left to right. BERT4Rec is proposed to address the limitations, using the deep bidirectional self-attention mechanism. For LLMs-based recommendation models, BIGRec converts interactions between user and item to natural language and explores powerful generative and natural language understanding ability of LLMs on recommendation tasks. Another important paradigm in LLMs-based recommendation systems is item tokenization. LETTER uses semantic regularization, collaborative regularization, and diversity regularization to inject hierarchical semantic information and collaborative signals into fair identifiers.
As LETTER is an enhancement of TIGER and originates from it. LETTER focuses on item tokenization and it is supposed to be instantiated on one generative rec model, for which we adopt TIGER, which is consistent with the original paper of LETTER. Therefore, in this section, the core mechanisms of six models, including TIGER, are outlined.

3.2.1 Neural collaborative filtering

NCF (Neural network-based Collaborative Filtering) [9] uses multi-layer perception (MLP) to learn the interaction from data to address the implicit feedback limitation.
For NCF general framework, the output of one layer is used as the input of next one. The input feature of the bottom input layer is the identity of users and items, which can be transformed into binarized sparse vector with one-hot encoding. NCF framework then can be easily extended to address the cold-start problem for the generic feature representation. The embedding layer projects them to user and item embeddings, the dense vectors. The multi-layer perceptron (MLP) architecture, neural collaborative filtering layers, then map the embeddings to prediction scores. The illustration of NCF framework is shown as Figure 1. The output layer predicts the scores after the training performed by minimizing the pointwise

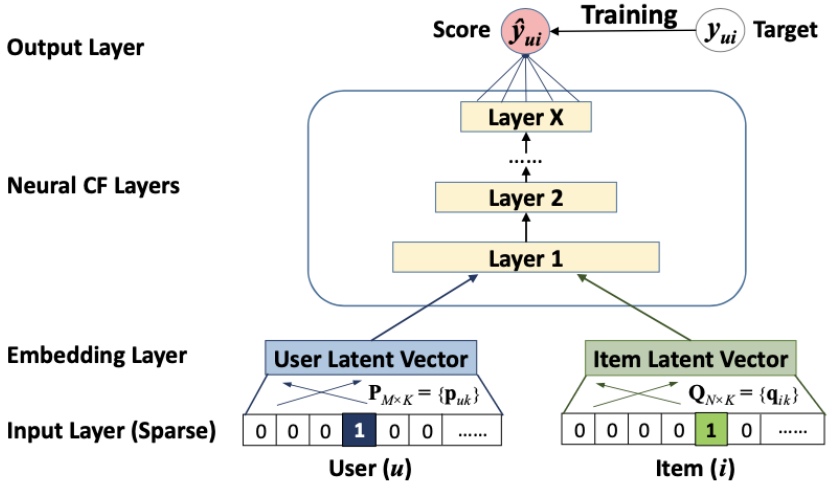


Figure 1: Neural Collaborative Filtering Framework[9]

loss. However, since implicit feedback is binarized 1 or 0, the value can be treated as a label

indicating the relevance or irrelevance. The prediction score then reflects the likelihood of an item being relevant to a user and constrained in the range [0,1] through the activation function. The recommendation task with implicit feedback then is converted to the binary classification task.
The conventional MF can be viewed as a special case of NCF framework, in which the identity function is used as the activation function of the output layer and the edge weights of the output layer are set to a uniform vector of ones. Meanwhile, if the activation function is set to a non-linear function, like the sigmoid function as Equation 14, where x denotes the innner product of the user embedding and item embedding. And the output layer weights are learned from data through log loss, then the Generalized Matrix Factorization (GMF) is defined.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{14}$$

As shown in Figure 1, the inputs of users and items are processed separately. Multi-Layer Perceptron (MLP) is implemented to learn the interaction between user and item latent features. The network structure of MLP is designed as tower pattern, which helps to learn more abstract features of data.
GMF applies a learned kernel to model latent feature interactions, while MLP uses a non-linear kernel to learn these interactions directly from data. Through sharing the same embedding layer and the combination of their interactions' outputs, the fusion model of GMF and MLP, NeuMF (Neural Matrix Factorization), is achieved, shown as Figure 2.

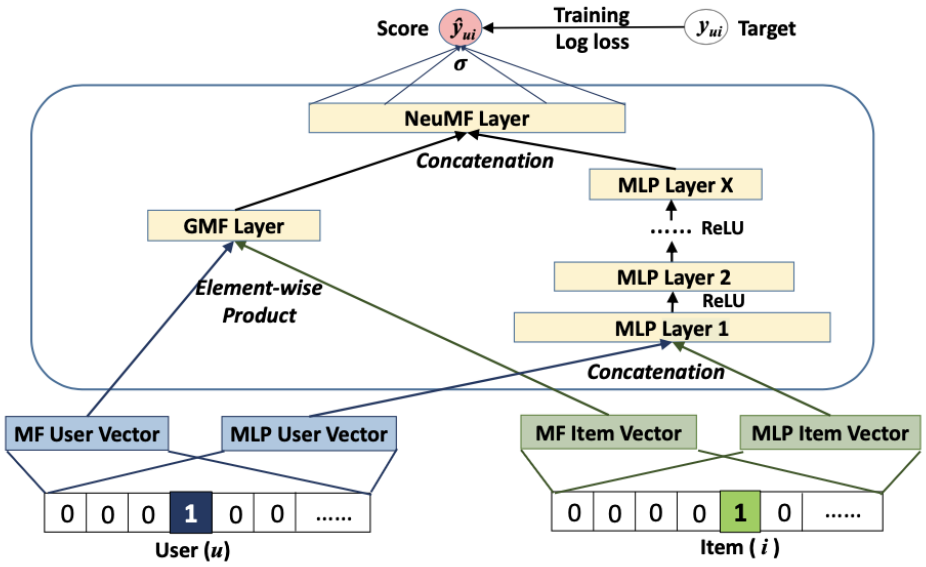


Figure 2: Neural Matrix Factorization Model [9]

3.2.2 Self-attentive sequential recommendation

Self-Attention based Sequential Recommendation model (SASRec) [14] applies self-attention mechanisms, suitable for parallel, sequential recommendation tasks. It predicts the next item based on the user's previous action sequence. The core components of this model include the embedding layer, self-attention blocks, and prediction later.

The user action sequence is represented as $S^u = (S_1^u, S_2^u, \ldots, S_{|S^u|}^u)$. For SASRec, at the time step t, the fixed-length sequence $(S_1^u, S_2^u, \ldots, S_{|S^u|-1}^u)$ is fed into the model as the training sequence to predict the next item $(S_2^u, S_3^u, \ldots, S_{|S^u|}^u)$. If the sequence length exceeds the maximumimal length n, then only the most recent n items are taken into account. If the sequence is shorter than the maximum length n, it is padded with zeros to reach the required length. Since SASRec applies the self-attention, rather than RNNs or CNNs, the learnable position embedding is injected into the input embedding.
The attention layer is employed to compute the weighted sum of all values.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}[14] \tag{15}$$

Equation 15 defines the scaled dot-product attention, where Q denotes the queries, K denotes the keys, V denotes the values, and $\sqrt{d}$ is the scale factor. Shown as Equation 15, the embedding is used as the input to the self-attention operations and is transformed into queries, keys, and values through linear projections learning asymmetric interactions.
The self-attention mechanism allows to use the information of the whole sequence. However, the causality must be considered. The model are only allowed to consider the first several items to predict the next item. To solve the challenge, all the links between Q_i and K_j, where $i < j$, must be forbidden.
Self-attention is the linear model, a point-wise two-layer feed-forward network is applied to inject the nonlinearity to the model:

$$\mathbf{F}_i = \text{FFN}(\mathbf{S}_i) = \text{ReLU}(\mathbf{S}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}[14] \tag{16}$$

where $\mathbf{S}_i$ denotes the i-th output of the self-attention layer, $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$ are $d \times d$ matrices, and $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$ are d-dimensional vectors. The interaction between S_i and S_j is forbidden to avoid information leakage.
To learn more complex information, the self-attention blocks are stacked. This complex structure causes several challenges, including overfitting, instability, and increased training time requirement. Therefore, residual connections to capture the low-layer features, layer normalization to stable the and accelerate the training, and dropout to reduce overfitting are applied. The relevance score of the next item is predicted based on the output of the final self-attention block, and recommendations are generated by ranking items according to these scores, where the higher scores indicate greater relevance. While using shared item embeddings helps reduce overfitting, it may limit the model's ability to capture asymmetric interactions. However, SASRec effectively overcomes this limitation by learning non-linear transformations earlier in the network.

3.2.3 Sequential recommendation with bidirectional encoder representations from Transformer

BERT4Rec(Sequential Recommendation with Bidirectional Encoder Representations from Transformer) is built on the self-attention layer. As illustrated in Figure 3, this model consists of L stacked bidirectional Transformer layers and the position information is updated iteratively. However, learning from both left and right sides of a sequence can lead to information leakage. To address this issue, BERT4Rec introduces Cloza task. In this step, the model randomly masks certain items in the sequence and learns to predict the corresponding IDs based on both left and right information. Since BERT4Rec is designed for recommendation tasks, the

last item in the sequence is supposed to be masked as the target of the recommendation tasks. This approach allows BERT4Rec to leverage bidirectional context while still performing effective next-item prediction as recommendation model.
The hidden representations h_i^l at each layer l for each position i are computed in parallel using Transformer layer, based on the input sequence of length t. For parallel computation, the hidden representations $h_i^l \in \mathbb{R}^d$ are stacked into the matrix $H^l \in \mathbb{R}^{t \times d}$.

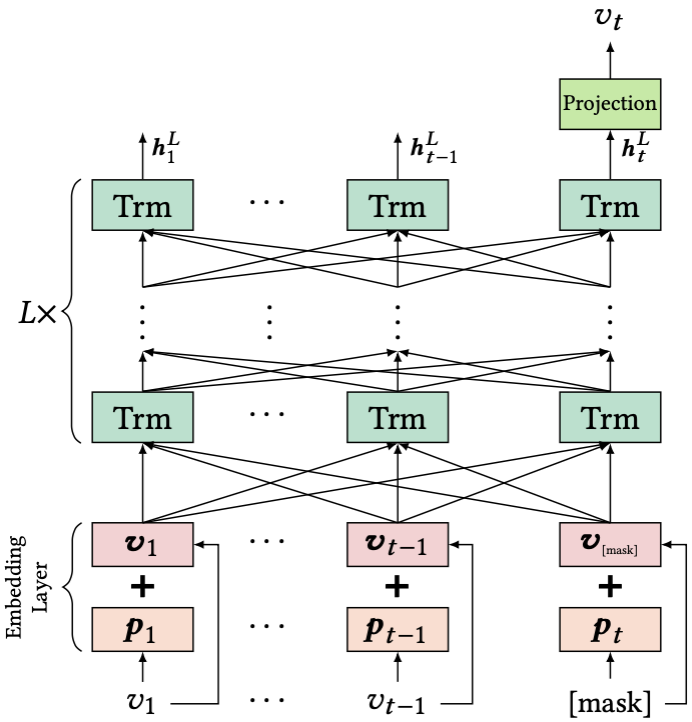


Figure 3: BERT4Rec Model Architecture [32]

The Transformer layer contains two sub-layers, as shown in Figure 4, including the Multi-Head Self-Attention using linear projection and Position-wise Feed-Forward Network injecting the nonlinearity into the model.

$$\mathsf{MH}(H^l) = [\mathsf{head}_1; \mathsf{head}_2; \ldots; \mathsf{head}_h]W^O[32] \tag{17}$$

$$\mathsf{head}_i = \mathsf{Attention}(H^l W_i^Q, H^l W_i^K, H^l W_i^V)[32] \tag{18}$$

Multi-head attention projects H^l into h sub-spaces as Equation 17 and apply h attention functions in parallel as Equation 18. $W_i^Q \in \mathbb{R}^{d \times d/h}$, $W_i^K \in \mathbb{R}^{d \times d/h}$, $W_i^V \in \mathbb{R}^{d \times d/h}$, and $W_i^O \in \mathbb{R}^{d \times d}$ are projections matrices. The attention function is defined as Equation 19. Q denotes query, K denotes key, V denotes value, and $\sqrt{d/h}$ denotes the temperature. Q, K, and V are learned from the same H^l, but with different learned projection matrices W_i^Q, W_i^K, and W_i^V.

$$\mathsf{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathsf{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d/h}}\right)\mathbf{V}[32] \tag{19}$$

$$\mathsf{FFN}(\mathbf{x}) = \mathsf{GELU}(\mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}[32] \tag{20}$$

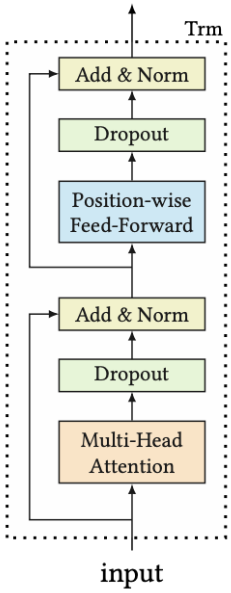


Figure 4: Transformer Layer [32]

$$\mathsf{GELU}(x) = x\,\Phi(x)[32] \tag{21}$$

Position-wise Feed-Forward Network is applied to the outputs of the self-attention sub-layer to inject the nonlinearity in two steps, as shown in Equation 20. The first transition is $\mathbf{x}\mathbf{W}^{(1)}+\mathbf{b}^{(1)}$, followed by Gaussian Error Linear Unit (GELU) activation represented as Equation 21. $\mathbf{W}^{(1)} \in \mathbb{R}^{d\times 4d}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{4d\times d}$, $\mathbf{b}^{(1)} \in \mathbb{R}^{4d}$, and $\mathbf{b}^{(2)} \in \mathbb{R}^{d}$ are learnable parameters. The nonlinearity is applied independently to each h. For H^l, the transition is represented as Equation 22.

$$\mathsf{PFFN}(H^l) = \left[\mathsf{FFN}(h_1^l)^\top; \ldots; \mathsf{FFN}(h_t^l)^\top\right]^\top [32] \tag{22}$$

Stacking multiple Transformer layers enables to capture more complicated information, but it also introduces some challenges such as increased training costs. To address this, residual connection, layer normalization, and dropout are incorporated. The learnable positional embeddings are injected to the input to preserve position information, and the input sequence length is fixed for consistencey. And the shared item embedding matrix is used to reduce overfitting and model size. The two-layer feed-forward network with GELU activation as Equation 21 is used to predict the distribution over target items.

$$P(v) = \mathrm{softmax}\left(\mathrm{GELU}\left(h_t^L W^P + b^P\right) E^\top + b^O\right)[32] \tag{23}$$

where h_t^L denotes the masked item after L layers, W^P denotes the learnable projection matrix, b^P, b^O are bias terms, and E is the embedding matrix.

3.2.4 Bi-step grounding paradigm for recommendation

BIGRec (Bi-step Grounding Paradigm for Recommendation) [3] is a two-step grounding framework designed to evaluate the comprehensive ranking capabilities of LLMs.

Given the vast language space created by the strong generative capacity of LLMs, BIGRec introduces two grounding steps: 1. Grounding LLMs from language space to the recommendation space for recommendation task. 2.Grounding from the recommendation space to the actual item space.
The language space refers to the set of all possible language sequences that LLMs could generate. This space is extremely large, making it impractical to generate recommendations directly from it. The recommendation space is a filtered subset of the language space, containing the entities satisfying the users' preference. However, these entities include both real and imaginary items. The third space, actual item space, contains only the real, valid items within the recommendation space. By grounding the model through these three spaces, BIGRec filters candidates layer by layer, ultimately selecting the target items from a vast number of possibilities.
The implementation primarily relies on two stages of instruction tuning. First, instruction tuning is performed on the alpaca self-instruct data using LLaMA (Large Language Model Meta AI) [35], enhancing the LLMs' understanding ability towards the instructions. Second, recommendation-specific instruction tuning, shown as Table 1, is applied to constrain the output of LLMs from the vast language space to the recommendation space.

Table 1: Example of the instruction-tuning data for the step of grounding to the space.

Instruction Input	
Instruction:	Given ten movies that the user watched recently, please recommend a new movie that the user likes to the user.
Input:	The user has watched the following movies before: "Traffic (2000)", "Ocean's Eleven (2001)", ... "Fargo (1996)" [3]
Instruction Output	
Output:	"Crouching Tiger, Hidden Dragon (Wu hu zang long) (2000)"

To ensure the the recommended items are real entities, the model compares the embeddings of the generated tokens and the embeddings of the actual items. The L2 distance representing the similarity is calculated, as Equation 24, where emb_i denotes the embedding of the i-th item, $oracle$ denotes the embedding of the outputs generated by the LLM, and D_i denotes the L2 distance between them.

$$D_i = \|\mathsf{emb}_i - \mathsf{oracle}\|_2[3] \tag{24}$$

Based on the similarity values, the generated tokens are ranked in ascending order of L2 distance to determine which ones correspind to real-work items.
In addition to validating the authenticity of items, it is also important to incorporate popularity and collaborative information. To inject the popularity, the popularity factor of each item is calculated and is normalized as Equation 25. $\mathcal{N}$ denotes the set of user-item interactions in the training data, $\mathcal{N}^j$ denotes the number of observed interactions for item j in $\mathcal{N}$, $\mathcal{I}$ denotes all items, C_i denotes the popularity of the i-th item, and P_i denotes the normalized value of

C_i.

$$\begin{cases} C_i = \dfrac{\mathcal{N}^i}{\sum_{j \in \mathcal{I}} \mathcal{N}^j}, \\ P_i = \dfrac{C_i - \min_{j \in \mathcal{I}}\{C_j\}}{\max_{j \in \mathcal{I}}\{C_j\} - \min_{j \in \mathcal{I}}\{C_j\}}, \end{cases} \text{[3]} \tag{25}$$

Then the L2 distance is re-weighted by popularity information, adjusted from Equation 24 to Equation 26, where $\hat{D}_i$ denotes the normalized D_i, and γ is the hyperparameter determining the influence of popularity.

$$\begin{cases} \hat{D}_i = \dfrac{D_i - \min_{j \in I}\{D_j\}}{\max_{j \in I}\{D_j\} - \min_{j \in I}\{D_j\}}, \\ \tilde{D}_i = \dfrac{\hat{D}_i}{(1 + P_i)^\gamma}, \end{cases} \text{[3]} \tag{26}$$

For collaborative filtering information, the prediction score predicted by collaborative filtering can be injected in the similar way as popularity factor in Equation 26. However, the increased volume of statistical information, popularity and collaborative information, doesn't improve the performance of BIGRec as substantial as traditional methods.

3.2.5 Transformer index for generative recommenders

TIGER (Transformer Index for GEnerative Recommenders) [24] is a generative retrieval model for sequential recommendation task. It introduces the concept of Semantic ID, which is a sequence of tokens generated from the item's content information. This Semantic ID is then used to train the generative recommendation model.

To generate Semantic ID based on item's content information, dense content embeddings are first generated using a pretrained text encoder, such as SentenceT5. Then, the embedding is quantized to generate a set of codewords/tokens. These tokens are ordered and are referred to as the Semantic IDs of items.

The residual quantum variable autoencoder (RQ-VAE) is used to quantize and to generate the Semantic IDs. The content information of items are denotes as x and the encoder is denoted as ξ. The latent representation is denoted as z, defined as Equation 27.

$$z := \mathcal{E}(x) \text{[24]} \tag{27}$$

The level is denoted as d. At the initial level, where $d = 0$, the initial residual is r_0, defined as Equation 28.

$$r_0 := z \text{[24]} \tag{28}$$

For each level d, the codebook is defined as Equation 29, where K denotes the codebook size.

$$C_d := \{e_k\}_{k=1}^{K} \text{[24]} \tag{29}$$

At zero-th level, the zero-th codeword is defined as Equation 30, which is the index of the closest embedding.

$$c_0 = \arg\min_i \|r_0 - e_k\| \text{[24]} \tag{30}$$

For each level d, c_d denotes the index of the closest embedding, defined as Equation 31. The closest embedding e_{c_d} is recorded.

$$c_d = \arg\min_i \|r_d - e_i\| [24] \tag{31}$$

For the residual of next level, $d+1$, it is defined as Equation 32.

$$r_{d+1} := r_d - e_{c_d} [24] \tag{32}$$

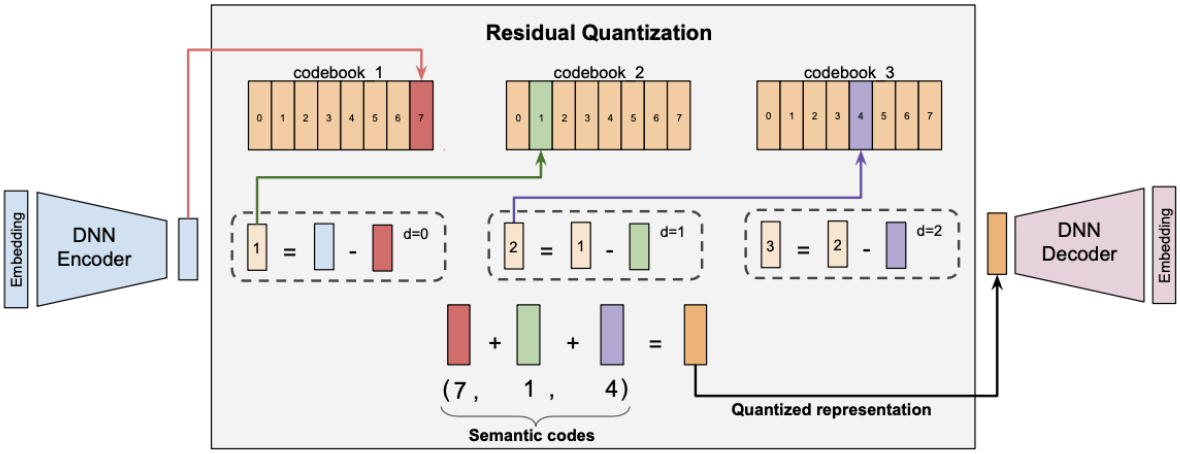


Figure 5: RQ-VAE [24]

This process of Equation 31 and Equation 32 is repeated until the Semantic ID is fully generated. The whole process is demonstrated as Figure 6. If the codebook has m levels, the process repeats m times to eventually obtain the complete Semantic ID, denoted as $(c_0, c_1, \ldots, c_{m-1})$. Then the quantized representation of $\hat{z}$, is defined as Equation 33. $\hat{z}$ is used by decoder to recreate x, and $\hat{x}$ is the output of the decoder.

$$\hat{z} := \sum_{d=0}^{m-1} e_{c_d} [24] \tag{33}$$

The loss of RQ-VAE is defined as Equation 34, where $\mathcal{L}_{\text{recon}}$ and $\mathcal{L}_{\text{rqvae}}$ is defined as Equation 35 and Equation 36, respectively.

$$\mathcal{L}(x) := \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{rqvae}} [24] \tag{34}$$

$$\mathcal{L}_{\text{recon}} := \|x - \hat{x}\|^2 [24] \tag{35}$$

$$\mathcal{L}_{\text{rqvae}} := \sum_{d=0}^{m-1} \|\text{sg}[r_d] - e_{c_i}\|^2 + \beta \|r_d - \text{sg}[e_{c_i}]\|^2 [24] \tag{36}$$

The Transformer model is then trained as the sequential recommendation model based on these Semantic IDs, $(c_0, c_1, \ldots, c_{m-1})$. The sequence of users is denoted as $(\text{item}_1, \ldots, \text{item}_n)$,

the model is supposed to predict the next item. The Semantic ID for item$_i$ is denoted as $(c_{i,0}, c_{i,1}, \ldots, c_{i,m-1})$. Then the sequence of user is converted to

$$(c_{1,0}, \ldots, c_{1,m-1},\ c_{2,0}, \ldots, c_{2,m-1},\ \ldots,\ c_{n,0}, \ldots, c_{n,m-1})$$

The next item predicted by the sequence-to-sequence model is denoted as

$$(c_{n+1,0}, c_{n+1,1}, \ldots, c_{n+1,m-1})$$

3.2.6 Learnable item tokenization for generative recommendation

LETTER (a LEarnable Tokenizer for generaTivE Recommendation) [38]is proposed to meet the requirements of identifiers by integrating hierarchical semantics, collaborative signals, and code code assignment diversity.

This model is developed based on RQ-VAE [16] and introduces semantic regularization to generate identifier with hierarchical semantic, in two steps. The first step is semantic embedding extraction. Content information, such as titles and genres from MovieLens 1M "movies.dat" file, and titles and descriptions from the Amazon meta data, is firstly used to extract the semantic embedding s. This embedding is then compressed into $z = Encoder(s), s \in \mathbb{R}^d$. The second step is semantic embedding quantization. Embedding z is then quantized into the code sequence using L codebooks, where L represents the length of the identifier. The loss for semantic regularzation is defined as Equation 37, where l denotes the level, $C_l = \{\mathbf{e}_i\}_{i=1}^{N}$ denotes the codebook, $\mathbf{e}_i$ is the learable code embedding, N denotes the size of codebook, c_l denotes the code index for the level l, r_{l-1} denotes the residual for level $l-1$, and $\hat{s}$ denotes the recreated semantic embedding.

$$\begin{cases} \mathcal{L}_{\text{Sem}} = \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{RQ-VAE}}, \\ \mathcal{L}_{\text{Recon}} = \|\mathbf{s} - \hat{\mathbf{s}}\|^2, \quad \text{[38]} \\ \mathcal{L}_{\text{RQ-VAE}} = \sum_{l=1}^{L} \left\|\text{sg}\left[\mathbf{r}_{l-1}\right] - \mathbf{e}_{c_l}\right\|^2 + \mu \left\|\mathbf{r}_{l-1} - \text{sg}\left[\mathbf{e}_{c_l}\right]\right\|^2, \end{cases} \tag{37}$$

Collaborative regularization is introduced to make items with similar collaborative information to have similar code sequences. Firstly, the CF (collaborative filtering) embeddings of items are obtained by well-trained CF recommendation model. Then these embeddings are used to be aligned with the quantized embeddings from semantic regularization to inject collaborative signals. The CF loss is defined as Equation 38, where h denotes CF embedding, $< a, b >$ denotes the inner product between a and b, and B denotes the batch size.

$$\mathcal{L}_{\text{CF}} = -\frac{1}{B} \sum_{i=1}^{B} \frac{\exp\left(\langle \hat{z}_i, h_i \rangle\right)}{\sum_{j=1}^{B} \exp\left(\langle \hat{z}_i, h_j \rangle\right)} \text{[38]} \tag{38}$$

Diversity regularization is introduced to mitigate the code generation bias. The code embeddings are cluster into K groups by using constrained K-means [5]. The distribution is then determined based on the distance among the code embeddings, encouraging diversity in the generated codes. The loss of diversity regularization is defined as Equation 39, where $e^i_{c_l}$ denotes the nearest code embedding of item i, e_+ denotes code embedding of a randomly selected sample from c_l, and $e_j, \quad j \in \{1, \ldots, N\} \setminus \{c_l\}$ denotes all code embeddings except e_{c_l}

$$\mathcal{L}_{\text{Div}} = -\frac{1}{B} \sum_{i=1}^{B} \frac{\exp\left(\langle e^i_{c_l}, e_+ \rangle\right)}{\sum_{j=1}^{N-1} \exp\left(\langle e^i_{c_l}, e_j \rangle\right)} \text{[38]} \tag{39}$$

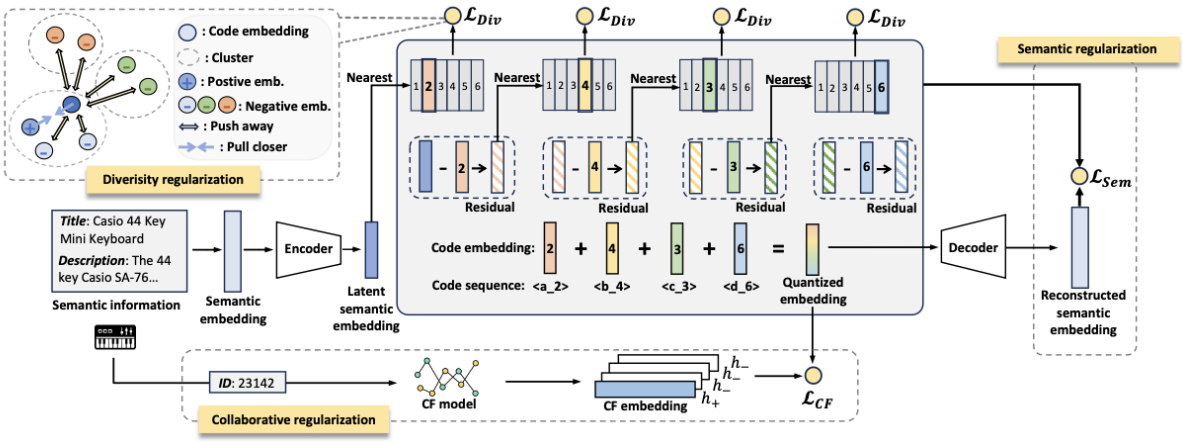


Figure 6: LETTER [38]

In conclusion, the training loss for LETTER is defined as Equation 40, where α and β are set to regulate the influence of collaborative and diversity regularization. The whole process of LETTER is demonstrated as Figure 6.

$$\mathcal{L}_{\text{LETTER}} = \mathcal{L}_{\text{Sem}} + \alpha\mathcal{L}_{\text{CF}} + \beta\mathcal{L}_{\text{Div}}, [38] \tag{40}$$

4 Data

4.1 Data sources and dataset scale

In this research, we use four datasets from two different sources. The first one is MovieLens 1M Dataset, a well-established benchmark provided by GroupLens Research [1]. It contains 1,000,209 anonymous ratings for approximately 3,900 movies, rated by 6,040 users from approximately. The dataset is released in February 2023.

The other datasets are from the Amazon Review Data (2018) and Amazon Review Data (2014), which contain two main components: a reviews file and a product metadata file. In our study, we focus on three datasets: Amazon 2018 Video Games category, Amazon 2018 Luxury Beauty category, and Amazon 2014 Beauty category. For all three categories, we use the 5-core subset for reviews, meaning that each remaining user and item has at least five reviews. Specifically, the Amazon 2018 Luxury Beauty dataset contains 34,278 reviews and metadata for 12,308 products. The Amazon 2018 Video Games dataset is larger, containing 497,577 5-core reviews and metadata for 84,893 products. The Amazon 2014 Beauty contains 198,502 reviews and 259,204 products metadata.

4.2 Structure, schema and examples

4.2.1 MovieLens 1M

The MovieLens 1M Dataset contains three files, including "ratings.dat", "users.dat", and "movies.dat", each capturing a different aspect of the user–item interaction.

[1] (http://www.grouplens.org/)

(a) `ratings.dat` Schema

Field Name	Range	Description	Example
UserID	Range between 1 and 6040	Unique user identifier	1
MovieID	Range between 1 and 3952	Unique movie identifier	1193
Rating	5-star scale	User's rating for the movie	5
Timestamp	Seconds since 1970-01-01	Time of ratings	978300760

(b) `movies.dat` Schema

Field Name	Description	Example
MovieID	Unique movie identifier	1
Title	Movie title with release year	Toy Story (1995)
Genres	Fixed set of categories	Animation|Children's|Comedy

(c) `users.dat` Schema

Field Name	Description	Example
UserID	Unique user identifier	1
Gender	User gender	F
Age	Age group category ID	1
Occupation	Occupation category ID	10
Zip-code	User's postal code	48067

Table 2: Data structures of the three files in MovieLens 1M: `ratings.dat`, `movies.dat`, and `users.dat`.

All ratings and interactions are contained in the file "ratings.dat" and are in the following format:

$$UserID :: MovieID :: Rating :: Timestamp$$

Each user in the dataset has provided at least 20 ratings, ensuring a minimum level of engagement. All ratings are whole-star values on a 5-point scale. The Timestamp field represents the time of the rating in seconds since the Unix epoch (1970-01-01 00:00:00 UTC).
The "movies.dat" file provides information about each movie, using the format:

$$MovieID :: Title :: Genres$$

Titles are the same as the ones provided by the IMDB, including the year of release. Genres are pipe-separated and chosen from predefined set. Some MovieIDs may not correspond to actual movies due to duplicate or test entries.
The "users.dat" file contains demographic information about each user in the format:

$$UserID :: Gender :: Age :: Occupation :: Zip - code$$

The detailed schema of the three files of MovieLens 1M, including field names, ranges, descriptions, and example values, is shown in Table 2.

4.2.2 Amazon Review Data

Amazon Review Data (2018 version and 2014 version) include the review data and the metadata for items. The review data is stored in a JSON format, with one review per line. The

metadata includes information such as product descriptions, prices, sales rank, brand, and co-purchasing links. The detailed schema of the Amazon Review Data including reviews file and meta file, is shown in Table 3 and Table 4. The content of Amazon 2018 and Amazon 2014 is slightly different. In Table 3 and Table 4, the fields indicating the version of dataset appear only in this dataset.

Table 3: Field descriptions and examples from Amazon Review Data

Field Name	Description	Example
`overall`	Rating score (1–5)	5.0
`verified`	Whether the purchase was verified	true
`reviewTime`	Date of the review (raw format)	`"01 5, 2018"`
`reviewerID`	Unique ID of the reviewer	`"A2HOI48JK8838M"`
`asin`	Unique product ID	`"B00004U9V2"`
`style (2018)`	Product attributes (e.g., size, color)	`"Size:" : "0.9 oz."`
`reviewerName`	Name of the reviewer	`"DB"`
`reviewText`	Full content of the review	`"This handcream has a..."`
`summary`	Short summary/title of the review	`"Beautiful Fragrance"`
`unixReviewTime`	Review timestamp (Unix time)	1515110400
`helpful(2014)`	helpfulness rating of the review	2/3

Table 4: Field descriptions and examples – Amazon metadata

Field Name	Description	Example
`category`	List of categories the product belongs to	`["Video Games", "PC", "Games"]`
`description`	Description of the product	`"This software is BRAND NEW..."`
`title`	Name of the product	`"Ice Cream Truck"`
`brand`	Brand name	`"Sunburtst"`
`feature`	Bullet-point features of the product	`["8 MB RAM...", "SVGA..."]`
`also_view/buy`	Related products also viewed	`["B00005LBVS", "B002CMU748", ...]`
`main_cat`	Main category	`"Video Games"`
`price`	Price in USD (at time of crawl)	`"$5.69"`
`asin`	ID of the product	`"0439342260"`
`imageURL`	URL of the product image	`["https://...jpg"]`
`imageURLHighRes`	URL of high-resolution image	`["https://...jpg"]`

4.3 Descriptive statistics

MovieLens 1M is the densest dataset among four datasets. Even the interactions of user and item still follow the long-tail distribution, the proportion of popular item is more than 60% and all the users are highly active. This characteristics helps to explore new items when the cutoff is set to higher ranks. But the limitation of MovieLens 1M is that the order of data doesn't reflect the actual order of users' behaviors in real life.

All three Amazon Review datasets show evident long-tail effect, meaning that most items have very few interactions. Amazon 2018 Video Games has the largest number of users and is the sparsest among four datasets. Amazon 2018 Luxury Beauty is relatively small in user size but

Table 5: Density of MovieLens 1M

Users	Items	Interactions	Density
6,040	3,883	1,000,209	4.27%

Table 6: Density of Amazon Review Datasets based on meta items

Dataset	Users	Items (Meta)	Interactions	Density
Amazon 2018 Video Games	55,223	84,893	497,577	0.0106%
Amazon 2018 Luxury Beauty	3,811	12,308	34,278	0.0730%
Amazon 2014 Beauty	22,363	259,204	198,502	0.0034%

show higher density. Item interactions in this dataset are highly concentrated within a small proportion of items, while it has a high proportion of cold-start items. This interaction pattern reveals strong proportion bias and imbalance in Amazon 2018 Luxury Beauty. The scale of Amazon 2014 Beauty medium and density is also between Amazon 2018 Video Games and Amazon 2018 Luxury Beauty. It has no cold-start items and the distribution of interactions is relatively balanced. In addition, the item titles from Amazon 2018 Luxury Beauty and Amazon 2014 Beauty are more detailed and complicated than these from Amazon 2018 Video Games. The descriptions of items from Amazon 2018 Luxury Beauty and Amazon 2014 Beauty also more informative.

4.3.1 MovieLens 1M

The MovieLens 1M dataset contains 1,000,209 reviews, which translates to a density of roughly 4.27% and a sparsity of about 95.73%, shown as Table 5.

User-centric

In the MovieLens 1M dataset, each user has, on average, 165.6 interactions, with a median of 96 and a standard deviation of 192.75. Every user has contributed at least 20 ratings.

The users-centric interactions distribution for MovieLens 1M is illustrated as histogram Figure 20.

The "users.dat" file in the MovieLens 1M dataset contains each user's demographic details, specifically their gender, age, and occupation. Gender is denoted as "M" for male and "F" for female. Age and Occupation are selected from predefined categories and represented as categorical codes. The distributions of these attributes are shown in the Figures 21.

The "movies.dat" file contains the genre of the movies and the distribution is illustrated as Figure 22.

Item-centric

In the MovieLens 1M dataset, each item has, on average, 269.89 interactions, with a median of 123.5 and a standard deviation of 384.05. Items that have been rated more than 50 times are defined as popular items, while those with less than 5 ratings are considered cold-start items. In total, there are 2,514 popular items, approximately 64.74% of all interacted items. Meanwhile, 290 items fall into the cold-start items category, accounting for about 7.47%.

The items-centric interactions distribution for MovieLens 1M is illustrated as long-tail Figure 23.

Ratings

Figure 24 illustrates the rating rating statistics for MovieLens 1M. Figures 25 shows the rating statistics for Luxury Beauty and Video Games from Amazon Review Data (2018). For each

Table 7: Density of Amazon Review Datasets based on interacted items

Dataset	Users	Items (Interacted)	Interactions	Density
Amazon 2018 Video Games	55,223	17,408	497,577	0.0518%
Amazon 2018 Luxury Beauty	3,811	1,581	34,278	0.5689%
Amazon 2014 Beauty	22,363	12,101	198,502	0.0734%

Table 8: User and item interaction statistics on MovieLens 1M

Metric	User	Item (Movie)
Mean Interactions	165.60	269.89
Median Interactions	96.00	123.50
Std. Dev. Interactions	192.75	384.05
Active Users (≥ 20)	6040	–
Active User Ratio	1.0000	–
Popular Items (≥ 50)	–	2514
Popular Item Ratio	–	0.6474
Cold-Start Items (< 5)	–	290
Cold-Start Item Ratio	–	0.0747

dataset, the left subplot shows the overall distribution of rating scores, while the right subplot shows the yearly trend of the number of ratings over time.

4.3.2 Amazon Review Data

The Luxury Beauty category in the 2018 Amazon Review data comprises 34,278 reviews from 3,811 users. Although its metadata file lists 12,308 products resulting in a sparsity of approximately 98.94%, only 1,581 items actually have any user interactions.
In Amazon 2018 Video Games Review dataset, there are 497,577 ratings from 55,223 users. The sparsity is 94.82%. Same as Luxury Beauty category, the value is calculated with the total number of items listed in meta file, and the there are 17,408 items actually having interactions. Amazon 2014 Beauty contains 198,502 reviews from 22,363 users, with the sparsity about 99.997%. The value is calculated using all items listed in meta file. The sparsity becomes 99.93% when calculated based on the interacted items.
The density of Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty are shown as Table 6 and 7.
User-centric
In the Amazon 2018 Luxury Beauty dataset, user has on average 8.98 interactions, with a median of 7 and a standard deviation of 8.96. In the Amazon 2018 Video Games dataset, has on average 9.01 interactions, with a median of 6 and a standard deviation of 10.7. In Amazon 2014 Beauty, user has about 8.88 interactions on average, with a median of 6 and a standard deviation of 8.16. Each user has provided at least 5 ratings. The users-centric interactions distribution for Amazon 2018 Luxury Beauty, Amazon 2018 Video Games, and Amazon 2014 are illustrated respectively as long-tail curve and histogram, shown as Figure 26.
Item-centric
In Amazon 2018 Luxury Beauty dataset, each item has, on average, 21.68 interactions, with a median of 13 and a standard deviation of 57.36. Items that have been rated more than 50

Table 9: User and item interaction statistics on Amazon 2018 Luxury Beauty and Amazon 2018 Video Games

Metric	Amazon 2018 Luxury Beauty		Amazon 2018 Video Games	
	User	**Item**	**User**	**Item**
Mean Interactions	8.98	21.68	9.01	28.58
Median Interactions	7.00	13.00	6.00	13.00
Std. Dev. Interactions	8.96	57.36	10.70	52.34
Active Users (≥ 5)	3811	–	55220	–
Active User Ratio	0.9979	–	0.9999	–
Popular Items (≥ 50)	–	65	–	2257
Popular Item Ratio	–	0.0411	–	0.1297
Cold-Start Items (< 5)	–	143	–	114
Cold-Start Item Ratio	–	0.0904	–	0.0065

Table 10: User and Item Interaction Statistics on Amazon 2014 Beauty

Metric	**User**	**Item**
Mean Interactions	8.88	16.40
Median Interactions	6.00	9.00
Std. Dev. Interactions	8.16	23.61
Active Users ($\geq$)	22 363	–
Active User Ratio	1.0000	–
Popular Items (≥ 50)	–	742
Popular Item Ratio	–	0.0613
Cold-Start Items (< 5)	–	0
Cold-Start Item Ratio	–	0.000

times are defined as popular items, while those with less than 5 ratings are considered cold-start items. In total, there are 65 popular items, approximately 4.11% of all interacted items. Meanwhile, 143 items fall into the cold-start items category, accounting for about 9.04%.
In Amazon 2018 Video Games dataset, each item has, on average, 28.85 interactions, with a median of 13 and a standard deviation of 52.34. Items that have been rated more than 50 times are defined as popular items, while those with less than 5 ratings are considered cold-start items. In total, there are 2257 popular items, approximately 12.97% of all interacted items. Meanwhile, 114 items fall into the cold-start items category, accounting for about 0.65%.
In Amazon 2014 Beauty dataset, item has about 16.4 interactions on average, with a median of 9 and a standard deviation of about 23.61. There are 742 popular items, about 6.13% of all interacted items. And there is 0 cold-start items in this dataset.
The item-centric interactions distribution for Amazon 2018 Luxury Beauty, Amazon 2018 Video Games and Amazon 2014 Beauty is illustrated respectively as long-tail, shown as Figure 27.

5 Experimental Methodology

5.1 Model-specific input requirements

Each of the five models discussed in Section 3.2 has its own specific requirements. To ensure a fair comparison of the five models' performance, we use the hyper-parameter settings provided in their original papers as our default settings.

Prior to model training, all data are preprocessed to satisfy the different input requirements of each model. For NCF, the data must be split into three separate files: training data, testing data, and negative candidates. The training and testing files share the same format: each line contains user ID, item ID, rating, and timestamp, separated by spaces. The negative candidates file serves as the candidate set for ranking evaluation. It starts with the user ID and the item ID from the corresponding user's test data, followed by that user's negative samples. All entries are separated by spaces, maintaining the same train-test split structure.

The input files of SASRec and BERT4Rec follow the same format requirements, each line containing a user ID and an item ID, separated by a space.

The instruction-tuning for BIGRec consists of instruction input and a corresponding instruction output. The instruction input includes both the instruction itself and input shown as Table 1. As a result, the input files are in JSON format, containing the fields: instruction, input, and output as required. For the instruction part, it demonstrates the task prompt as "Given a list of movies/video games/beauty products the user has watched/played/used before, please recommend a new movie/video game/beauty product that the user likes to the user." The input part contains ten items the user has previously interacted with, while the output part specifies the target item to be recommended. All the interactions are sorted by the timestamp globally and the training, validation, and testing sets are split in a ratio 8:1:1 based on the chronological order. Considering BIGRec is based on LLMs and the constraints of computational cost and time cost, the validation and testing sets are each limited to 5000 samples. If the original split contains more than 5000 samples, 5000 are randomly sampled with the fixed seed 42. Otherwise, the full set is retained without modification.

LETTER emphasis on effective item tokenization and then is instantiated on generative recommendation model TIGER. The input files for item tokenization component contain the interaction file and item file, both in JSON format. The item file includes the semantic information, such as the title and the description of items, depending on specific datasets. This information is used to generate the item semantic embeddings using the base model LLaMA-7B. The interaction file includes the user's historical interactions. In addition, the 32-dimensional item embeddings generated by SASRec are also obtained to incorporate collaborative filtering information. Combining all the information, item tokenization is performed using the 4-level codebooks to generate the final index file in JSON format, which then serves as the input to TIGER.

5.2 Data consistency and evaluation protocol

Our research aims to conduct the comprehensive analysis and comparisons among five recommendation models. To ensure fairness of comparisons and data consistency, standardized data preprocessing and a unified protocol are essential.

As a result, we establish a standardized workflow based on the training strategy and full ranking evaluation approach of BIGRec, which incorporates the specific input requirements of

each model.
In the original implementation of NCF, SASRec, BERT4Rec, and LETTER, the leave-one-out strategy is used. The user's last interaction is served as the target item, and the second most recent for validation. The remaining interactions are used for training. The splitting is performed without global timestamp sorting, only relying on each user's local chronological order. This may lead to inconsistency in temporal context across all users, where interaction from different time periods are mixed in training, validation, and testing. And the candidate set is formed by randomly sampling a number of items that the user has not interacted with. The ground truth item is then ranked among these sampled items to evaluate the model's performance. This doesn't align with real-world application scenarios and could lead to information leakage.
Rather than leave-one-out strategy and negative sampling, we adopt full ranking strategy to evaluate the performance of all models. For each dataset, we sort all interactions globally by timestamp and split the training, validation, and testing sets with 8:1:1 ratio. Missing data, referring to cases where users have insufficient interactions, are filtered out during the process. For example, some users may appear in only one or two parts of training, validation, or testing sets, rather than having interactions in all three sets. For each dataset, the scale of training, validation, and test sets is shown as Table 11. To ensure fairness of comparison and to take the resource limitations of LLMs-based recommendation systems, we apply the same sampling strategy for testing set for all models. If the original split contains more than 5000 samples, 5000 are randomly sampled with the fixed seed 42.

Table 11: Scale of interactions in training, validation, and test sets for each dataset

Dataset	**Interactions**	**Training**	**Validation**	**Testing**
MovieLens 1M	1,000,209	800,167	100,021	100,021
Amazon 2018 LuxuryBeauty	34,278	27,422	3,428	3,428
Amazon 2018 VideoGame	497,577	398,601	49,488	49,488
Amazon 2014 Beauty	198,502	158,802	19,850	19,850

For each test user, the target item is ranked against all other items the user has not interacted with. This approach provides a comprehensive assessment of the ranking ability of each model. As a result, the candidate sets is all the items the user has not interacted with. The user may have several target items in the test set. During evaluation, each target item is ranked separately against the candidate set.

Table 12: Candidate set sizes used before switching to full ranking for each model

Model	**Splitting Strategy**	**Candidate Set Size**
NCF	Leave-one-out	100
SASRec	Leave-one-out	101
BERT4Rec	Leave-one-out	101
BIGRec	8:1:1	full ranking
LETTER	Leave-one-out	beam search

To evaluate the performance of the recommendation models, especially their ranking ability, we use the widely-used top-K ranking metrics: Hit Ratio (HR@K) and Normalized Discounted

Cumulative Gain (NDCG@k). HR@K, defined as Equation 41, measures whether the target item is in the top-K recommended items. The final value of HR@K is averaged over each test item of all users in the testing set. A higher value indicates the model has better capacity of retrieving relevant items within the top-K list.

$$\text{HR@K} = \begin{cases} 1, & \text{if the target item is in the top-K list} \\ 0, & \text{otherwise} \end{cases} \tag{41}$$

NDCG@K, as defined in Equation 43, is the normalized DCG by ideal DCG(IDCG) as a reference. IDCG represents the ideally ranked list.

$$\text{DCG@K} = r_1 + \sum_{i=2}^{|K|} \frac{r_i}{\log_2(i+1)} \tag{42}$$

$$\text{NDCG@K} = \frac{DCG(K)}{IDCG} \tag{43}$$

DCG@K is defined as Equation 42, where r_i denotes the relevance of the item at position i. Unlike HR@K only check whether the target item is retrieved, NDCG@K also considers its position in the ranked list. This makes it a more comprehensive evaluation of ranking quality. A higher NDCG@K value indicates that the target item is ranked closer to the top of the list, reflecting better model ranking performance.

6 Results and analysis

This section provides an analysis of the experimental results of five recommendation models – NCF, SASRec, BERT4Rec, BIGRec, LETTER – across four datasets, MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, Amazon 2014 Beauty.

Section 6.1 starts with the analysis of performance of five models on the relatively neutral dataset, Amazon 2018 Video Games. Inspired with this case study, we also conduct the case study of performance of LETTER, BERT4Rec, SASRec, and NCF across four datasets. Based on the features of these models, we analyze the potential reasons why the model exhibits different performance across different datasets.

Section 6.2 shift the focus to the comparative analysis among sequential recommendation systems. We start with the case study of BIGRec, analyzing its performance across four datasets. Based on these observations, we then introduce comparison between BIGRec and BERT4Rec and comparison between BIGRec and LETTER. Moreover, we find that LETTER and SASRec show consistent performance patterns across datasets. Based on it, we discuss the potential influence of collaborative filtering embedding from SASRec to LETTER.

6.1 Overall performance of five recommendation models across four datasets

The overall performance of five recommendation models, including NCF, SASRec, BERT4Rec, BIGRec, and LETTER, is summarized in Table 13 and 14. Table 13 report the Hit Rate (HR) and Table 14 report the Normalized Discounted Cumulative Gain (NDCG), evaluated at cut-off values K=1, 3, 5, 10, and 20. The best result for each metric of each model on each dataset is highlighted in bold.

In this part, we analyze and discuss the performance of these five recommendation models across four datasets. The analysis focuses on how each model performs under varying data characteristics and domains, providing insights into their generalization and effectiveness in different recommendation scenarios.

Table 13: HR performance of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on four datasets: MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, Amazon 2014 Beauty (k = 1, 3, 5, 10, 20).

Dataset	Model	@1	@3	@5	@10	@20
MovieLens 1M	NCF	0.0094	0.0206	0.0302	0.0498	0.0830
	SASRec	0.0020	0.0195	0.0223	0.0349	0.0885
	BERT4Rec	**0.0136**	**0.0288**	**0.0449**	**0.0796**	**0.1338**
	BIGRec	0.0014	0.0026	0.0036	0.0044	0.0066
	LETTER	0.0061	0.0267	0.0424	0.0741	0.1238
Amazon 2018 Video Games	NCF	0.0017	0.0060	0.0145	0.0280	0.0517
	SASRec	0.0072	0.0120	0.0160	0.0266	0.0424
	BERT4Rec	0.0131	**0.0317**	**0.0468**	**0.0695**	**0.1052**
	BIGRec	**0.0142**	0.0230	0.0270	0.0334	0.0454
	LETTER	0.0039	0.0049	0.0088	0.0176	0.0313
Amazon 2018 Luxury Beauty	NCF	0.0020	0.0195	0.0223	0.0349	0.0885
	SASRec	0.0146	0.0249	0.0370	0.0580	0.0832
	BERT4Rec	0.0012	0.0070	0.0099	0.0228	0.0380
	BIGRec	0.1239	0.1413	0.1432	0.1490	0.1665
	LETTER	**0.1874**	**0.3411**	**0.3602**	**0.3809**	**0.3964**
Amazon 2014 Beauty	NCF	**0.0055**	**0.0131**	**0.0188**	**0.0278**	0.0372
	SASRec	0.0034	0.0096	0.0155	0.0261	**0.0413**
	BERT4Rec	0.0027	0.0067	0.0101	0.0150	0.0227
	BIGRec	0.0036	0.0068	0.0090	0.0144	0.0234
	LETTER	0.0020	0.0088	0.0147	0.0254	0.0342

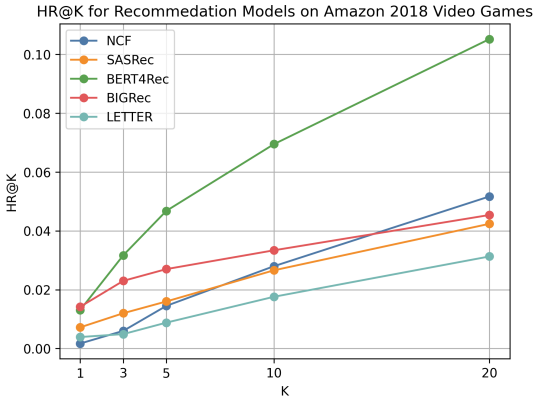
6.1.1 Case study: performance of recommendation models on Amazon 2018 Video Games dataset

On the Amazon 2018 Video Games dataset, BIGRec achieves the best performance with an HR@1 0f 0.0142, slightly outperforming BERT4Rec, HR@1 of 0.0131. As the relevance is binary in all five models, according to Equation 43,when k is set to 1,

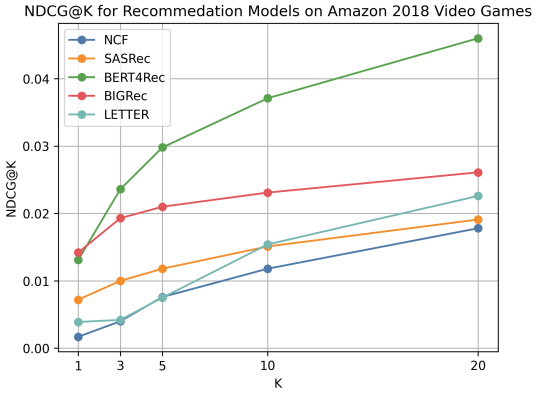
$$\text{NDCG@1} = \frac{\text{DCG@1}}{\text{IDCG@1}} \quad \text{where} \quad \text{DCG@1} = \frac{\text{r}_1}{\log_2(1+1)} = \text{r}_1$$

In this situation, the values of HR and NDCG are the same. This suggests that, for this dataset, both models are strong at placing the ground truth item at the very top of the ranked recommendation list.

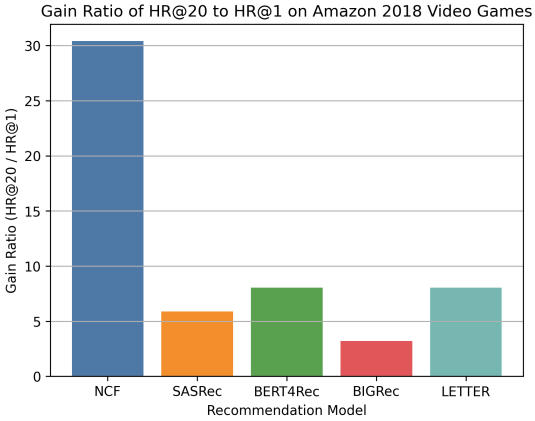
NCF performs worst on Amazon 2018 Video Games with an HR@1 of 0.0017. Meanwhile, according to Figure 7b, NCF consistently yields the lowest NDCG scores across all values



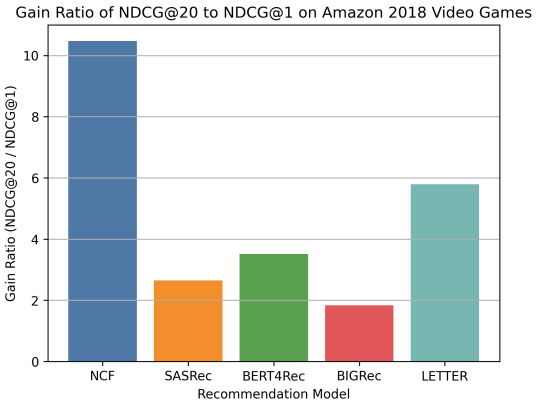
(a) HR@K of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Video Games



(b) NDCG@K of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Video Games



(c) Gain ratio (HR@20/HR@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Video Games



(d) Gain ratio (NDCG@20/NDCG@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Video Games

Figure 7: HR@K (K=1, 3, 5, 10, 20), NDCG@K (K=1, 3, 5, 10, 20), gain ratio (HR@20/HR@1), and gain ratio (NDCG@20/NDCG@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Video Games

Table 14: NDCG performance of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on four datasets: MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, Amazon 2014 Beauty (k = 1, 3, 5, 10, 20).

Dataset	Model	@1	@3	@5	@10	@20
MovieLens 1M	NCF	0.0094	0.0158	0.0198	0.0260	0.0343
	SASRec	0.0020	0.0125	0.0137	0.0176	0.0307
	BERT4Rec	**0.0136**	0.0224	0.0290	0.0404	0.0539
	BIGRec	0.0014	0.0021	0.0025	0.0028	0.0033
	LETTER	0.0061	**0.0252**	**0.0410**	**0.0618**	**0.0869**
Amazon 2018 Video Games	NCF	0.0017	0.0040	0.0076	0.0118	0.0178
	SASRec	0.0072	0.0100	0.0118	0.0151	0.0191
	BERT4Rec	0.0131	**0.0236**	**0.0298**	**0.0371**	**0.0460**
	BIGRec	**0.0142**	0.0193	0.0210	0.0231	0.0261
	LETTER	0.0039	0.0042	0.0075	0.0154	0.0226
Amazon 2018 Luxury Beauty	NCF	0.0020	0.0125	0.0136	0.0176	0.0307
	SASRec	0.0146	0.0204	0.0254	0.0322	0.0385
	BERT4Rec	0.0012	0.0044	0.0056	0.0097	0.0136
	BIGRec	0.1239	0.1342	0.1351	0.1370	0.1413
	LETTER	**0.1874**	**0.2258**	**0.2474**	**0.2767**	**0.3067**
Amazon 2014 Beauty	NCF	**0.0055**	**0.0099**	0.0122	0.0151	0.0175
	SASRec	0.0034	0.0070	0.0095	0.0129	0.0167
	BERT4Rec	0.0027	0.0049	0.0064	0.0080	0.0099
	BIGRec	0.0036	0.0055	0.0064	0.0081	0.0104
	LETTER	0.0020	0.0085	**0.0132**	**0.0205**	**0.0255**

among the five models, indicating comparatively worse ranking performance on Amazon 2018 Video Games. The gain ratio of NCF are shown in Figure 7c and 7d. These figures clearly demonstrate the growth of all five models on Amazon 2018 Video Games dataset from K=1 to K=20, in terms of both HR and NDCG metrics. It can be observed that NCF exhibits the largest growth, but still falls behind others in overall performance trends.

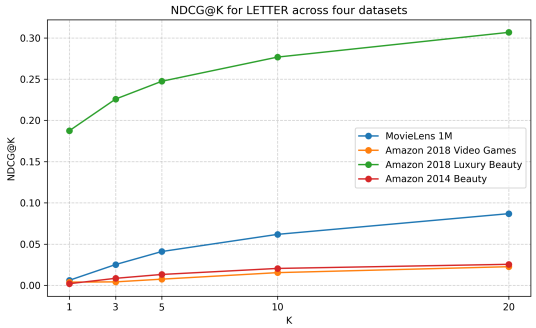
A potential reason for this under-performance is that NCF is fundamentally a collaborative filtering method based on implicit feedback. It models user preferences based solely on observed interactions, without leveraging the temporal order or detailed patterns in users' historical actions. However, in the video games domain, the next game a user engages with is often influenced by the most recently played. Since NCF lacks the ability to model such sequential context, it's unable to effectively model this pattern.

LETTER also exhibits subpar performance for this dataset. At K=1, it slightly outperforms NCF, which is reasonable given that LETTER is designed for sequential recommendation. However, as K increases to 3,5,10, and 20, LETTER consistently yields the lowest HR scores among all five models. It indicates that the limited effectiveness of LETTER in modeling long-term user preferences. One potential reason for LETTER's subpar performance lies in the generation of the item tokenization file in JSON format. As described in sub-section 5.1, this process requires an input item file containing item-specific information.

For Amazon 2018 datasets, both Video Games category and Luxury Beauty category, the item

representation is constructed using the title and description fields extracted from the meta file. However, the title and description fields has substantial difference in the quality and quantity of these fields across the two categories, particularly in the description field. In the Luxury Beauty category, descriptions tend to be detailed and informative, while in the Video Games category, many descriptions are either missing or very brief. This difference in textual richness may explain the weaker performance of LETTER in the Video Games domain. In the case of MovieLens 1M dataset, it is build using the title and genres fields.

(a) HR@K of LETTER across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty



(b) NDCG@K of LETTER across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

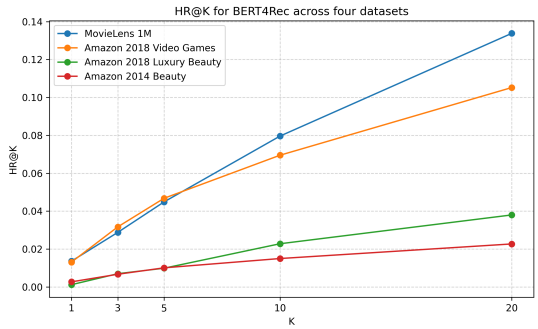
Figure 8: HR@K (K=1, 3, 5, 10, 20) and NDCG@K (K=1, 3, 5, 10, 20) of LETTER on MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

6.1.2 Case study: performance of LETTER across datasets

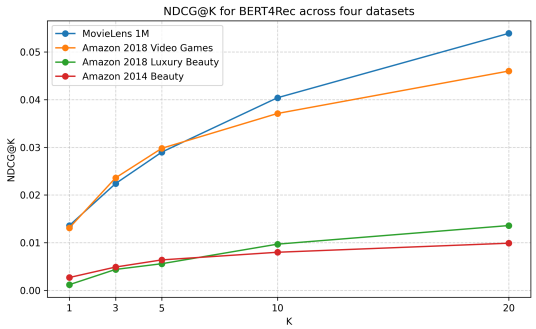
The performance of LETTER across four datasets is demonstrated as Figure 8. The performance of LETTER on the Amazon 2018 Luxury Beauty category is clearly superior to that on Amazon 2018 Video Games, Amazon 2014 Beauty, and MovieLens 1M. This is probably because for LETTER, the quality of the item tokenization directly affects model's performance. Except for the richer textual information mentioned in Section 6.1.1, the interaction file in JSON format is also very important when generating the index files. As a result, the frequency and distribution of interactions may affect which items are included or emphasized during the process, potentially impacting the effectiveness of model.

According to Table 5, the density of MovieLens 1M is 4.27%. As Table 6 describes the density of Amazon Review dataset, it is calculated with the total number of products in meta files. The actual density calculated using only the interacted items is about 0.5689% for Amazon 2018 Luxury Beauty category, 0.0734% for Amazon 2014 Beauty category, and 0.0518% for Amazon 2018 Video Games category, shown as Table 7.

This reveals a clear comparison: in terms of interaction density, MovieLens 1M is the densest among the four datasets, followed by Amazon 2018 Luxury Beauty, while Amazon 2014 Beauty and Amazon 2018 Video Games remain the sparsest. Amazon 2018 Luxury Beauty and MovieLens 1M are almost ten times and one hundred denser than Amazon 2014 Beauty and Amazon 2018 Video Games.



(a) HR@K of BERT4Rec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty



(b) NDCG@K of BERT4Rec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

Figure 9: HR@K (K=1, 3, 5, 10, 20) and NDCG@K (K=1, 3, 5, 10, 20) of BERT4Rec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

This large gap in density helps explain why LETTER performs substantially better on Amazon 2018 Luxury Beauty and MovieLens 1M than that on Amazon 2018 Video Games and Amazon 2014 Beauty. Likewise, the closeness in density between Amazon 2018 Video Games and Amazon 2014 Beauty accounts for their nearly aligned performance curves shown on Figure 8. Compared with Amazon 2018 Luxury Beauty dataset, Amazon 2018 Video Games contains much less textual information, specifically in the item description field. Among the four datasets, it is also the most sparse in terms of user-item interactions. Furthermore, unlike MovieLens 1M, the Amazon 2018 Video Games doesn't benefit from a high number of interactions per user or per item. These limitations, in both textual richness and interaction density, collectively help explain why LETTER performs the worst on the Amazon 2018 Video Games.

6.1.3 Case study: performance of BERT4Rec across datasets

Figure 9 clearly demonstrates the performance of BERT4Rec across the four datasets. BERT4Rec achieves the best performance on MovieLens 1M, followed by Amazon 2018 Video Games, and the lowest performance is observed on Amazon 2018 Luxury Beauty and Amazon 2014 Beauty. Table 5 and 8 show that MovieLens 1M contains 1,000,209 interactions, with about 165.6 interactions per user and 269.89 interactions per item. In comparison, Amazon 2018 Luxury Beauty, Amazon 2018 Video Games, and Amazon 2014 Beauty datasets have significantly fewer interactions. Amazon 2018 Video Games contains 497,577 interactions, which is about half of MovieLens 1M. Amazon 2014 Beauty contains 198,502 interactions, about 1/5 of MovieLens 1M. Amazon 2018 Luxury Beauty dataset has 34,278 interactions, only 1/29 of MovieLens 1M.

The average number of interactions per user and per item is also much lower in these three datasets, shown as Table 9. The average interactions per user and per item in these three datasets is about one to two orders of magnitude lower than MovieLens 1M. The number of interactions could influence the input sequence length. If the user has fewer interactions than the input sequence length, then the model would do the padding.

During the training process of BERT4Rec, it randomly masked items to learn how to predict

masked items based on the surrounding context, using information from both the left and the right. As a result, when users have fewer interactions like the three Amazon Review dataset, the number of items available to be masked is correspondingly much less, which limits BERT4Rec's capacity to learn from context. This could be the reason that BERT4Rec's performance on three Amazon Review datasets is lower, compared with the MovieLens 1M dataset.

Besides, there are other potential factors that result in the difference performance of BERT4Rec across these four datasets. From Figure 9, it is clear to see that when $K \leq 5$, the curves of MovieLens 1M and Amazon 2018 Video Games are almost overlap, indicating that their performance is quite similar at lower cutoffs. In particular, at the lower cutoffs, BERT4Rec on Amazon 2018 Video Games performs slightly better than that on MovieLens 1M. The potential reason is the difference in the cold-start item ratio between these two datasets.

For Amazon 2018 Video Games, there are only 114 cold-start items, defined as items with fewer than 5 interactions, among 17,408 interacted items. It results in a cold-start ratio is approximately 0.65%, shown as Table 9, which means the majority items have been interacted multiple times.

Compared with Amazon 2018 Video Games, MovieLens 1M exhibit higher cold-start ratios of 7.47%. The cold-start ratio of MovieLens 1M is about 11.49 times higher than that of Amazon 2018 Video Games. This potentially explains why BERT4Rec on Amazon 2018 Video Games outperforms it on MovieLens 1M at lower cutoffs.

Similarly, in another pair of datasets, Amazon 2018 Luxury Beauty and Amazon 2014 Beauty, this pattern is also been seen. These two datasets have very similar overall performance. And Amazon 2014 Beauty has zero cold-start item, with a cold-start ratio 0. While Amazon 2018 Luxury Beauty contains 143 cold-start items, with a cold-start item ratio of about 9.04%. In the lower cutoffs, $K \leq 5$, Amazon 2014 Beauty shows slightly better performance than Amazon 2018 Luxury Beauty.

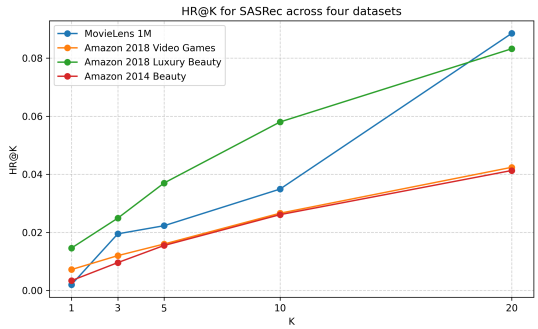
Both cases suggests the impact of cold-start item ratio on performance of BERT4Rec at lower cutoffs, but there is other factors we need to take into consideration. Starting from $K = 10$, MovieLens begins to show a noticeable advantage over Amazon 2018 Video Games, suggesting that its superior performance becomes obvious as the recommendation list gets longer.

One possible reason for this is the difference in the proportion of popular items, defined as items with more than 50 interactions.

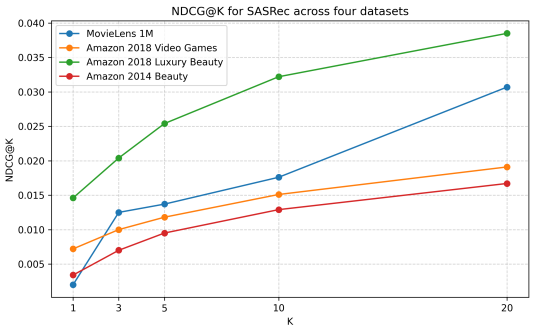
Shown as Table 8, popular items account for 64.74% of all interactions, whereas in Amazon 2018 Video Games, the number is 12.97%. It indicates that MovieLens 1M has much more interactions on a small set popular items, approximately 5.41 times that of Amazon 2018 Video Games. For Amazon 2018 Luxury Beauty, the popular item ratio is 4.11%, which is far lower that other two datasets. In fact, popular item ratio of Amazon 2018 Luxury Beauty is nearly 16 times lower than MovieLens 1M, and more than three times lower than Amazon 2018 Video Games.

When $K \leq 5$, the top items in both MovieLens 1M and Amazon 2018 Video Games datasets can be easily covered by their respective sets of popular items, which could lead to the similar performance at lower cutoffs. However, as K increases to 10, the difference in the popular item sets starts to make difference.

For the Amazon 2018 Video Games, the popular item set consists of approximately 12.97%. As the model tries to recommend more items, the model may need to explore less popular items or long-tail items, which are generally harder to predict accurately. In contrast, MovieLens 1M has a much larger popular item set, approximately 64.74% of interactions. There are 2514 popular items from only 3,883 items. This remains sufficient to support high-quality



(a) HR@K of SASRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty



(b) NDCG@K of SASRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

Figure 10: HR@K (K=1, 3, 5, 10, 20) and NDCG@K (K=1, 3, 5, 10, 20) of SASRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

recommendations even at higher cut-offs. In terms of both the number of popular items and the proportion relative to the total items, MovieLens 1M holds a clear advantage over Video Games, especially in terms of proportion. It allows BERT4Rec to remain strong performance on MovieLens 1M.

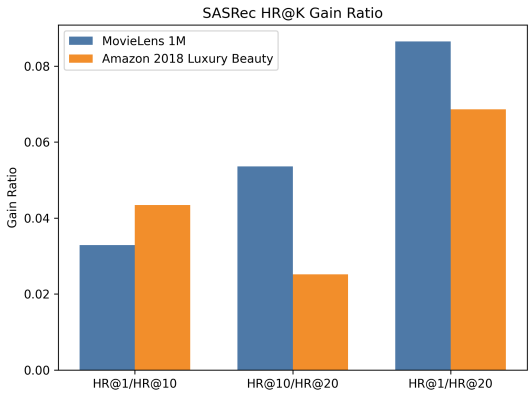
Overall, MovieLens 1M has much more popular items, a higher average interactions, and it maintains strong performance throughout. This is specifically evident when K is large, where the high proportion of popular items gives it a clear advantage. Amazon 2018 Video Games has a lower popular item ratio and moderate interaction counts. However, it contains very few cold-start items, which helps BERT4Rec perform well when K is small. AS K increases, the limited set of popular items makes the performance falls behind the performance on MovieLens 1M. Luxury Beauty, on the other hand, performs the worst for BERT4Rec. It doesn't have sufficient interactions, limiting the capacity of BERT4Rec to learn how to predict the masked items based on the surrounding context from the very start. In addition, it has more cold-start items and lower proportion of popular items, which also leads to the poor performance of BERT4Rec on it.

6.1.4 Case study: performance of SASRec across datasets

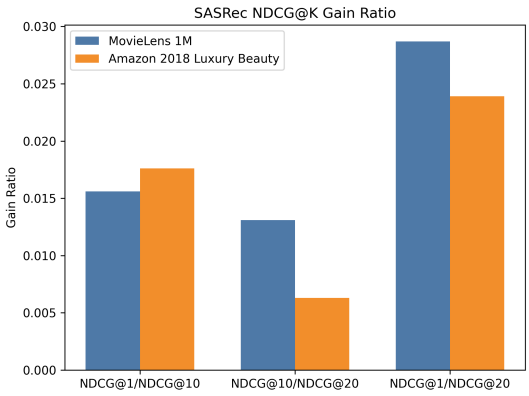
On the SASRec, Amazon 2018 Luxury Beauty demonstrates a clear advantage, shown as Figure 10. It achieves the highest performance starting from K=1 and remains consistently ahead, with a smooth and steady upward curve.

At K=1, Amazon 2018 Video Games dataset scores 0.0072 and Amazon Beauty 2014 scores 0.0034, clearly outperforming MovieLens 1M with 0.0020. But then their performance falls behind and remains the lowest across all subsequent K values, showing slow and nearly linear growth. Shown as Table 9 and Table 8, the cold-item ratio 0.65% of Amazon 2018 Video Games and 0.00% of Amazon 2014 Beauty datasets are much lower than 7.47% of MovieLens 1M. This may account for better performance of SASRec on both Amazon 2018 Video Games and Amazon 2014 Beauty than that on MovieLens 1M at K=1.

However, the density of Amazon 2018 Video Games and Amazon 2014 Beauty is only approx-



(a) HR@K gain ratio of SASRec on MovieLens 1M and Amazon 2018 Luxury Beauty



(b) NDCG@K gain ratio of SASRec on MovieLens 1M and Amazon 2018 Luxury Beauty

Figure 11: Gain ratio (HR@1/HR@10, HR@10/HR@20, HR@1/HR@20, NDCG@1/NDCG@10, NDCG@10/NDCG@20, NDCG@1/NDCG@20) of SASRec on MovieLens 1M and Amazon 2018 Luxury Beauty

imately 0.0518% and 7.34%, respectively. These two are the sparsest datasets. Besides, for Amazon 2018 Luxury Beauty, it only contains 12,308 items in the meta file, which leads to a much smaller vocab. And MovieLens 1M only contains 3,883 items. On contrast, the size of Amazon 2018 Video Games vocab is 84,893, almost 7 times larger than Amazon 2018 Luxury Beauty and almost 22 times larger than MovieLens 1M. The size of Amazon 2014 Beauty vocab is 259,204, approximately 21 times larger than Amazon 2018 Luxury Beauty and nearly 67 times larger than MovieLens 1M.

SASRec's poorest performance on the Amazon 2018 Video Games and Amazon 2014 Beauty could be mostly explained by the combination of severe sparsity and scale.

For MovieLens 1M, it starts with the lowest performance at K=1, but has the steepest increase in HR as K approaches 20. At K=20, HR@20 of MovieLens 1M surpasses that of Amazon 2018 Luxury Beauty, although its NDCG@20 remains slightly lower. SASRec applies self-attention mechanism to predict the next item based on users' historical actions. For dense datasets, it tends to capture long-range dependencies, while it put more emphasis on recent user behaviors for sparse datasets.

Amazon 2018 Luxury Beauty and MovieLens 1M datasets are relatively dense, with densities of approximately 0.5689% and approximately 4.27%, respectively. MovieLens 1M is noticeably denser than Amazon 2018 Luxury Beauty, but the performance of Amazon 2018 Luxury Beauty is better, which is possibly because the order of data in MovieLens 1M can't accurately reflect the order of the sequence of user's behaviors.

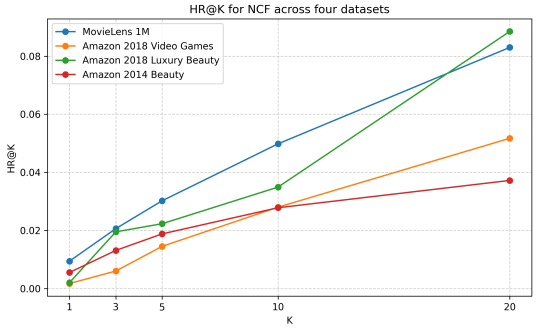
Users in Amazon 2018 Luxury Beauty have 9 interactions in average, while in MovieLens 1M users have approximately 165.9 interactions in average. Even more interactions could be an advantage in some cases, long sequences could also induce noise, eventually leading to worse Hit Ratio for SASRec on MovieLens 1M in low cutoffs.

When cutoff is high, like at K=20, the performance of SASRec on MovieLens 1M is better than that on Amazon 2018 Luxury Beauty. The popular item ratio in Amazon 2018 Luxury Beauty is only approximately 4.11%, while that in MovieLens 1M is 64.74%.

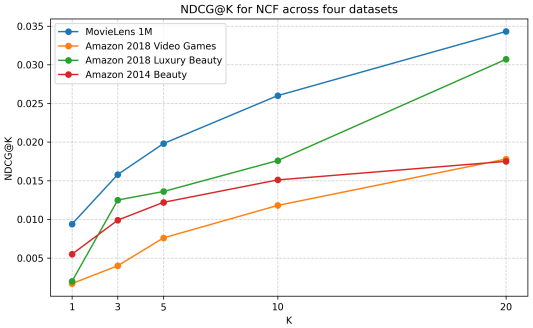
As K increases, for Amazon 2018 Luxury Beauty, SASRec needs to explore more long-tail

items. But for MovieLens 1M, it has much more popular items, and all the items have at least 20 interactions. It means that even SASRec needs to explore long-tail items, predictions on MovieLens 1M are consequently easier and more accurate.
As a result, the performance gain from K=10 to K=20 is much larger for Movielens than that for Amazon 2018 Luxury Beauty. The gain ratio for SASRec on Amazon 2018 Luxury Beauty and MovieLens 1M datasets are shown in in Figure 11. It demonstrates the gain ratio from K=1 to K=10, from K=10 to K=20, and from K=1 to K=20.

6.1.5 Case study: performance of NCF across datasets



(a) HR@K of NCF across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

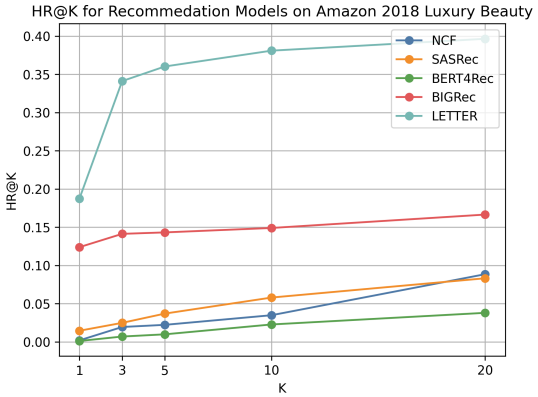


(b) NDCG@K of NCF across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

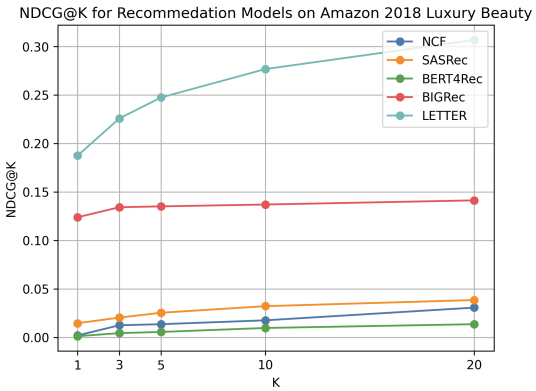
Figure 12: HR@K (1, 3, 5, 10, 20) and NDCG@K (1, 3, 5, 10, 20) of NCF across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

For NCF, at K=1, MovieLens 1M outperforms by a large margin, while the Amazon 2018 Video Games and Amazon 2018 Luxury Beauty score worst. Overall, MovieLens 1M maintains its top performance until around K=20, where HR@K score of Amazon 2018 Luxury Beauty surpasses it. However, NDCG@K scores show that MovieLens 1M holds an advantage throughout. For the overall trend, Amazon 2018 Video Games and Amazon 2014 Beauty datasets perform the worst consistently across the four datasets on NCF.
NCF focuses on collaborative filtering based on the implicit feedback. The core mechanism of NCF is to use deep neural networks to model latent features of users and items. It learns from the interactions. As a result, the quality and quantity of interactions are the key factors to determine NCF's performance across different datasets.
MovieLens 1M dataset stands out clearly from other three Amazon Review datasets when it comes to interaction patterns. On average, each user and each item in MovieLens 1M dataset have a substantially higher number of interactions available for learning. In contrast, Amazon Review datasets show much lower average interactions per user and per item. For MovieLens 1M, the values of average interactions per user and per item are approximately 165.6 and 268.89 respectively. In comparison, Amazon 2018 Video Games has average of about 9.01 interactions per user and about 28.58 interactions per item. Amazon 2018 Luxury Beauty contains only about 8.98 interactions per user and only about 21.68 interactions per item. Amazon 2014 Beauty includes about 8.88 interactions per user and only about 16.4 interactions per item. This means that, each of the three Amazon Review datasets contains

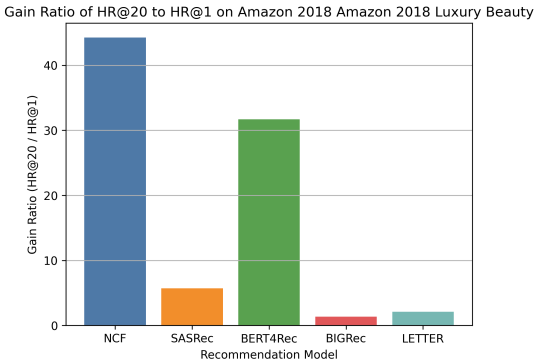
only about 5% as many interactions per user and 10% as many interactions per item as MovieLens 1M.
Besides, all the users in MovieLens 1M have at least 20 interactions, whereas Amazon Review datasets only guarantee at least 5 interactions of each user. Moreover, there are about 2514 items in MovieLens 1M dataset has at least 50 interactions, accounting for about 64.74% of all items. Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty datasets only have about 12.97%, 4.11%, and 6.13% of items meeting this threshold, which is roughly 20.03%, 6.35%, and 9.46% of that in MovieLens 1M.



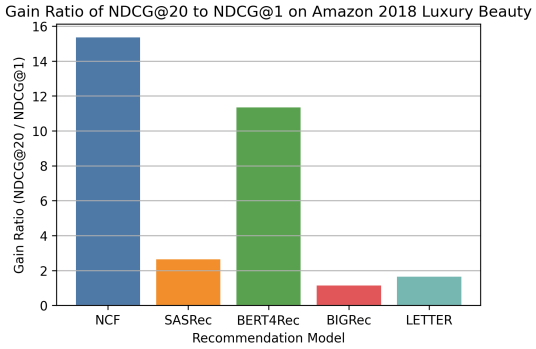
(a) HR@K of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Luxury Beauty



(b) NDCG@K of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Luxury Beauty



(c) Gain ratio (HR@20/HR@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Luxury Beauty



(d) Gain ratio (NDCG@20/NDCG@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Luxury Beauty

Figure 13: HR@K (K=1, 3, 5, 10, 20), NDCG@K (K=1, 3, 5, 10, 20), gain ratio (HR@20/HR@1), and gain ratio (NDCG@20/NDCG@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on Amazon 2018 Luxury Beauty

Table 15: HR and NDCG performance of sequential models on datasets: MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, Amazon 2014 Beauty (k = 1, 10, 20)

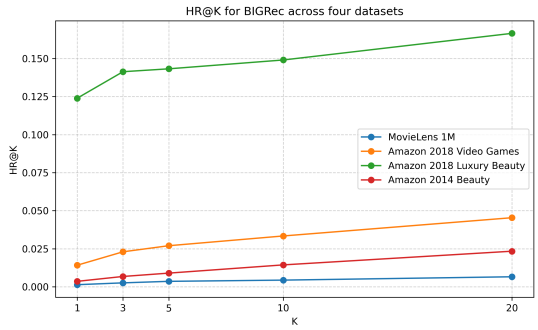
Dataset	Model	HR@k			NDCG@k		
		1	10	20	1	10	20
MovieLens 1M	SASRec	0.0020	0.0349	0.0885	0.0020	0.0176	0.0307
	BERT4Rec	**0.0136**	**0.0796**	**0.1338**	**0.0136**	0.0404	0.0539
	BIGRec	0.0014	0.0044	0.0066	0.0014	0.0028	0.0033
	LETTER	0.0061	0.0741	0.1238	0.0061	**0.0618**	**0.0869**
Amazon 2018 Video Games	SASRec	0.0072	0.0266	0.0424	0.0072	0.0151	0.0191
	BERT4Rec	0.0131	**0.0695**	**0.1052**	0.0131	**0.0371**	**0.0460**
	BIGRec	**0.0142**	0.0334	0.0454	**0.0142**	0.0231	0.0261
	LETTER	0.0039	0.0176	0.0313	0.0039	0.0154	0.0226
Amazon 2018 Luxury Beauty	SASRec	0.0146	0.0580	0.0832	0.0146	0.0322	0.0385
	BERT4Rec	0.0012	0.0228	0.0380	0.0012	0.0097	0.0136
	BIGRec	0.1239	0.1490	0.1665	0.1239	0.1370	0.1413
	LETTER	**0.1874**	**0.3809**	**0.3964**	**0.1874**	**0.2767**	**0.3067**
Amazon 2014 Beauty	SASRec	0.0034	**0.0261**	**0.0413**	0.0034	0.0129	0.0167
	BERT4Rec	0.0027	0.0150	0.0227	0.0027	0.0080	0.0099
	BIGRec	**0.0036**	0.0144	0.0234	**0.0036**	0.0081	0.0104
	LETTER	0.0020	0.0254	0.0342	0.0020	**0.0205**	**0.0255**

6.2 Performance of traditional and LLMs-based recommendation model for sequential recommendation tasks across four datasets

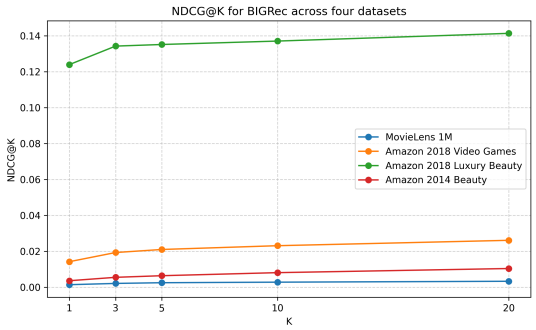
The performance of the sequential recommendation settings of both traditional and LLMs-based recommendation models, SASRec, BERT4Rec, BIGRec, and LETTER which is instantiated on TIGER, is summarized in Table 15. It reports the Hit Rate (HR) and the Normalized Discounted Cumulative Gain (NDCG), evaluated at cut-off values K=1, 10, and 20. The best result for each metric of each model on each dataset is highlighted in bold.

6.2.1 Case study: in-depth analysis and comparative evaluation of BIGRec

Figure 14 clearly shows that BIGRec performs much better on Amazon 2018 Luxury Beauty, compared with Amazon 2018 Video Games, Amazon 2014 Beauty and MovieLens 1M datasets. Starting at K=1, the HR@1 for BIGRec on Amazon 2018 Luxury Beauty is about 0.1239, which is roughly 7.73 times higher than 0.0142 it achieves on Amazon 2018 Video Games and nearly 35 times higher than 0.0036 it achieves on Amazon 2014 Beauty. Meanwhile, HR@1 of BIGRec on MovieLens 1M is only 0.0014. It is also worth noting that although BIGRec does not do as

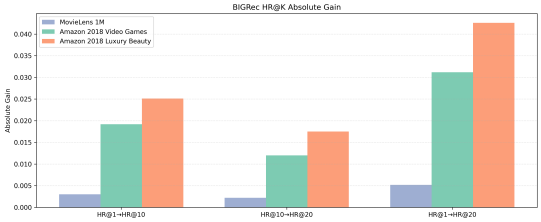


(a) HR@K of BIGRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

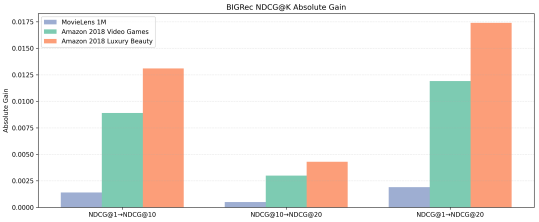


(b) NDCG@K of BIGRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty

Figure 14: HR@K (K=1, 3, 5, 10, 20), and NDCG@K (K=1, 3, 5, 10, 20) of BIGRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and Amazon 2014 Beauty



(a) HR@K absolute gain of BIGRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty



(b) NDCG@K absolute gain of BIGRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty

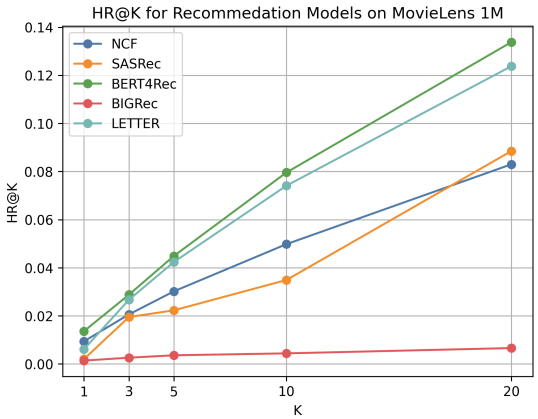
Figure 15: HR@K and NDCG@K absolute gain (K=1 to K=20, K=10 to K=20, K=1 to K=10) of BIGRec across MovieLens 1M, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty

well on Amazon 2018 Video Games and Amazon Beauty as it does on Amazon 2018 Luxury Beauty, they have a similar trend. The curves steadily improve as K increases. However, BIGRec on MovieLens 1M not only starts off with the poorest performance but also shows almost no growth as K increases. The absolute gain of BIGRec across the three datasets of K = 1 to K = 10, K = 10 to K = 20, and K = 1 to K = 20 is clearly illustrated in Figure 15.

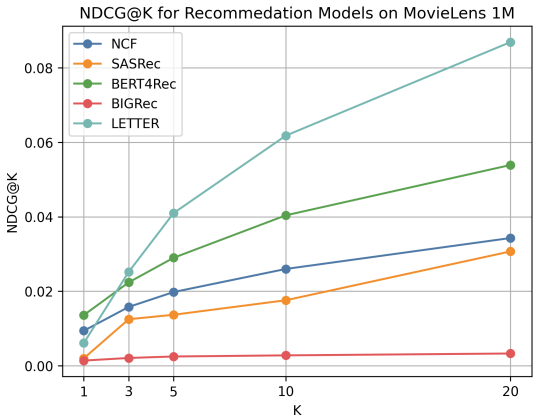
One possible reason for BIGRec's poor performance on MovieLens 1M is that ratings in this dataset may not accurately reflect the users' actual viewing order. Even though MovieLens 1M has a relatively high density, reaching about 4.27%. It is also with a substantial average number of interactions per user and per item and each item has at least 20 interactions. The cold-start item ratio, about 7.47%, is also moderate. This dataset differs from the other three Amazon Review datasets in one key aspect. Specifically, users often rate multiple scores at once in some particular cases. Unlike shopping platforms, some applications and websites are specifically designed for rating content, for movies, music, or books. In this situation, the rating order doesn't necessarily correspond to the users' actual viewing sequence. This lack of chronological order may introduce noise rather than useful information, potentially influencing

the sequential recommendation model's performance.

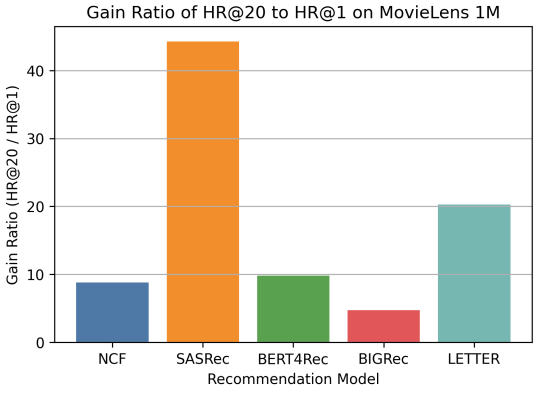
Comparative study: BIGRec and BERT4Rec Another observation that supports this explanation comes from the training process of BIGRec. While the training loss keeps decreasing steadily, the evaluation loss remains stuck around a certain value. This phenomenon persists even after tuning hyper-parameters, which implies the presence of noise in the data. The noteworthy point is that, even though MovieLens 1M performs poorly on most sequential recommendation models, it achieves its best performance on BERT4Rec, which is also designed for sequential recommendation tasks. The performance of MovieLens 1M across these four sequential recommendation paradigms is shown in Figure 16a and 16b.



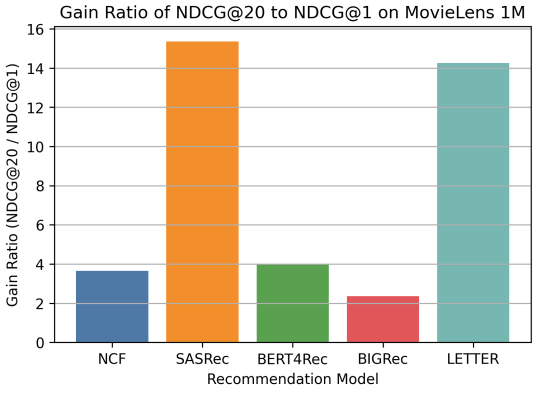
(a) HR@K of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on MovieLens 1M



(b) NDCG@K of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on MovieLens 1M



(c) Gain ratio (HR@20/HR@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on MovieLens 1M

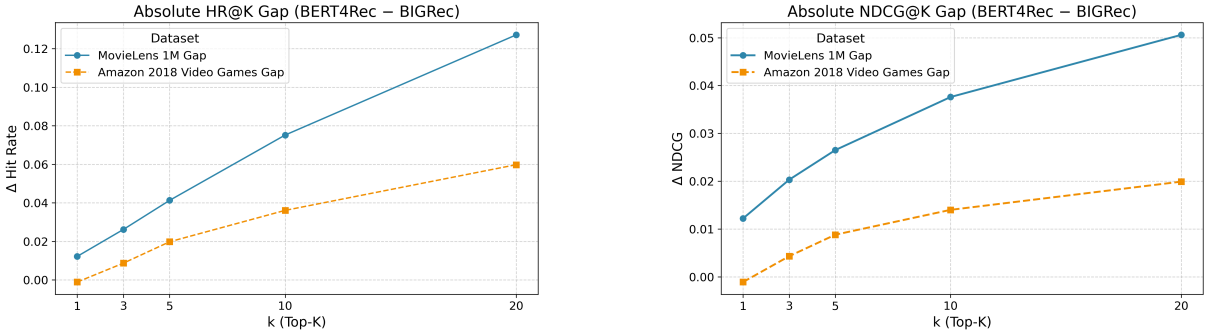


(d) Gain ratio (NDCG@20/NDCG@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on MovieLens 1M

Figure 16: HR@K (K=1, 3, 5, 10, 20), NDCG@K (K=1, 3, 5, 10, 20), gain ratio (HR@20/HR@1), and gain ratio (NDCG@20/NDCG@1) of NCF, SASRec, BERT4Rec, BIGRec, and LETTER on MovieLens 1M

The poor performance of MovieLens 1M on other sequential recommendation paradigms is highly likely due to its insufficient capacity to accurately reflect the actual sequence of users' real-life actions. Interestingly, this limitation may explain why it performs so well on BERT4Rec.

The features of BERT4Rec effectively address this specific issue of MovieLens 1M, while still preserving other important statistical properties.
The core mechanism of BERT4Rec is to randomly mask some items within the sequence and learn to predict the ids of masked items based on the context from both directions, left and right. This bidirectional learning is the most unique feature of BERT4Rec. In contrast, other sequential recommendation models, like SASRec, can only learn from the left side. This distinction is crucial as it determines whether the model can effectively address or inadvertently reinforce the limitation of MovieLens 1M.
Unlike MovieLens 1M, of which the performance differs significantly between BIGRec and BERT4Rec, the Amazon Video Games consistently reflects the actual users' behaviors sequence. This is likely because Amazon 2018 Video Games originates from the shopping platform, where the users' actions generally follow the true chronological order reflected in the review timestamps.

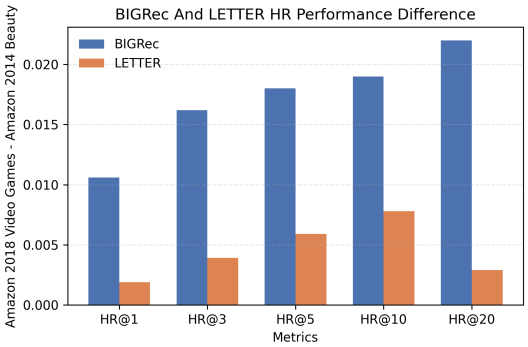


(a) Absolute HR@K gap between BIGRec and BERT4Rec on MovieLens 1M and Amazon 2018 Video Games

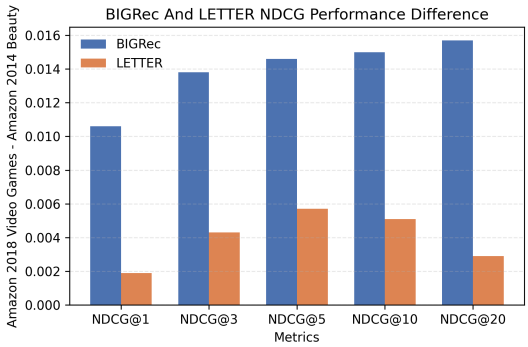
(b) Absolute NDCG@K gap between BIGRec and BERT4Rec on MovieLens 1M and Amazon 2018 Video Games

Figure 17: Absolute HR@K (K=1, 3, 5, 10, 20) gap and NDCG@K (K=1, 3, 5, 10, 20) gap between BIGRec and BERT4Rec on MovieLens 1M and Amazon 2018 Video Games

Comparative study: BIGRec and LETTER In contrast to MovieLens 1M, other three datasets are from Amazon, which is the shopping website. On such shopping platforms, users' review tend to align more closely with the actual chronological order of their purchases. As shown clearly in Figure 15, the scores for all three datasets from Amazon increase steadily as K grows.
BIGRec's strong performance on Amazon 2018 Luxury Beauty is evident since K=1 and continues consistently throughout. Both the popular item ratio and cold-start item ratio of these three datasets have some impact. But the primary factor is likely the difference in density. Amazon 2018 Luxury Beauty has a density of 0.57%, compared to 0.05% of Amazon 2018 Video Games and 0.07% of Amazon 2014 Beauty, showing a ten-fold and eight-fold difference. This high density helps explain the clear advantage of Amazon 2018 Luxury Beauty on BIGRec. However, the performance of BIGRec on Amazon 2018 Video Games and on Amazon 2014 Beauty show clear difference, shown as Figure 14. Even they have similar density and interaction patterns, with both an average of about 9 interactions per user.
The potential reason is that, in Amazon 2014 Beauty, the titles are more complex and difficult to distinguish from one another. The items in Amazon 2014 Beauty are beauty products. Compared with the titles from video games products, the titles from beauty products are way



(a) HR@K absolute difference between Amazon 2018 Video Games and Amazon 2014 Beauty on BIGRec and LETTER



(b) NDCG@K absolute difference between Amazon 2018 Video Games and Amazon 2014 Beauty on BIGRec and LETTER

Figure 18: HR@K (K=1, 3, 5, 10, 20) and NDCG@K (K=1, 3, 5, 10, 20) absolute difference between Amazon 2018 Video Games and Amazon 2014 Beauty on BIGRec and LETTER

more complicated, often including various attributes. It is very common that they mention the ingredients used, such as Vitamin C, retinol, or various plant extracts. The titles also highlight the intended benefits, such as whitening or moisturizing. In addition, they specify the product type. They also need to specifically ends with the volume.

Embedding match is the core evaluation mechanism in BIGRec. It is based on the similarity score between the prediction embedding and the item embedding to rank. When titles include a series of similar attributes, it induces much noise. These long and similar titles are very likely to be mapped to very close points in vector space.

Another observation that supports this explanation is the performance of LETTER on Amazon 2018 Video Games and Amazon 2014 Beauty, shown as Figure 8. Rather than embedding match, LETTER resorts to item tokenization, a sequence of identifiers. Even the same beauty product with different volumes, they could have different token id. For this reason, unlike BIGRec, the performance of LETTER on on Amazon 2018 Video Games and Amazon 2014 Beauty have close performance curve. The performance differences between Amazon 2018 Video Games and Amazon 2014 Beauty on BIGRec and LETTER are shown as Figure 18.

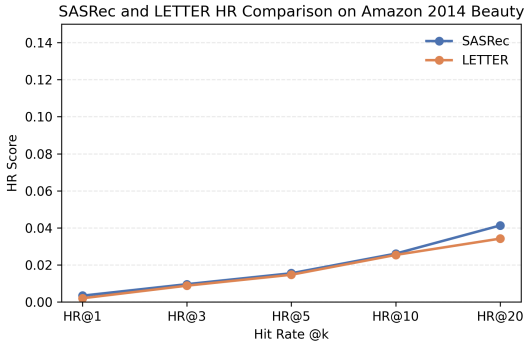
6.2.2 Case study: the impact of SASRec on LETTER

In LETTER, CF embedding is required to inject collaborative information to the code sequence. Notably, CF embedding is generated from another sequential recommendation model, SASRec. As a result, LETTER and SASRec exhibits similar performance trends on certain datasets, Amazon 2018 Video Games and Amazon 2014 Beauty.

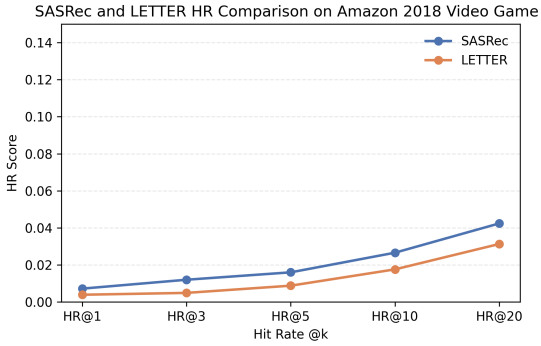
Figure 19 clearly demonstrates the performance trends of LETTER and SASRec on four datasets.

However, on MovieLens 1M and Amazon 2018 Luxury Beauty, the performance trends of LETTER and SASRec doesn't align, and LETTER substantially outperforms SASRec. It is likely because the high density of MovieLens 1M and Amazon 2018 Luxury Beauty as mentioned in Section 6.1.2.

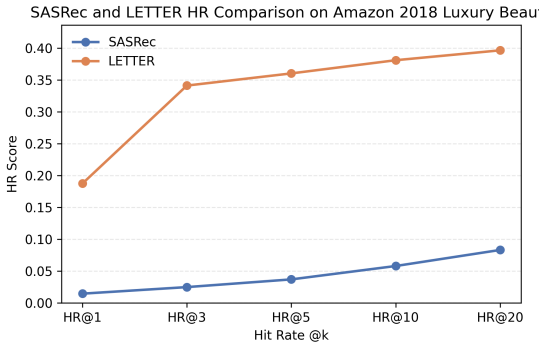
The slight performance drop of LETTER compared with SASRec could be attributed to parameter difference. Specifically, when generating the collaborative filtering embeddings for LETTER, the embedding dimensions must align. To ensure this alignment, the hidden units



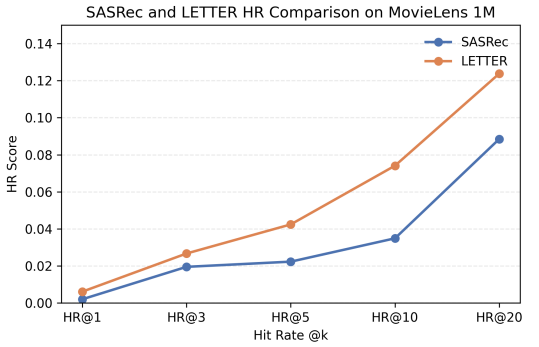
(a) Performance difference (HR@K) between LETTER and SASRec on Amazon 2014 Beauty



(b) Performance difference (HR@K) between LETTER and SASRec on Amazon 2018 Video Games)



(c) Performance difference (HR@K) between LETTER and SASRec on Amazon 2018 Luxury Beauty



(d) Performance difference (HR@K) between LETTER and SASRec on MovieLens 1M

Figure 19: Performance difference HR@K (K=1, 3, 5, 10, 20) between LETTER and SASRec on Amazon 2014 Beauty, Amazon 2018 Video Games, Amazon 2018 Luxury Beauty, and MovieLens 1M

is set to 32, while it is originally set to 50 when SASRec is used independently. This change may have affected the final performance difference between these two models.

7 Discussions

7.1 Limitations

In this study, although we choose a diverse set of recommendation models, including conventional, sequential, and generative methods, the scope of model selection is still limited. For the comparison among sequential models, group sequential is absent. And the generative recommendation models we use – BIGRec and LETTER – are both sequential models. But there are also other generative models, which we don't include in this study, such as the text-only generative recommendation models.

Another point worth noting is that LETTER is a generative recommendation model, but it is an enhancement of TIGER which is also a generative recommendation model. As a

result, its improved performance may be attributed not only its own innovations but also the features of TIGER. This raises the question of whether too many contributing factors are involved, potentially effecting the fairness of the comparison. In addition, during the process of generating Semantic ID for LETTER, we inject the collaborative filtering embedding from SASRec. Meanwhile, for both Amazon 2014 Beauty and Amazon 2018 Luxury Beauty, SASRec and LETTER exhibit a strong consistency in performance. Even we conduct some analysis on this consistency, the extent to which it influences standalone or combined with other components remains unclear. It further complicates fair evaluation and model attribution.
We use ranking-based metrics, HR and NDCG, not considering other aspects of recommendation quality, such as diversity and user satisfaction. Moreover, our comparison solely focuses on model performance and doesn't take the computational efficiency into account, such as time and resource consumption.

8 Conclusions

8.1 Research questions

RQ1: What are the potential factors influencing the performance of recommendation models and how these factors make impact?
The potential factors influencing the performance of recommendation models include data sparsity, user-item interaction patterns, the reliability of sequence information, and domain-specific features. These factors substantially influence the effectiveness and generalization capacity of recommendation models.
The number of average interactions per user directly effects the effective length of the sequence, which eventually impacts the capacity of sequential setting of traditional recommendation models to learn from it.
In addition, the sparsity/density of datasets plays an important role, across different recommendation models. Higher density represents more sufficient interactions for models to learn. For non-sequential setting of traditional recommendation models like NCF, it resorts to deep neural networks to learn the interaction from data. Lacking of sufficient interactions could induce overfitting and poor generalization capacity, while higher density helps to learn more robust and stable. For non-sequential setting of traditional recommendation models, the length of sequence strongly affects performance. SASRec relies on self-attention from Transformer while BERT4Rec resorts to Transformer with masked item prediction. Both of them are designed to capture the context of users' historical actions. However, when sequences are short, they are limited to capture the long-term semantics. While high density means that even there exist some users have short length of sequence, high data density still helps the models to capture reliable patterns from other users. Our findings demonstrate that for LLMs-based recommendation models, the highest-density dataset performs substantially better than other datasets on both BIGRec and LETTER. The higher density provides richer contextual information for LLMs to capture semantic information and the representations of items become more reliable. As a result, LLMs-based recommendation models show remarkable performance on high density datasets, confirming that data density is one of the key influencing factors.
The domain feature of datasets influences the performance of recommendation models. Differences in dataset domains could result in inherent differences of textual information. In some cases, this could be beneficial, as it provides richer textual information for model to learn. While in other cases, it may introduce excessive noise that impairs the performance of models.

Such differences have great impact on LLMs-based recommendation systems, as they largely rely on the capabilities of LLMs. In this thesis, the most notable difference caused by variations in dataset domains is the differences in title complexity. Although this difference is most evident for LLMs-based recommendation models, the influence it causes on BIGRec and LETTER varies as each model has its own specific working mechanism. For BIGRec which is based on the similarity score between the prediction embedding and the item embedding to rank, more complicated the title represents more noise. But for LETTER which relies on item tokenization, the limitation is reduced.

The reliability of the order of data also plays important role, especially for non-sequential setting of traditional recommendation models. For some datasets, like MovieLens 1M, the order of data doesn't accurately reflect the actual sequence of the user's behaviors. As the timestamp in MovieLens 1M dataset record the time of ratings submitting, rather than the time when users actually watch movies. This lack of reliable chronological order could introduce noise and prevents models from learning effectively, resulting in poor generalization capability. This limitation is evident for LLMs-based recommendation models, particularly on BIGRec. BIGRec heavily relies on accurate temporal order to capture semantic dependencies between items. When the order of data does not reflect the actual sequence of the user's behaviors, it leads to notable poor performance. In contrast, even LETTER and SASRec are also affected by this limitation, their working mechanisms help to reduce the impact to some extent. LETTER mitigates this limitation by using item tokenization to extract semantic information. SASRec relies on self-attention mechanism from Transformer to model local dependencies, making it less dependent on the sequence order. Among all these models, BERT4Rec is the least affected by this limitation, as its architecture is designed with this situation into consideration. The core mechanism of BERT4Rec is randomly mask items and learn to predict the ids of masked items from both directions. This bidirectional learning, which is the most unique feature of BERT4Rec, effectively addresses the limitations of unreliable sequence order.

For non-sequential setting of traditional recommendation models, both the length and density of user-item interactions are critical factors, while the order of interactions plays a relatively minor role.

Another important factor of datasets is the cold-start item ratio, which substantially affects performance of model at smaller K values, particularly in top-1 recommendation. A lower proportion of cold-start items generally leads to higher score in top-1 recommendation.

Similar with the cold-start item ratio, the proportion of popular items in a dataset also has a notable impact on recommendation model performance. When the number of popular items is limited, as the K increases, the model is supposed to explore long-tail or cold-start items, which are much less frequently interacted. This induces the decline in recommendation performance, at higher cut-off values.

RQ2: In what scenarios are different recommendation models are suitable?

Our results demonstrate that model performance varies depending on the characteristics of datasets, as shown below:

- For sequential recommendation tasks yet with datasets where the order of the user-item interactions doesn't accurately reflect the actual sequence of the user's behaviors, it is better to choose sequential recommendation model that are robust to such inconsistencies, such as BERT4Rec. As the bidirectional learning capacity of BERT4Rec helps address this issue by capturing context information from both left and right side, making it less influenced by the inconsistency between order of data and order of sequential behaviors.

- For sequential recommendation tasks on datasets where the length of interactions is not sufficient enough but has high density, SASRec is a better choice. As high density helps for the limited sequence length for SASRec.
- For sequential recommendation tasks on datasets which contain rich textual information, generative recommendation models, especially LETTER, are particularly suitable, as they can leverage semantic signals to improve recommendation quality.
- For sequential recommendation task on high-density datasets, BIGRec is more suitable, as it can effectively learn from the sufficient interaction information.
- For implicit feedback and tasks focusing on predicting the user's preference, rather than the next item in a sequence, NCF is more suitable. As it models static user-item interactions, and it performs well on high-density datasets.

8.2 Future work

Based on the analysis and conclusions presented above, and considering the limitations of this study, several directions for future work are proposed as follows:

- Classic model designed for sequential recommendation tasks like BERT4Rec have been proposed to address the inconsistent interaction order by leveraging its bidirectional learning feature. However, LLMs-based sequential recommendation model have not fully explore this aspect. The LLMs-based sequential recommendation model's robustness to datasets, where the order of interactions doesn't represent the actual sequence, is an important direction.
- LLMs-based sequential recommendation models perform well on high-density datasets. In low-density scenarios, future work could explore the injection of richer descriptive information. LETTER injects the descriptive information related to items, but user-centric information has been neglected. It limits model's capacity to recommend based on user characteristics.

A Descriptive figures of datasets

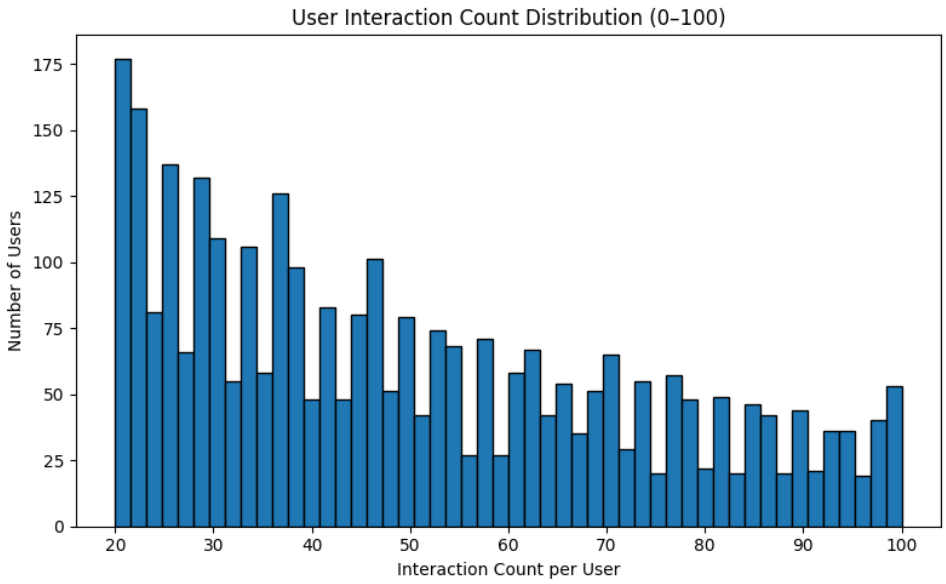
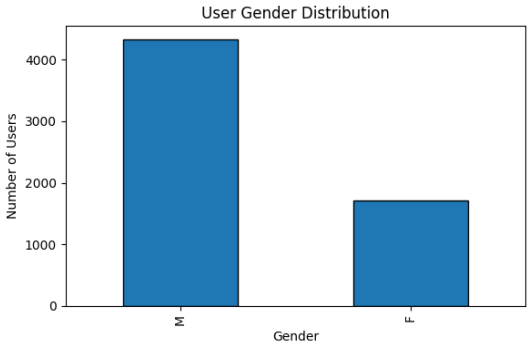
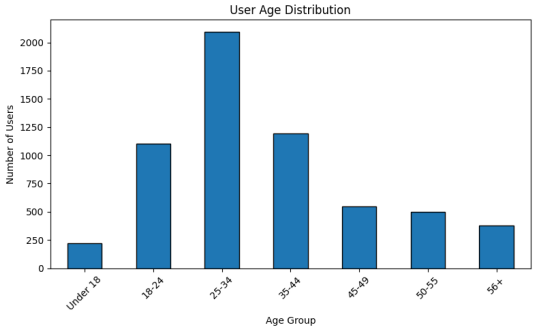


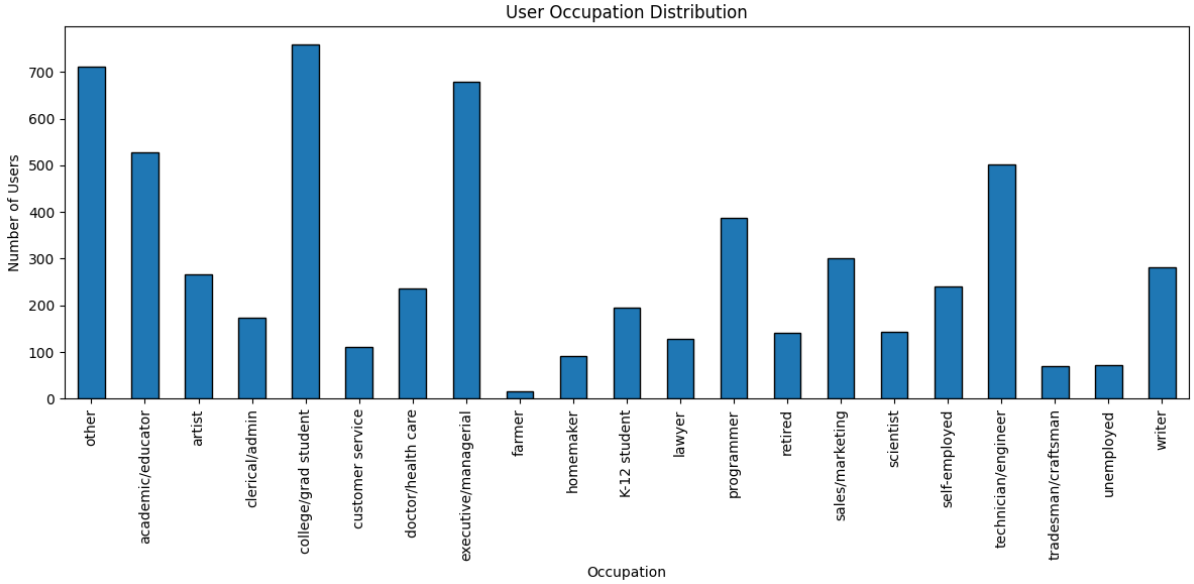
Figure 20: User interaction statistics in MovieLens 1M: Histogram



(a) Gender distribution in MovieLens 1M



(b) Age distribution in MovieLens 1M



(c) Occupation distribution in MovieLens 1M

Figure 21: Distribution of user demographics in MovieLens 1M: gender, age, and occupation

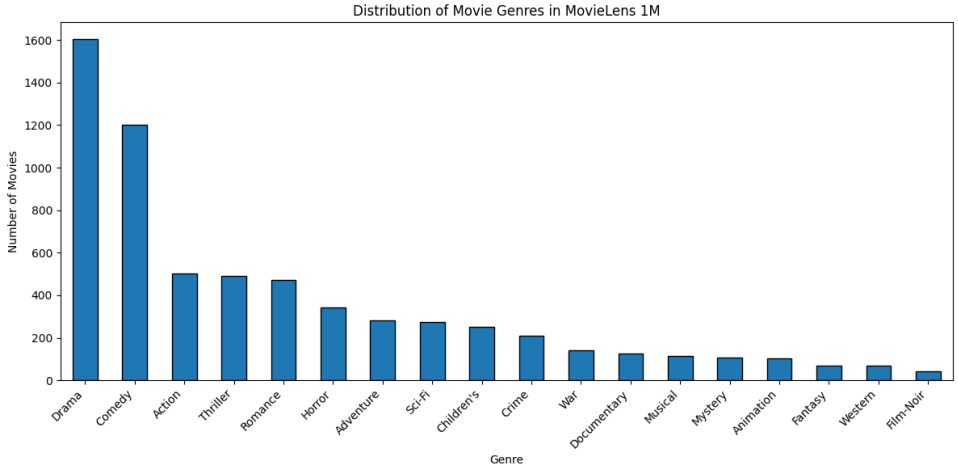


Figure 22: Movie genres distribution in MovieLens 1M

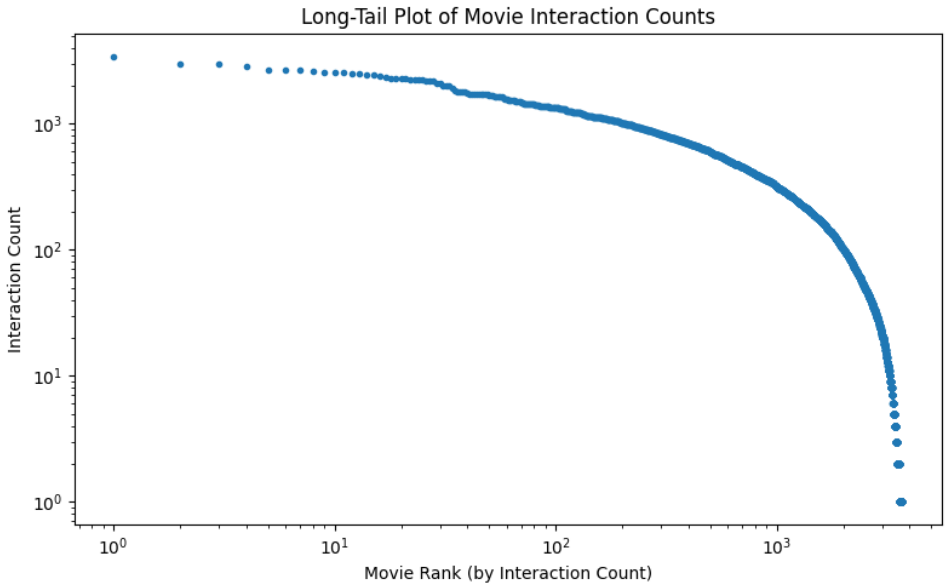
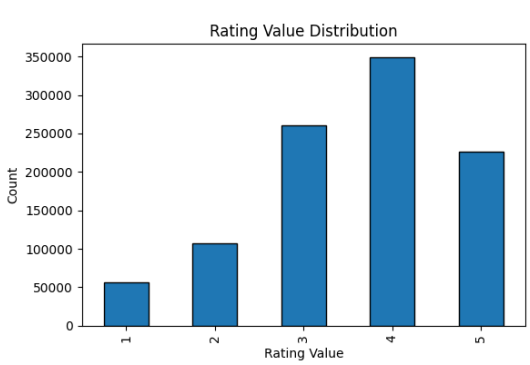
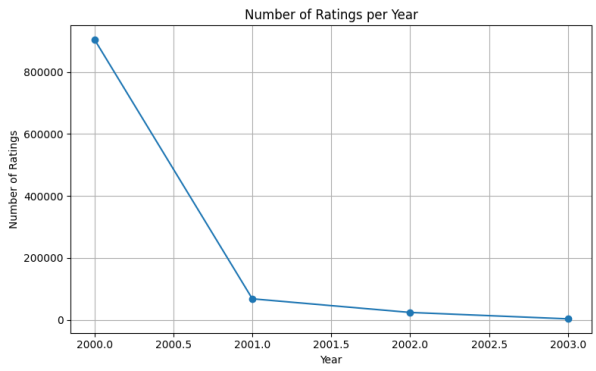


Figure 23: Item interaction statistics in MovieLens 1M: long-tail curve

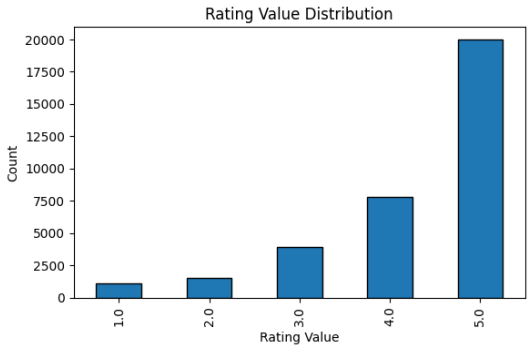


(a) Rating distribution in MovieLens 1M

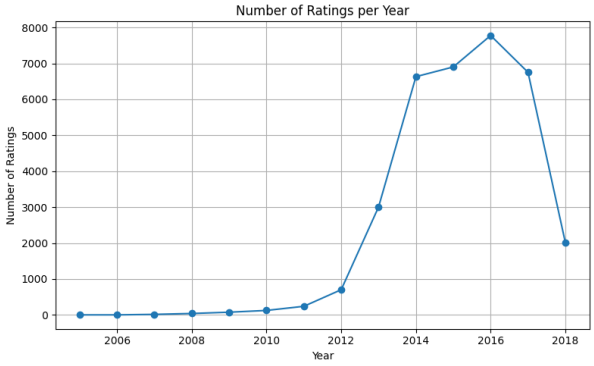


(b) Yearly rating volume trend in MovieLens 1M

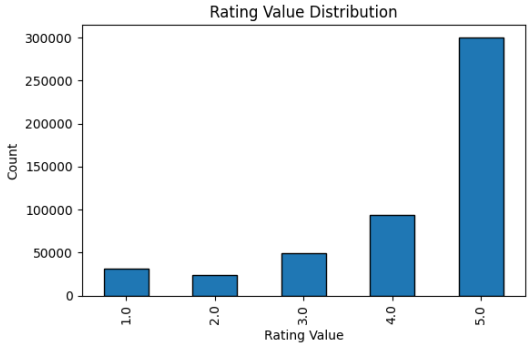
Figure 24: Rating statistics of MovieLens 1M: rating distribution and yearly rating volume trend



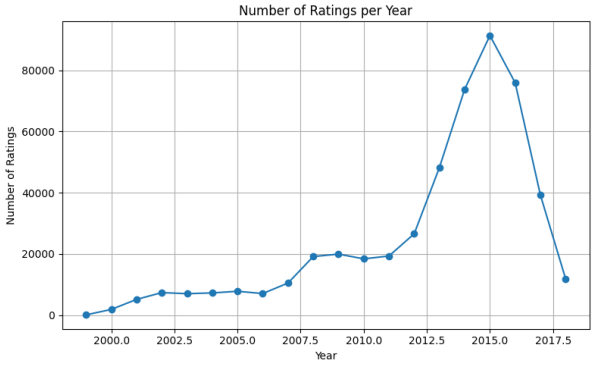
(a) Rating distribution in Amazon 2018 Luxury Beauty



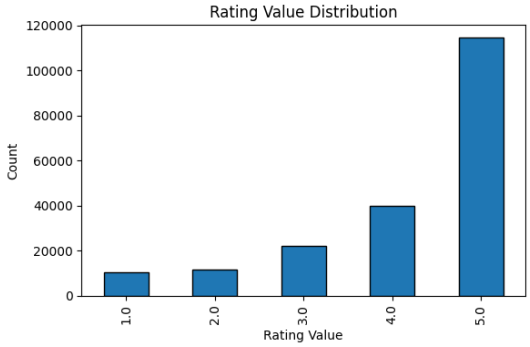
(b) Yearly rating volume Trend in Amazon 2018 Luxury Beauty



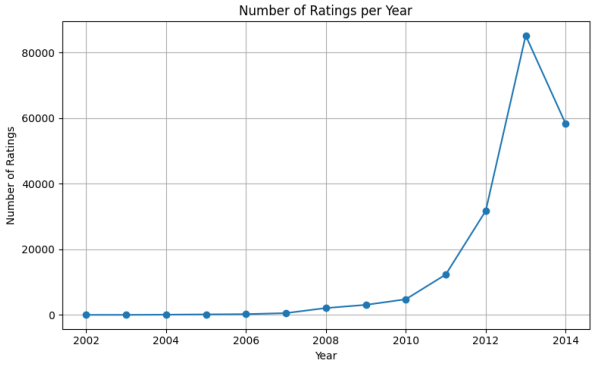
(c) Rating distribution in Amazon 2018 Video Games



(d) Yearly rating volume trend in Amazon 2018 Video Games

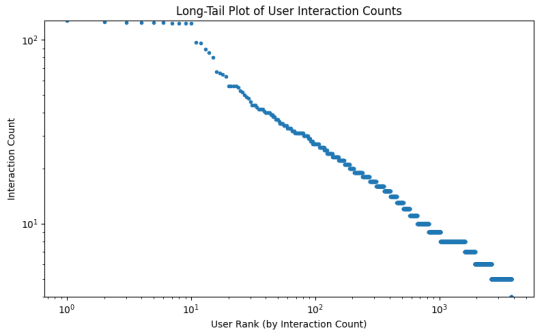


(e) Rating distribution in Amazon 2014 Beauty

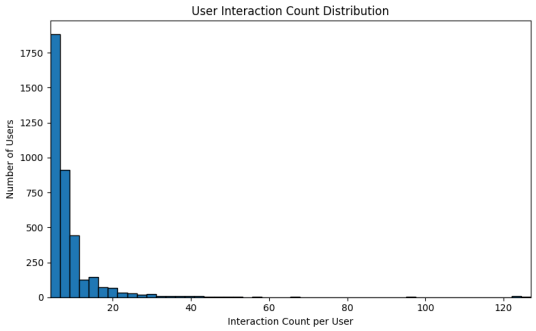


(f) Yearly rating volume Trend in Amazon 2014 Beauty

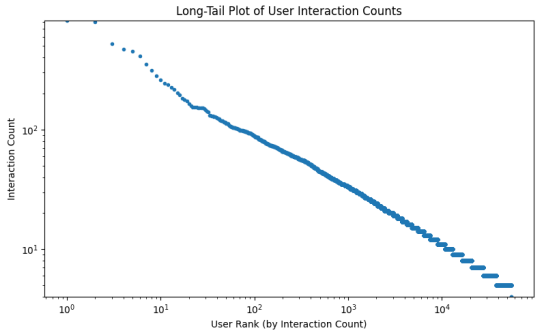
Figure 25: Rating statistics of Amazon 2018 Luxury Beauty, Amazon 2018 Video Games, and Amazon 2014 Beauty: rating distribution and yearly rating volume trend



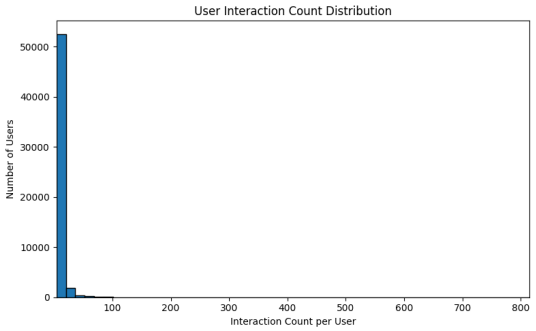
(a) User interaction statistics in Amazon 2018 Luxury Beauty: long-tail curve



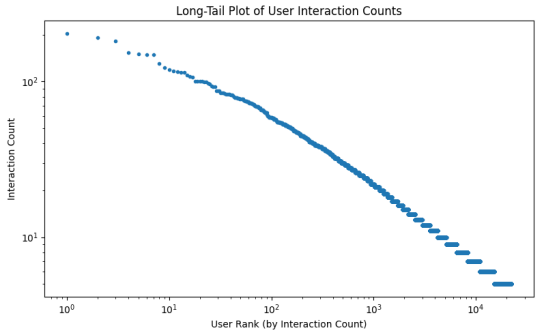
(b) User interaction statistics in Amazon 2018 Luxury Beauty: histogram



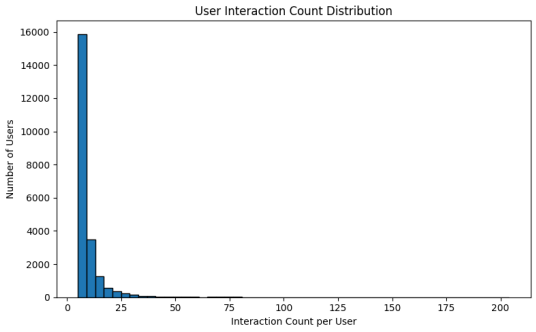
(c) User interaction statistics in Amazon 2018 Video Games: long-tail curve



(d) User interaction statistics in Amazon 2018 Video Games: histogram

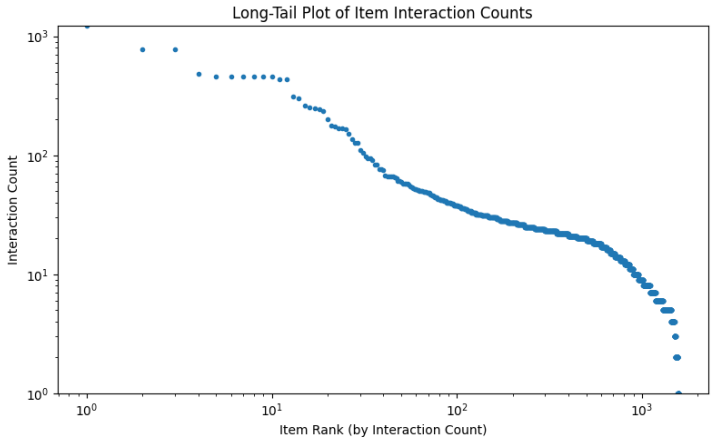


(e) User interaction statistics in Amazon 2014 Beauty: long-tail curve

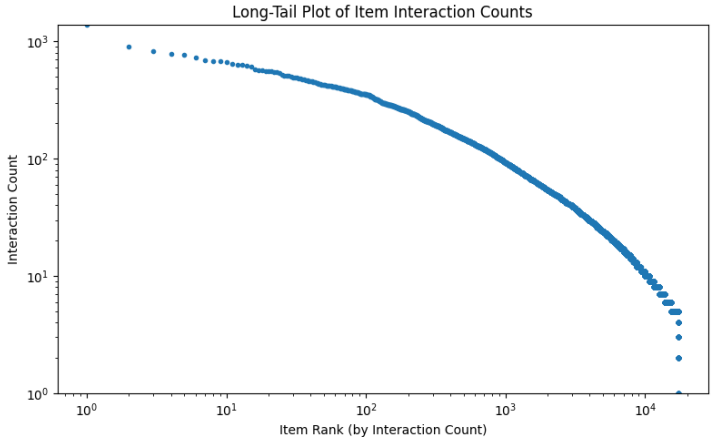


(f) User interaction statistics in Amazon 2014 Beauty: histogram

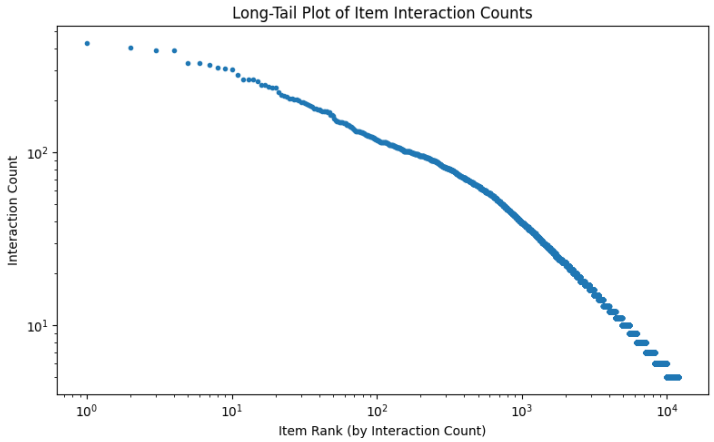
Figure 26: User interaction statistics in Amazon 2018 Luxury Beauty, Amazon 2018 Video Games, and Amazon 2014 Beauty Datasets: long-tail curve and histogram



(a) Item interaction statistics in Amazon 2018 Luxury Beauty: long-tail curve



(b) Item interaction statistics in Amazon 2018 Video Games: long-tail curve



(c) Item interaction statistics in Amazon 2014 Beauty: long-tail curve

Figure 27: Item interaction statistics in Amazon 2018 Luxury Beauty, Amazon 2018 Video Games, and Amazon 2014 Beauty Datasets: long-tail curve

References

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[2] Keqin Bao, Jizhi Zhang, Xinyu Lin, Yang Zhang, Wenjie Wang, and Fuli Feng. Large language models for recommendation: Past, present, and future. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2993–2996, 2024.

[3] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Chong Chen, Fuli Feng, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Transactions on Recommender Systems*, 2023.

[4] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014, 2023.

[5] Paul S Bradley, Kristin P Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

[6] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 378–387, 2021.

[7] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2010.

[8] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4274–4282, 2015.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

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

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.

[12] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item ids for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 195–204, 2023.

[13] Sanna Iivanainen, Jarkko Lagus, Henri Viertolahti, Lauri Sippola, and Jussi Koivunen. Investigating large language model (llm) performance using in-context learning (icl) for interpretation of esmo and nccn guidelines for lung cancer., 2024.

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

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[16] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022.

[17] Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. Large language models for generative recommendation: A survey and visionary discussions. *arXiv preprint arXiv:2309.01157*, 2023.

[18] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. *Advances in neural information processing systems*, 31, 2018.

[19] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 811–820, 2015.

[20] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*, pages 951–961, 2016.

[21] Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 365–374, 2024.

[22] Prem Melville, Raymond J Mooney, Ramadass Nagarajan, et al. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, 23:187–192, 2002.

[23] Haohao Qu, Wenqi Fan, Zihuai Zhao, and Qing Li. Tokenrec: learning to tokenize id for llm-based generative recommendation. *arXiv preprint arXiv:2406.10450*, 2024.

[24] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

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

[27] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112, 2015.

[29] Sumit Sidana, Mikhail Trofimov, Oleh Horodnytskyi, Charlotte Laclau, Yury Maximov, and Massih-Reza Amini. User preference and embedding learning with implicit feedback for recommender systems. *Data Mining and Knowledge Discovery*, 35:568–592, 2021.

[30] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013.

[31] Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis. Fair sequential group recommendations. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 1443–1452, 2020.

[32] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

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

[34] Jie Tang and Jing Zhang. A discriminative approach to topic-based citation recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 572–579. Springer, 2009.

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

[36] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. *Advances in neural information processing systems*, 26, 2013.

[37] Wen Wang, Wei Zhang, Jun Rao, Zhijie Qiu, Bo Zhang, Leyu Lin, and Hongyuan Zha. Group-aware long-and short-term graph representation learning for sequential group recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1449–1458, 2020.

[38] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2400–2409, 2024.

[39] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

[40] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 153–162, 2016.

[41] Quan Yuan, Gao Cong, and Chin-Yew Lin. Com: a generative model for group recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 163–172, 2014.

[42] Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems*, 2023.