Universiteit Leiden
The Netherlands

Opleiding Informatica

A Comparison between Traditional and LLM Recommendation System Performance

Chuck Yin Sin

Supervisors:
Dr. Zhaochun Ren & Jujia Zhao

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

16/01/2026

Abstract

Recommender systems are used to help navigate users through online services by providing personalized suggestions. The field of recommender systems is undergoing a significant transformation with the advent of Large Language Models (LLMs). Although traditional sequential recommendation models have significantly improved recommender systems, LLM-based models are beginning to become more prevalent. The use of LLMs allows the models to give recommendations with little to no input data. This thesis compares the performance of BERT4Rec and SASRec, two traditional sequential models, with BIGRec, a generative LLM-based model, on two unified datasets, MovieLens and Amazon Games. In order to measure the models, NDCG and Recall will be used as performance metrics. The results show that SASRec is the best performing model on average based on the performance metrics. The two traditional models show a big decrease in performance on the Games dataset, indicating that they struggle on less popularity biased datasets. The NDCG and Recall of BIGRec is poor compared to the traditional models on the MovieLens dataset. BIGRec was able to outperform the models when fully trained on the Games dataset. There was no significant decrease in performance on the Games dataset, indicating its property of not taking advantage of popularity bias. Further research should focus on fully training BIGRec and comparing the performance on larger datasets.

Contents

1 Introduction

Recent developments in technology and the increasing reliance on online services have resulted in recommender systems becoming an increasingly important part of digital platforms. Recommender systems help users navigate through the massive amount of online content by applying personalized filters to users to automatically suggest items that likely align with the interests of the user. Digital platforms such as streaming services, e-commerce, and gaming make use of recommender systems to provide users with relevant recommendations to meet customer satisfaction by giving users a personalized experience.

In the past, Traditional recommender systems used techniques such as collaborative filtering [WHC+16] (CF) and content-based filtering [MvS00] to recommend items to users. They use historical interactions between users and items to predict relevant items. Sequential Transformer [Vas23] models like BERT4Rec [SLW+19], and SASRec [KM18] further enhance the ability to recommend items by analyzing user behavior sequences [SLW+19].

More recently, Large Language Model-based [WLW+24] recommender systems have emerged as a new way to generate recommendations. These systems use Large Language Models (LLMs) with increased semantic understanding and contextual understanding. They also have a better understanding of natural language. This allows the systems to use zero-shot prediction [SVR21] and few-shot prediction [BMR+20] to recommend items to the user with little to no data. BIGRec [BZW+23] is an example of an LLM-based model.

Most studies have tended to focus on evaluating models using a limited set of configurations. As a result of this, it is difficult to make direct comparisons between the models.

This thesis focuses on evaluating the performance of two traditional recommendation models and one LLM-based recommendation model on two different datasets. One dataset has a significant bias towards popular items, and the other has a more balanced spread, with a better distribution between popular and less popular items, ensuring a weaker bias. The Normalized Discounted Cumulative Gain (NDCG) and the Recall of each model are the two performance metrics to be compared. The aim of this thesis is to provide insight into the strengths and weaknesses of each model relative to each other.

The developers of the three recommendation models have published their codes on GitHub. However, each model uses different datasets and has a unique approach to splitting the data. This thesis will analyze the models on the MovieLens-1M[1], and Amazon Video Games[2] datasets. The datasets will be divided into an 8:1:1 split, where 80% of the dataset is the training set, 10% is the testing set, and 10% is the validation set. Full ranking will be performed to ensure fairness. The results of the ranking are compared against each other to determine the best performing model.

[1] https://grouplens.org/datasets/movielens/1m/

[2] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

1.1 Thesis overview

Section 2 provides an overview of the existing research in the field of recommendation models and LLMs for recommendation systems. Section 3 will explain each of the three recommendation models used for the experiments. The two datasets used for the experiments and how the experiments are conducted will be shown in section 4. The results of the experiments will be interpreted in section 5 and discussed in section 6. Finally, the conclusion and possible future research is contained in section 7.

This thesis compares the NDCG and Recall metrics of traditional sequential and LLM-based recommendation models. This will be conducted on a strongly popularity biased and a lesser popularity biased dataset.

This Bachelor's thesis is supervised by Dr. Z. Ren and PhD student J. Zhao at the Leiden Institute of Advanced Computer Science (LIACS).

1.2 Research Question

The aim of this research is to answer the following research questions:

- What are the differences in ranking and retrieval performance between traditional and LLM-based models on a dataset with strong popularity bias?
- What are the differences in ranking and retrieval performance on a dataset with a lesser degree of popularity bias?

2 Related Work

Throughout the years, there have been many developments in recommendation models, with recent developments focusing on LLMs.

2.1 Collaborative Filtering

Historically, collaborative filtering (CF) has been one of the most popular approaches for recommendation systems [WHC+16]. By incorporating data from past user behavior into a rating feedback matrix, new ratings can be predicted with the help of CF. The relationship between users and items allows the recommender system to predict whether or not a user who did not interact with a certain item would enjoy interacting with that item based on the ratings of users who have similar ratings [RRK+25]. In addition, the recommender system will gain the ability to predict new items for the same user based on their existing ratings. One model that uses CF is CF-Diff [HPS24], a collaborative filtering method based on diffusion models.

However, better recommendation models have been developed more recently. These models are able to analyze user historical interactions in a sequential manner as opposed to CF models, where each interaction is separate.

2.2 Sequential Recommendation Models

Sequential recommendation models extend beyond traditional CF by explicitly modeling the temporal dynamics of user behavior. The aim is to provide the next item to the user by analyzing user behavior sequences [SLW+19]. The properties of the item, user, and behavior are stored for future reference By interacting with chronologically ordered items [PPW+24]. This allows the models to better predict the next item that a user is likely to engage with. Because of this, these models are capable of developing over time as they recognize that user interests shift over time. This is because, unlike its predecessors, sequential recommendation takes chronologically ordered user-item sequences as input.

There are a lot of sequential recommendation models, including BERT4Rec [SLW+19] and SASRec [KM18], which are used in the experiments of this research. Other models include the Markov Decision Processes-based recommender system [SBH15], a model that applies Markov chains to recommendation models. A Markov chain is a stochastic process that is used to describe systems that transition from one state to another in a sequence of steps [CLM12]. Another model is GRU4Rec [HKBT16], a session-based sequential recommendation model using Gated Recurrence Units (GRUs) to predict the next item in a user's session without requiring long-term user histories. This addresses the limitations caused by the sparse sequential data.

2.3 LLM-based Recommendation Models

As LLMs have developed substantially over the course of recent years, researchers began exploring the use of LLMs to generate recommendations [WLW+24]. By interpreting and generating natural language, these models are able to provide more personalized recommendations. They are also capable of adapting to the change in user interests in real-time. LLM-based recommendation models allow zero-shot prediction [SVR21], the ability to predict tasks, which is giving recommendations in our case, without the need to see an example of this during the training process. This is done using semantic information such as attributes, descriptions, and relationships present in known items. Few-shot prediction [BMR+20] is another feature of LLM-based models. It allows the model to perform tasks with only a small number of training examples. By averaging embeddings, comparing new instances to existing items, and self-fine-tuning, the model will be able to learn to perform the new task.

Although many LLM-based models have recently emerged, different parts of the models are targeted. LC-Rec [ZHL+23] aims to bridge the semantic mismatch by aligning the collaborative semantics with the language semantics of the LLM in a unified framework. LETTER, a LEarnable Tokenizer for generaTivE Recommendation [WBL+24] attempts to improve item tokenization by creating a learnable tokenizer to adaptively generate tokens that identify items. This model uses LC-Rec and TIGER, a Transformer Index for GEnerative Recommenders [RMS+23] to retrieve items.

3 Methodology

The aim of this research is to compare the performance between traditional and LLM-based recommendation models to determine which is better at generating recommendations.

3.1 Performance Metrics

The two metrics used to measure the performance of the recommendation models are NDCG and Recall.

3.1.1 NDCG

NDCG measures and ranks the items according to their relevance. The most relevant items are ranked at the top. The score is between 0 and 1, with 1 being a perfect ranking, where the items are all sorted correctly by relevance, while a score of 0 means that no item is relevant. To calculate the NDCG value for the top k recommendations, the Discounted Cumulative Gain (DCG) is divided by the Ideal Discounted Cumulative Gain (IDCG).

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{1}$$

In order to compute the DCG, the following formula is used:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2}$$

where k is the cutoff rank and rel_i is the relevance score of the item at position i. DCG is calculated by accumulating relevance scores of the recommended items while applying logarithmic discount to lower ranks. This causes lower ranked items to become less relevant. The IDCG is calculated by taking the maximum DCG at k. It is the maximum possible DCG when all items are perfectly ranked. NDCG will address the ranking performance of the research question, because both presence and ordering are important for the ranking performance. This evaluates the performance of giving the best recommendations possible.

3.1.2 Recall

Recall, also known as the True Positive rate, measures how many retrieved items are relevant. The score is also between 0 and 1, with 1 indicating that each retrieved item is relevant, while a score of 0 means that no retrieved item is relevant. The Recall value of the top k recommendations is calculated using the following formula:

$$\text{Recall@}k = \frac{|\{\text{relevant items}\} \cap \{\text{top-}k \text{ recommended items}\}|}{|\{\text{relevant items}\}|} \tag{3}$$

where k is the cutoff rank. The retrieval performance of the research question will be answered using the Recall metric due to Recall being determined by the proportion of relevant items retrieved. Because Recall disregards the positions and thus ordering of the items, it is primarily used to determine the coverage of each model.

3.2 BERT4Rec

BERT4Rec is a sequential recommendation model that uses a bidirectional transformer encoder to perform sequential recommendation tasks [SLW+19]. The aspect that makes BERT4Rec stand

out from other sequential models is that BERT4Rec uses bidirectional self-attention as opposed to left-to-right unidirectional architecture to model user behavior. This enables each item in the user sequence to take both past and future items into account.

The BERT4Rec model consists of multiple Transformer layers. In each layer, an output sequence is generated from an input sequence of the same length. Each Transformer layer consists of a Multi-Head Self-Attention sublayer and a Position-wise Feed-Forward Network sublayer. In the Multi-Head Self-Attention layer, the model weighs the importance of all other items in the same input sequence as the item it is trying to predict. It is able to look at both past and future items because of the bidirectional property. The output of this layer then passes through the Position-wise Feed-Forward Network layer to let the model learn non-linear relationships within each item embedding.

In order to train the model, a Masked Language Model (MLM), also known as the *Cloze* task [Tay53], is used [DCLT19]. By replacing a portion of the items in the input sequence with a `[MASK]` token, the model will be trained to predict the masked items based on the other items in the sequence. For inference, a `[MASK]` token will be appended to the last item of the input sequence. From here, the trained model will predict the masked item. The items with the highest probability will be the resulting recommendation.

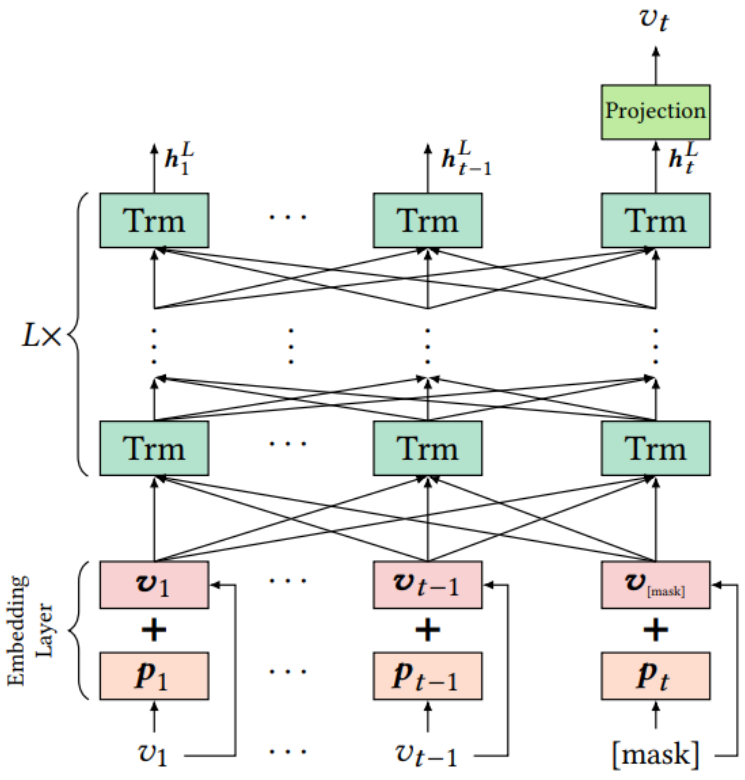

Figure 1: An illustration of the model architecture of BERT4Rec[SLW+19].

BERT4Rec will run calling `main.py` using the `train_bert` template [Chu20]. The template provides the model with the arguments used. In the case of this experiment, 5 and 50 epochs will be used during the training process. Before training, the input sequences have to be of the same length, as a Transformer processes fixed-length sequences. This will be done by padding shorter sequences. `[PAD]` is added until the sequence reaches the desired sequence length, which is 100 in this experiment. Some of the unpadded input will be masked by replacing them with the `[MASK]` token. The masked sequence will be processed in the model, starting with the embedding and transformer layers, and then the output corresponding to the masked positions is passed through the prediction layer to get logits for all possible items.

The output goes through a softmax function to convert the logits into a probability function, then the loss is calculated [SLW+19]. Afterwards, the model attempts to predict the next item by evaluating it on the test split. The last item of each input sequence will be replaced with a `[MASK]` token. The model then attempts to predict the last item. Instead of using negative sampling during the evaluation, the item will be compared against all possible items, and the NDCG and Recall scores are calculated.

3.3 SASRec

Unlike BERT4Rec, SASRec is a unidirectional Self-Attentive Sequential Recommendation model that only uses past items in an input sequence to generate a new item [KM18]. The model consists of an Embedding layer, where each item will be converted into a dense vector representation. A learnable position embedding is also used, which allows the model to become aware of the position of each item within its sequence. SASRec also has a Self-Attention layer. However, unlike BERT4Rec, it is unable to look at future items in the sequence because the attention mask is causal. The model looks at all previous items to predict the next item. Similarly to BERT4Rec, the output will be passed through a Feed-Forward Network once it is finished. Afterwards, residual connections and layer normalization will be applied to the output.

The self-attention blocks use the Scaled Dot-Product Attention [VSP+23] formula in Formula 4 with inputs described in Formula5.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{4}$$

Q denotes the queries, K denotes the keys, and V denotes the values of an item. Each item is represented with a row. The formula uses $\frac{1}{\sqrt{d_k}}$ to prevent extremely large dot products due to high values of d_k.

$$S = \text{SA}(\hat{E}) = \text{Attention}\left(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V\right), \tag{5}$$

where the projection matrices $W^Q, W^K, W^V \in \mathbb{R}^{d\times d}$. These projection matrices are converted using linear projections on the input embedding $\hat{E}$. This allows the model to learn asymmetric interactions due to its increased flexibility.

Once the model has finished computing the output from the self-attention blocks, it will continue in the prediction layer, where the next item will be predicted. A ranking will be created based

on the relevance of each item. The most relevant items will have the highest ranking. This ranking representation will then be used to predict the next item.

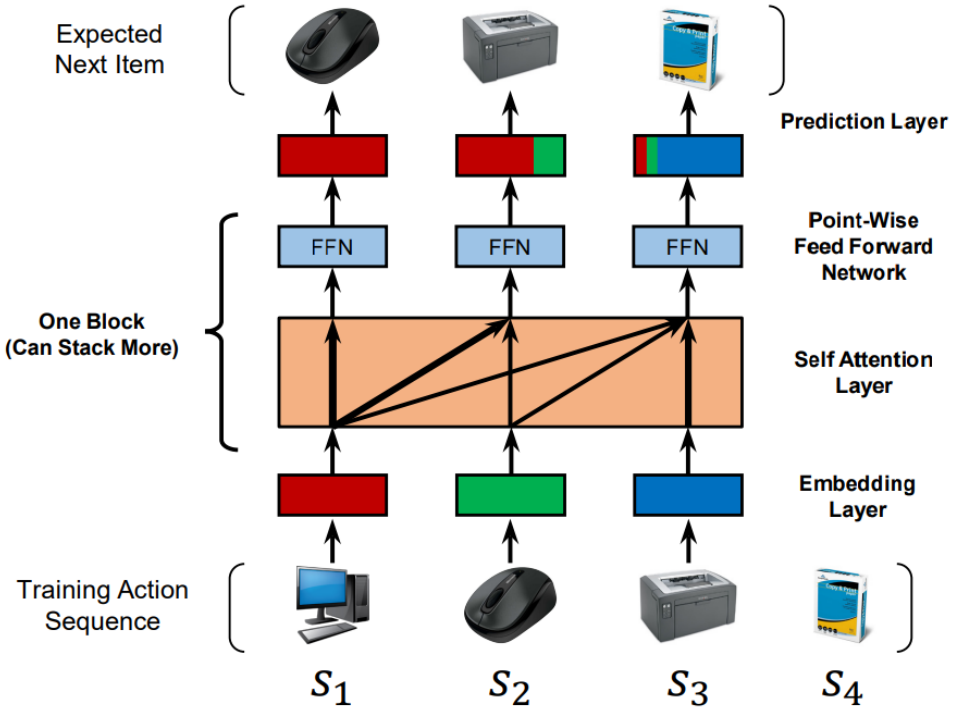

Figure 2: A simplified illustration of the training process of SASRec[KM18].

For SASRec [KM18], the input sequences also have to be padded. 5 and 50 epochs will be used during the training process. The maximum length of an input sequence is set to 200, and the dropout_rate will be set to 0.2 [Hua20]. There will be two stacked Transformer blocks and one attention head in the Self-Attention layer. The model will go through the stacked Transformer layers and the Self-Attention layer before predicting the output inside of the Prediction layer. SASRec uses binary cross-entropy (BCE) loss with negative sampling. For each positive user-item interaction in a training sequence, a set of negative items is randomly sampled from the entire item catalog. The model is then trained to predict a score of 1 for positive items and 0 for negative items.

$$-\sum_{S^u \in \mathcal{S}} \sum_{t \in [1,2,\ldots,n]} \left[\log(\sigma(r_{o_t,t})) + \sum_{j \notin S^u} \log(1 - \sigma(r_{j,t})) \right] \tag{6}$$

Formula 6 [KM18] represents the loss function, where $\sigma(\cdot)$ is the sigmoid function, $r_{o_t,t}$ is the predicted relevance score for the ground truth item o_t at timestep t, and j represents a negative item sample not in the input sequence S^u.

During the training process, the model will attempt to predict the next item $i + 1$ for each item i in the sequence. The BCE loss will be computed and the model will evaluate on the test split.

The last item will be predicted for each input sequence. Since the experiments use full ranking, negative sampling is replaced, and the NDCG and Recall will be computed by comparing against all items.

3.4 BIGRec

BIGRec is a Bi-step Grounding Paradigm for Recommendation model [BZW+23] that aims to recommend actual items that exist in the real world. This is achieved by grounding the language space to the recommendation space, which then gets grounded to the actual item space. The model will first perform instruction tuning to restrict the output from the language space. This must be done because the language space contains all possible language sequences that the model can generate. The model will be fine-tuned to generate a recommendation based on a user's past interactions with items. Thus, the model will not generate output irrelevant to what it is trained for.

However, not all generated recommendations exist in the real world. Sometimes, the model might recommend items that are not part of the actual item space. Therefore, the output must be grounded once more, from the recommendation space to the actual item space. The generated items will be mapped to real-world items before the latent representations are extracted. Afterwards, a similarity search will be performed to find the actual items most similar to the generated ones. Then a ranking is decided on the basis of similarity and popularity, and the final recommendation will be given.

For the ranking of BIGRec, it uses Formula 7 [BZW+23] to calculate the Euclidean distance (L2 distance) between the embeddings of the actual items.

$$D_i = \|\text{emb}_i - \text{oracle}\|_2 \tag{7}$$

emb_i represents the embedding of the i-th item, and oracle is the embedding of the output generated by the LLM. BIGRec then defines popularity and collaborative information. This is calculated in Formula 8 [BZW+23].

$$\begin{cases} C_i = \dfrac{N^i}{\sum_{j \in I} N^j}, \\ P_i = \dfrac{C_i - \min_{j \in I}\{C_j\}}{\max_{j \in I}\{C_j\} - \min_{j \in I}\{C_j\}}. \end{cases} \tag{8}$$

N is the set of user-item interactions in the training data, N^j is the number of observed interactions for item j in N, $\mathcal{I}$ is the set of all items, C_i is the popularity of the i-th item, and P_i is the normalized value of C_i. By incorporating a modified version of Popularity-bias Deconfounding and Adjusting (PDA) [ZFH+21], the Euclidean distance is reweighted using Formula 9 [BZW+23].

$$\begin{cases} \hat{D}_i = \dfrac{D_i - \min_{j \in \mathcal{I}}\{D_j\}}{\max_{j \in \mathcal{I}}\{D_j\} - \min_{j \in \mathcal{I}}\{D_j\}}, \\ \tilde{D}_i = \dfrac{\hat{D}_i}{(1 + P_i)^\gamma}, \end{cases} \tag{9}$$

D_i is the Euclidean distance between the i-th item embedding and the embedding of the outputs generated by the LLMs, $\hat{D}_i$ is the normalized D_i, and $\tilde{D}_i$ reweights $\hat{D}_i$ using popularity. Inverse popularity and a hyperparameter γ are used for the reweighting of $\hat{D}_i$.

In order to successfully run BIGRec [BZW+23], the items and users have to be embedded. Recommendation-specific instruction tuning will be performed to ground the language space to the recommendation space. The loss function will be a BCE with negative sampling.

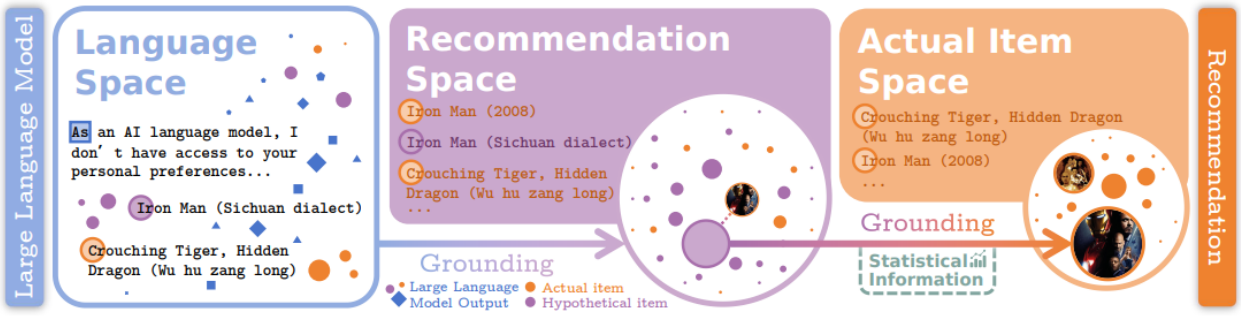

Figure 3: An illustration of the grounding process of BIGRec[BZW+23].

The model will be trained with its epoch count set to 5 and 50 and a maximum input sequence length of 512. Few-shot training will be utilized, since 1024 and 2048 samples will be used to train the model. During training, each sequence will construct an input prompt based on the history of the user. The next item of each sequence will be generated, and the loss will be calculated between the actual item tokens and the generated item tokens. Once this is done, inference will be performed to evaluate the performance of the model. The generated items will be converted into vector embeddings, which will be compared with the vector embeddings of the actual items. A similarity search will be performed to determine whether or not a generated item exists. The generated list with the top ranked items will be compared to the next item in the test set to calculate the NDCG and Recall.

4 Experiment Setting

4.1 Datasets

For the experiments, two datasets are used. The first dataset used is MovieLens-1M[3]. This dataset has a significantly higher bias towards popular items and will answer the first research question. The Amazon Video Games[4] dataset has a lesser degree of popularity bias. It contains a better distribution between popular and unpopular items. This is shown using Figure 4.1. There is a

[3] https://grouplens.org/datasets/movielens/1m/
[4] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

substantial drop in the interaction ratio in popularity groups 6 to 9 in the MovieLens dataset, whereas the Amazon Games dataset has a better distribution. This indicates a noticably weaker popularity bias for the Amazon Games dataset, as the users interact with a broader set of items compared to the MovieLens users. Using these two datasets allows the experiments to determine whether popularity bias affects the models.

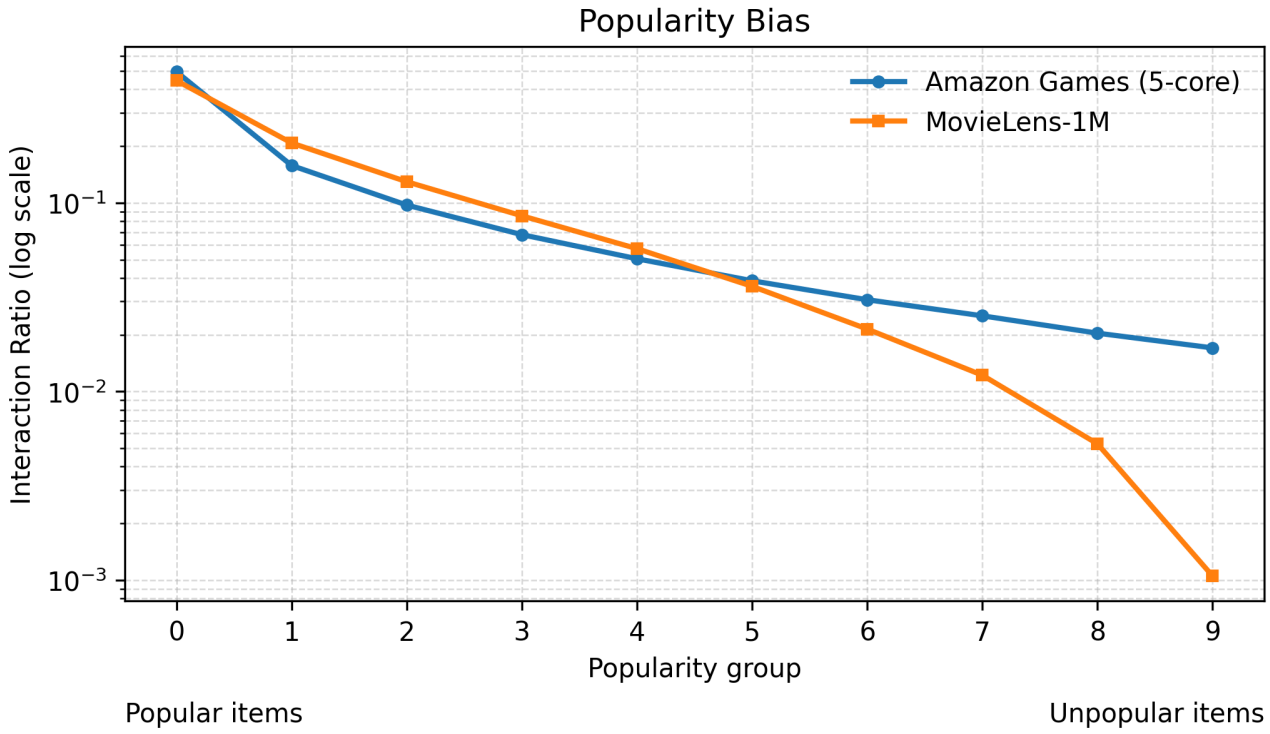

Figure 4: The popularity distribution of the two datasets

The datasets used in the experiments are already split into usable parts. For the MovieLens-1M dataset, the movies, ratings, and tags are already divided into their respective `.dat` files. The Amazon Video Games dataset used is the 5-core subset. The datasets will be split into an 8:1:1 ratio, and the test and validation sets will be limited to the first 5000 items, because of the slow runtime from BIGRec. Each recommendation model is modified to use the same data split.

4.2 Evaluation Method

All models will provide the Recall and NDCG evaluation metrics based on 5 epochs and 50 epochs on both datasets. BIGRec will use few-shot prediction using 1024 training samples as opposed to the entire training set, with the exception of the Games dataset with 5 epochs. This is because BIGRec has very long training times. BIGRec will also be trained using 2048 samples to compensate for the inability to train fully. Because the performance of full ranking are compared as opposed to regular recommendation, BERT4Rec and SASRec are modified to perform full ranking.

4.3 Implementation Details

All experiments will be conducted using Python, PyTorch, and various packages required for the models to operate. The specific versions used are shown in the following table. BIGRec requires a

Model	Python	PyTorch	Torchvision	CUDA Toolkit
BERT4Rec	3.6.13	1.5.0	0.6.0	10.1.243
SASRec	3.7.1	1.6.0	0.7.0	10.2.89
BIGRec	3.9.21	2.5.1	0.20.1	11.7.0

Table 1: Conda environment configurations for BERT4Rec, SASRec, and BIGRec.

slightly different set of packages compared to the `requirements.txt` file provided by the GitHub repository of BIGRec[5]. The differing package versions are:

- accelerate 1.4.0
- bitsandbytes 0.39.0
- peft 0.17.0
- transformers 4.49.0
- scipy 1.13.1
- numpy 2.0.2

Due to version incompatibilities between the required packages and the Leiden SSH servers, BERT4Rec and SASRec will run on an NVIDIA Geforce GTX 1660 Ti GPU on a local device, while BIGRec will run on 2 NVIDIA Geforce RTX 3090 on the vibranium.liacs.nl server. All models will run with seed 42 and will have their own conda environment. BERT4Rec and SASRec will be trained using the commands provided by their repositores. Conversely, BIGRec uses a shell script to run training and inference. The meta-llama/Llama-2-7b-hf[6] model will be used as the base model. Furthermore, a learning rate of 1e−4 will be used to train the datasets. The lora hyperparameters and the cutoff length remain unchanged.

5 Experiment Results

The results of the experiments will be provided and analyzed in Section 5.1 and Section 5.2. In order to compare the performances of the models, two different epoch counts are used to determine whether the epoch counts influence the performance of the traditional and LLM-based models relative to each other. In Section 5.1.1 and Section 5.2.1, the results of 5 epochs on the datasets will be provided. Section 5.1.2 and Section 5.2.2 will contain the results of the performance of 50 epochs on the datasets. Due to very long training times, it was only possible to run BIGRec fully trained on the Games dataset with 5 epochs.

[5] https://github.com/SAI990323/BIGRec
[6] https://huggingface.co/meta-llama/Llama-2-7b-hf

5.1 Performance Comparison on a Strongly Biased Dataset (RQ1)

The first experiment compares the performance of the models on a strongly popularity biased dataset. The MovieLens-1M[7] dataset is used to conduct this experiment.

5.1.1 5 Epochs

From the results of the three models on the MovieLens-1M dataset as depicted in Table 2, SASRec can be observed as the model with the highest performance. It consistently outperforms the other models across all metrics, achieving the highest Recall and NDCG values. BIGRec using few-shot prediction is the worst performer. Using 2048 training samples improved the performance of BIGRec by a significant amount, but it was still less than 50% of the performance of BERT4Rec.

Dataset	Metric	BERT4Rec	SASRec	BIGRec (1024)	BIGRec (2048)
ML-1M	Recall@5	0.0246	0.0523	0.0068	0.0104
	Recall@10	0.0468	0.0854	0.0108	0.0206
	Recall@20	0.0832	0.1412	0.0172	0.0350
	NDCG@5	0.0153	0.0338	0.0039	0.0068
	NDCG@10	0.0224	0.0445	0.0052	0.0100
	NDCG@20	0.0315	0.0584	0.0068	0.0136

Table 2: Performance comparison of the models on a strongly popularity biased dataset (5 epochs)

5.1.2 50 Epochs

When the models are run with 50 epochs instead of 5, the Recall and NDCG performance metrics are shown to have an increase of 100% on average in the traditional models, whereas BIGRec (1024) shows a bigger difference, with an increase of up to 436% (NDCG@5). Table 5.1.2 shows that SASRec has the highest performance on the MovieLens-1M dataset. BIGRec demonstrates a stronger performance compared to its performance with 5 epochs when run with 1024 training samples. In contrast to this, the performance of BIGRec with 2048 training samples is slightly worse than 1024 samples. Relative to the traditional models, BIGRec (2048) significantly underperformed compared to the performance with 5 epochs.

[7] https://grouplens.org/datasets/movielens/1m/

Dataset	Metric	BERT4Rec	SASRec	BIGRec (1024)	BIGRec (2048)
ML-1M	Recall@5	0.0540	0.1068	0.0184	0.0168
	Recall@10	0.0922	0.1654	0.0268	0.0232
	Recall@20	0.1612	0.2495	0.0382	0.0344
	NDCG@5	0.0324	0.0705	0.0144	0.0128
	NDCG@10	0.0445	0.0892	0.0171	0.0148
	NDCG@20	0.0618	0.1104	0.0199	0.0177

Table 3: Performance comparison of the models a strongly popularity biased dataset (50 epochs)

5.2 Performance Comparison on a Lesser Biased Dataset (RQ2)

The second experiment compares the performance of the models on a weaker popularity biased dataset. The Amazon Video Games[8] dataset is used to conduct this experiment.

5.2.1 5 Epochs

From the results of the three models on the Amazon Games dataset as depicted in Table 4, BIGRec can be observed as the model with the highest performance when fully trained. When only few-shot prediction from BIGRec is considered, BIGRec still outperforms the traditional models on both NDCG and Recall with the exception of Recall@20.

Dataset	Metric	BERT4Rec	SASRec	BIGRec (1024)	BIGRec (2048)	BIGRec (Full)
Games	Recall@5	0.0094	0.0141	0.0056	0.0178	0.0292
	Recall@10	0.0168	0.0226	0.0098	0.0254	0.0344
	Recall@20	0.0302	0.0389	0.0166	0.0354	0.0444
	NDCG@5	0.0066	0.0086	0.0038	0.0116	0.0253
	NDCG@10	0.0089	0.0113	0.0052	0.0141	0.0270
	NDCG@20	0.0123	0.0154	0.0069	0.0166	0.0296

Table 4: Performance comparison of the models on a weaker popularity biased dataset (5 epochs)

For the traditional models, there is a significant drop in both Recall and NDCG metrics on the Games dataset compared to the MovieLens-1M dataset. In contrast, the difference in the performance of BIGRec on both datasets is not as great. There is a slight decrease in Recall when BIGRec is run with 1024 training samples, and an increase in performance across all metrics when it is run with 2048 training samples.

5.2.2 50 Epochs

When the models are run with 50 epochs instead of 5, the Recall and NDCG performance metrics once again have an average increase of 100% in the traditional models, whereas BIGRec (1024) shows a bigger difference. Table 5.1.2 shows that SASRec is the best performing model on the

[8] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

Amazon Games dataset. BIGRec demonstrates a similar performance to a fully trained BIGRec with 5 epochs. The Recall and NDCG values on the Games dataset for BIGRec are comparable to those of SASRec with the exception of Recall@20, where SASRec is shown to have a 56.5% increase over BIGRec (1024). BIGRec (2048) shows similar performance to that of BIGRec (1024).

Dataset	Metric	BERT4Rec	SASRec	BIGRec (1024)	BIGRec (2048)
Games	Recall@5	0.0236	0.0252	0.0262	0.0242
	Recall@10	0.0372	0.0410	0.0324	0.0310
	Recall@20	0.0616	0.0670	0.0428	0.0414
	NDCG@5	0.0154	0.0154	0.0204	0.0205
	NDCG@10	0.0197	0.0204	0.0224	0.0227
	NDCG@20	0.0259	0.0269	0.0251	0.0253

Table 5: Performance comparison of the models on a weaker popularity biased dataset (50 epochs)

For the traditional models, the significant drop in both Recall and NDCG metrics on the Games dataset compared to the MovieLens-1M dataset is still present when using 50 epochs. In contrast, BIGRec shows a greater performance on the Games dataset, with a higher NDCG and Recall on every metric.

6 Discussion

This thesis aims to analyze the performance of BERT4Rec and SASRec, two traditional sequential recommendation models, with BIGRec, an LLM-based recommendation model. The models are run on the MovieLens-1M, a strongly popularity biased dataset, and Amazon Video Games 5-core, a dataset with weaker popularity bias. 5 epochs and 50 epochs are used in the experiments and the Recall and NDCG metrics are computed by the models.

6.1 Evaluation of the results

From the results of the experiments discussed in Section 5, there are a variety of observations to be made. The results with 5 epochs indicate that SASRec is the best performing model on the MovieLens-1M dataset, followed by BERT4Rec, with BIGRec having the lowest Recall and NDCG scores by a good margin.

For the Games dataset, BIGRec has the highest performance if the model is fully trained, and the highest NDCG scores with 2048 training samples if only few-shot prediction is considered. When the models are run with 50 epochs, SASRec significantly outperforms the other models on the MovieLens-1M dataset. This is likely due to SASRec being specifically designed to predict the next item. It is also the least complex model of the three, allowing it to reach its maximum results in an earlier epoch. BIGRec is significantly more complex, so it is possible that 50 epochs is not enough for BIGRec to reach its best results using few-shot prediction with 1024 and 2048 training samples. The NDCG metrics of BIGRec on the Games dataset are similar to those of SASRec, but

its overall performance is the weakest. In both experiments, the performance of the traditional models is significantly lower on the Games dataset compared to the MovieLens-1M dataset.

MovieLens-1M contains significantly more popular items than the Games dataset, which has a better balance between popular and unpopular items. Both BERT4Rec and SASRec are sequential models, which allow the capture of strong short-term sequential patterns around frequently interacted items. This results in a notable increase in performance compared to the less biased Games dataset. Conversely, BIGRec grounds items from the language space, to the recommendation space, and then to the actual item space. It does not notice the pattern and it does not take advantage of the popularity bias accordingly.

This is also the reason why BIGRec has similar scores across both datasets. With the Games dataset, there is substantially less popularity bias for the sequential models to capitalize on. This results in BERT4Rec and SASRec having lower metrics compared to the MovieLens-1M dataset. This allows BIGRec to achieve similar results despite having only a fraction of training samples using few-shot prediction.

6.2 Limitations

There are several limitations that hinder the evaluation of the experiments. Firstly, it is only possible to run BIGRec fully on the Games dataset with 5 epochs using this experiment setting. All other settings are unable to be computed due to the extremely long training time of BIGRec. By observing the fully trained BIGRec metrics on the Games dataset, it is entirely possible for BIGRec to outperform BERT4Rec and SASRec on both datasets if BIGRec is able to train using all training samples.

Although experiments were conducted with 2048 training samples to attempt to compensate for this, the results show very similar, and in most metrics, slightly lower performance compared to 1024 training samples when the models are training with 50 epochs. The higher epoch count likely causes a training sample size increase of 100% to be insignificant. This indicates that a small increase in training samples only shows benefits when there are fewer complete passes through the training sets.

Secondly, both test and validation sets contain only 5000 samples instead of everything. Using the full test and validation sets would result in a better representation of the experiments. This could result in BIGRec performing better than its current performance. This was not feasible due to the long inference and evaluation times of BIGRec. Furthermore, the experiments are performed on relatively small datasets. MovieLens-1M is used instead of MovieLens-10M due to the slow runtime of BIGRec during the research process.

7 Conclusions and Further Research

This thesis focuses on comparing the performances of BERT4Rec and SASRec, two traditional sequential recommendation models, with BIGRec, an LLM-based recommendation model, on

MovieLens-1M and Amazon Video Games, two public benchmark datasets with varying degrees of popularity bias. The data is divided into 80% training, 10% testing, and 10% validation sets. The models are run with 5 epochs and 50 epochs to get the Recall and NDCG metrics. For BIGRec, few-shot prediction is used due to the computational cost. From the results, it can be concluded that BIGRec has a higher performance ceiling than the traditional models when the dataset contains a lower degree of popularity bias. When the runtime of the models is considered, SASRec is the best performing model. The effectiveness of traditional and LLM-based recommendation models is compared in the conducted experiments.

Further research should aim to compare the performance of the models on larger datasets. Due to limitations of the hardware used in the experiments, it was not possible during this research. Comparison of the performance metrics of the fully trained BIGRec model is also a point of investigation. BIGRec has consistently outperformed both traditional models when fully trained on the Games dataset with 5 epochs. This could open the possibility of BIGRec outperforming the traditional models in all experiments. Lastly, using Retrieval-Augmented Generation (RAG) [LPP+20] to enhance the performance of the models is also a topic that could be focused on. RAG enhances the quality and relevance of the generated data by gathering information from specified documents relevant to the data.

References

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[BZW+23] 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. *arXiv preprint arXiv:2308.08434*, 2023.

[Chu20] Jae-Won Chung. Pytorch implementation for bert4rec, 2020.

[CLM12] Ka Chan, C. Lenard, and Terence Mills. An introduction to markov chains. 12 2012.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[HKBT16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016.

[HPS24] Yu Hou, Jin-Duk Park, and Won-Yong Shin. Collaborative filtering based on diffusion models: Unveiling the potential of high-order connectivity, 2024.

[Hua20] Zan Huang. Pytorch implementation for sasrec, 2020.

[KM18] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *arXiv preprint arXiv:1808.09781*, 2018.

[LPP+20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladislav Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[MvS00] Robin Meteren and Maarten van Someren. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age (MLnet/ECML2000) Workshop*, 2000.

[PPW+24] Liangwei Pan, Weiqing Pan, Meng Wei, Hongzhi Yin, and Zhong Ming. A survey on sequential recommendation. *arXiv preprint arXiv:2412.12770*, 2024.

[RMS+23] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval, 2023.

[RRK+25] Shaina Raza, Mizanur Rahman, Safiullah Kamawal, Armin Toroghi, Ananya Raval, Farshad Navah, and Amirmohammad Kazemeini. A comprehensive review of recommender systems: Transitioning from theory to practice, 2025.

[SBH15] Guy Shani, Ronen I. Brafman, and David Heckerman. An mdp-based recommender system, 2015.

[SLW+19] 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*, CIKM '19, pages 1441–1450, New York, NY, USA, 2019. ACM.

[SVR21] Damien Sileo, Wilhelmus Vossen, and Roel Raymaekers. Zero-shot recommendation as language modeling. *arXiv preprint arXiv:2112.04184*, 2021.

[Tay53] Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953.

[Vas23] Shazeer N. Parmar N. Uszkoreit J. Jones L. Gomez A. N. Kaiser L. Polosukhin I. Vaswani, A. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2023.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[WBL+24] Wenqi Wang, Hang Bao, Xiao Lin, Junlin Zhang, Yilei Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. *arXiv preprint arXiv:2405.07314*, 2024.

[WHC+16] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, 69, 10 2016.

[WLW+24] Qi Wang, Jindong Li, Shiqi Wang, Qianli Xing, Runliang Niu, He Kong, Rui Li, Guodong Long, Yi Chang, and Chengqi Zhang. Towards next-generation llm-based recommender systems: A survey and beyond, 2024.

[ZFH+21] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 11–20. ACM, July 2021.

[ZHL+23] Beichen Zheng, Yifan Hou, Hongyin Lu, Yifan Chen, Wayne Xin Zhao, Min Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. *arXiv preprint arXiv:2311.09049*, 2023.