Enhancing Large Language Model Based Sequential Recommender Systems with Pseudo Labels Reconstruction

Hyunsoo Na[1], Minseok Gang[1], Youngrok Ko[1], Jinseok Seol[2], Sang-goo Lee[1,3]

[1]Seoul National University, [2]Korea Advanced Institute of Science and Technology

[3]IntelliSys, Korea

{monchana, alstjr3754, yrko1, sglee}@europa.snu.ac.kr

jsseol@kaist.ac.kr

Abstract

Large language models (LLMs) are utilized in various studies, and have demonstrated potential to function independently as a recommendation model. However, training on user-item interaction sequences and additional textual information such as reviews often modifies the pre-trained weights of LLMs, diminishing their inherent strength in constructing and comprehending natural language sentences. In this study, we propose a **re**construction-based **L**LM **rec**ommendation model (ReLRec) that harnesses the feature extraction capability of LLMs, while preserving LLMs' sentence generation abilities. We reconstruct the user and item pseudo-labels generated from user reviews while training on sequential data, aiming to exploit the key features of both users and items. Experimental results demonstrate the efficacy of label reconstruction in sequential recommendation tasks.

1 Introduction

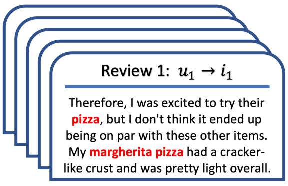
Recommender systems have achieved significant advancement, becoming essential in various domains such as e-commerce, streaming services, and social media. Despite their widespread application, traditional methods face several limitations, particularly in extracting and effectively utilizing textual information, such as user reviews, in addition to interaction data. Traditional models, (Kang and McAuley, 2018; Jannach and Ludewig, 2017; Sun et al., 2019; Ma et al., 2019; Tang and Wang, 2018), while effective in leveraging numerical data such as ratings and purchase history, often struggle with capturing the nuanced contextual information present in user reviews and other textual data. This limits their ability to fully understand user preferences and provide highly personalized recommendations.

Large language models (LLMs), like GPT (Brown et al., 2020), have transformed the field of recommender systems by excelling in understanding and generating natural languages. They effectively address the limitations of traditional models in handling textual data and are actively studied for their potential in various recommendation tasks. Several efforts have leveraged LLMs for zero/few-shot recommendation by incorporating user history and candidate items as input prompt (Zheng et al., 2023; Zhu et al., 2024; Zhao et al., 2024). Additionally, LLMs have been employed to address cold-start problem (Xi et al., 2023; Wang et al., 2024b) and have been utilized for data augmentation purposes (Wei et al., 2024; Ning et al., 2024; Ren et al., 2024). Moreover, studies that have aimed to train LLMs via efficient measures (Li et al., 2023a,b; Yu et al., 2024; Kaur and Shah, 2024) have demonstrated their potential to function as recommendation models.

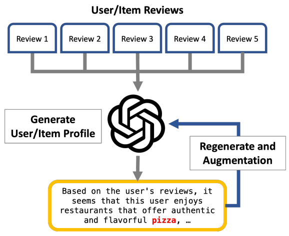
LLM-based recommender systems leverage pre-trained knowledge to understand diverse textual inputs and generate rich textual representations (Acharya et al., 2023; Wang et al., 2024a). This allows LLMs to provide contextually relevant suggestions that align closely with user preferences. However, training LLMs on recommendation datasets can disrupt their pre-trained weights, especially their generation capabilities (Li and Hoiem, 2017; Luo et al., 2023). Therefore, new methods are needed to use LLMs effectively while preserving their strengths.

Our proposed methodology addresses such challenges by leveraging LLMs' advanced textual representation generation in a sequential recommendation framework. To preserve the pre-trained knowledge of the LLM, we implemented separate embeddings for users and items while freezing the LLM's transformer layers and the word embedding. This ensures that the model retains its ability to construct coherent sentences. To train user and item embedding, we generated pseudo-labels for users and items based on their reviews. Finally, we

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 7213–7222



(a) User's review on an item.



(b) User/item pseudo-label generation.

Figure 1: Pseudo-label generation. (a) refers the review from a user to an item and (b) shows how reviews are concatenated to generate pseudo user/item label.

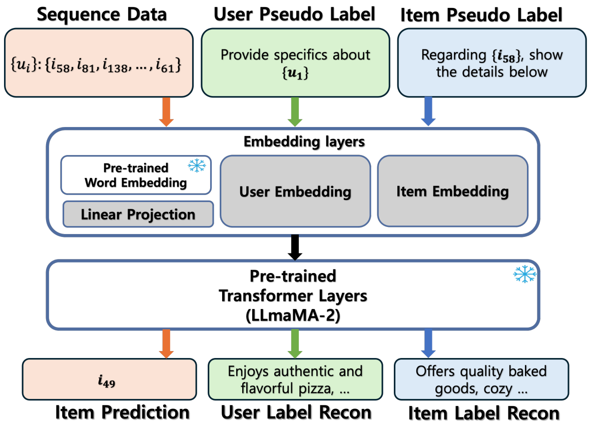


Figure 2: Illustration of ReLRec, sequential recommendation with label reconstruction.

trained the model using both sequential data and the pseudo-labels simultaneously.

The main contributions of this paper are: (1) proposing a **re**construction-based **LLM** model (ReLRec) that captures and effectively integrates user/item label features for next item predictions; (2) introducing a methodology that leverages the feature extraction capabilities of LLMs while preserving their inherent sentence generation abilities; and (3) demonstrating that encapsulating rich contextual information in labels and integrating it into sequential recommendations enhance the model's ability to understand and predict user preferences.

2 Proposed Approach

In this study, we propose ReLRec, a model composed of two primary components: the creation of user/item embeddings with pseudo-labels, and the training process that incorporates both sequential data and textual labels.

2.1 Pseudo-label Generation

We employed the RLMRec profile generation method (Ren et al., 2024) to create pseudo-labels for both users and items. As depicted in Fig. 1, we first aggregated and concatenated reviews associated with each item to construct an item profile. Similarly, user profiles were generated by concatenating the reviews submitted by each user. These concatenated reviews were subsequently passed to ChatGPT (i.e., gpt-3.5-turbo), which was utilized to produce labels for both items and users. Furthermore, the initially generated labels were reprocessed through ChatGPT to obtain variations in different formats. To capture diverse and meaningful features in the labels, we generated multiple label versions, all maintaining consistent contextual meaning, and randomly selected one for each training iteration. This approach ensures that the model is exposed to varied yet coherent label representations during training.

Dataset	#User	#Item	#Inter.	#Avg.	Sparsity
Yelp	11,092	11,011	321,581	29.0	99.737%
Book	10,830	9,333	211,909	19.6	99.790%

Table 1: Dataset statistics. **#User**, **#Item**, **#Inter.**, **#Avg.** denote the number of users, items, total interactions, and average user interactions respectively.

2.2 RelRec Architecture

LLM-based sequential recommender systems tend to overlook the LLM's capability to construct coherent sentences, focusing primarily on analyzing sequential data. The implementation of fine-tuned transformer layers and adapters to capture sequential patterns frequently results in a diminished capacity to generate relevant sentences and accurately capture contextual meaning. ReLRec leverages the capabilities of LLMs to comprehend and utilize textual information, incorporating rich, nuanced data encapsulated in pseudo-labels generated from user reviews. As shown in Figure 2, we implemented separate embeddings for users and items, initialized with the average of the word token embedding weights. A unique token is assigned to each user/item id (i.e., "iid-1" is a single token) to represent the attributes of each user/items. With the transformer layers and word embeddings frozen to preserve the LLM's inherent ability to understand and generate natural language, we added a projection layer to each embedding to map and integrate the pre-trained knowledge with newly trained text labels.

Models	Yelp						Amazon-book					
	R@5	**N@5**	**R@10**	**N@10**	**R@20**	**N@20**	**R@5**	**N@5**	**R@10**	**N@10**	**R@20**	**N@20**
STAMP	0.0232	0.0145	0.0393	0.0196	0.0673	0.0267	0.0699	0.0564	0.0908	0.0631	0.1217	0.0709
HGN	0.0229	0.0140	0.0433	0.0205	0.0764	0.0288	0.0356	0.022	0.0633	0.0309	0.1082	0.0422
GRU4Rec	0.0269	0.0161	0.0504	0.0237	0.0881	0.0331	0.0646	0.0441	0.0989	0.0552	0.1473	0.0673
SASRec	0.0210	0.0135	0.0386	0.0190	0.0730	0.0276	0.0624	0.0355	0.0959	0.0463	0.1443	0.0585
BERT4Rec	0.0153	0.0093	0.0283	0.0135	0.0540	0.0200	0.0380	0.0267	0.0585	0.0333	0.0869	0.0405
NARM	0.0272	0.0165	0.0527	0.0246	0.0910	0.0341	0.0650	0.0458	0.0947	0.0553	0.1424	0.0673
CL4SRec	0.0277	0.0171	0.0449	0.0226	0.0793	0.0312	0.0396	0.0235	0.0595	0.0299	0.0888	0.0372
ICLRec	0.0181	0.0117	0.0295	0.0154	0.0485	0.0202	0.0352	0.0236	0.0520	0.0290	0.0779	0.0356
P5	0.0240	0.0150	0.0390	0.0198	0.0587	0.0247	0.0318	0.0230	0.0445	0.0271	0.0629	0.0318
E4SRec	0.0180	0.0102	0.0541	0.0212	0.0902	0.0306	0.0369	0.0317	0.0831	0.0467	**0.1570**	0.0650
Ours	**0.0338**	**0.0208**	**0.0599**	**0.0292**	**0.1048**	**0.0406**	**0.0758**	**0.0570**	**0.1011**	**0.0651**	0.1377	**0.0741**
Improvement	24.3%	26.1%	10.7%	18.7%	15.2%	19.1%	8.4%	1.1%	2.2%	3.2%	-12.3%	4.5%

Table 2: Performance comparison of sequential recommendation models. **R** stands for Recall, and **N** refers to NDCG. **Bold** indicates the best result, while the underline is the runner-up.

2.3 Label-based Recommendation

ReLRec simultaneously trains on sequential interaction and user/item labels, ensuring that the context and features captured in the labels are reflected in the model's next item prediction. Each batch comprises of the sequential interaction and label of a user, along with a randomly selected item label.

Suppose I and U denote entire set of items and users. Let $i^k_{1:t} = \{i^k_1, \ldots, i^k_t\}$ represent the user-item interaction sequence, where $i^k_p \in I$ is the item interacted with by user u^k at timestamp p, and $u^k \in U$. The goal is to predict the next item i^k_{t+1} by training on $i^k_{1:t}$. For the next item prediction, we utilized cross-entropy loss of our backbone LLM.

$$\mathcal{L}_{\text{seq}} = -\sum_{k=1}^{|U|} \frac{1}{T_k} \sum_{t=1}^{T_k} \log P(i_{t+1} \mid i_{1:t}, u^k). \quad (1)$$

Here, T is the total number of time steps for each user. The labels of each user and item are reconstructed along with the next item prediction. Similarly, with S referring the entire set of word tokens, the label reconstruction aims to predict the token s_{t+1} given tokens $s_{1:t} = \{s_1, \ldots, s_t\}$.

$$\mathcal{L}_{\text{item}} = -\sum_{i \in I} \frac{1}{T_i} \sum_{t=1}^{T_i} \log P(s^i_{t+1} \mid s^i_{1:t}), \quad (2)$$

$$\mathcal{L}_{\text{user}} = -\sum_{u \in U} \frac{1}{T_u} \sum_{t=1}^{T_u} \log P(s^u_{t+1} \mid s^u_{1:t}), \quad (3)$$

where s^u and s^i denotes tokens within the labels of user u and item i, respectively. The overall learning objective function of ReLRec is the sum of the losses for the next item prediction and user/item label reconstruction.

$$\mathcal{L} = \alpha\mathcal{L}_{\text{seq}} + \beta\mathcal{L}_{\text{item}} + \gamma\mathcal{L}_{\text{user}}, \quad (4)$$

where α, β and γ are the loss weights for each task. During inference, we evaluate the model not only on sequential prediction, but also on user/item label reconstruction, to ensure that the model has learned the textual labels of users and items. We provide a randomly selected prompt to the model to reconstruct the user/item label. The Appendix A.1 provides example labels.

3 Experiment

3.1 Experimental Settings

Datasets and metrics We conducted experiments on two public datasets: **Yelp** (Asghar, 2016) and **Amazon-book** (McAuley et al., 2015). Each dataset includes user's reviews on items that user interacted with. We filtered out interactions with a rating below 3 and excluded users with less than 5 interactions. The statistics of each dataset are provided in Table 1. We evaluated recommendation performance using two widely adopted ranking metrics: *Recall@k* and *NDCG@k* with $k \in \{5, 10, 20\}$.

Baselines We compared our model with ten baseline sequential recommendations: STAMP (Liu et al., 2018), HGN (Ma et al., 2019), GRU4Rec (Tan et al., 2016), SASRec (Kang and McAuley, 2018), BERT4Rec (Sun et al., 2019), NARM (Li et al., 2017), CL4SRec (Xie et al., 2022), ICLRec (Chen et al., 2022), P5 (Geng et al., 2022) and E4SRec (Li et al., 2023a). STAMP, HGN, GRU4Rec, SASRec, BERT4Rec and NARM experiments are conducted using RecBole v1.2.0 (Xu et al., 2023).

Setup ReLRec uses Llama-2-7b (Touvron et al., 2023) as the backbone model. Dimension for each

Methods			Yelp		Book	
User	PL	Recon.	R@20	N@20	R@20	N@20
X	**X**	-	0.0764	0.0302	0.1086	0.0609
X	**O**	-	0.0750	0.0299	0.1044	0.0556
O	**X**	**X**	0.0727	0.0289	0.1035	0.0592
O	**O**	**X**	0.0755	0.0302	0.1049	0.0586
O	**O**	**O**	**0.0969**	**0.0371**	**0.1309**	**0.0706**
Improvement			28.3%	22.8%	20.5%	15.9%

Table 3: Comparing methods for using user embedding (**User**), projection layer (**PL**) and reconstruction loss (**Recon.**). All embeddings are randomly initialized.

projection layer is set to 512 for all datasets. α, β and γ are set to 0.75, 0.3 and 0.9, respectively. ReLRec was trained with RTX A6000.

3.2 Performance Comparison

Table 2 presents the experiment results of ReLRec compared to sequential recommendation baselines. Our model outperforms baseline models in most metrics, by up to 24.3% in *Recall* and 26.1% in *NDCG*. The performance increase is highlighted especially in Yelp dataset. By incorporating contextual label information into the sequential recommendation, ReLRec suggests items that are similar to the labels of user and answer item. Additional examples and analysis are in the Appendix A.2.1.

3.3 Ablation and Effectiveness Analysis

Reconstruction loss We conducted an analysis to evaluate the effectiveness of our model, specifically examining whether implementing the user embedding and label reconstruction loss enhance performance. As shown in Table 3, simply adding user embedding to train each user did not necessarily improve results. However, including label reconstruction loss significantly boosted performance, indicating that encapsulating textual information in the labels and integrating it into the sequence prediction was effective. Effect of text projection layer is in the Appendix A.2.2.

Label inconsistency To evaluate the robustness of our reconstruction-based method, we conducted experiments analyzing the impact of label inconsistency on model performance. Noise was introduced by switching item and user labels across datasets, and the model's ability to learn from these inconsistent labels was assessed. As shown in Table 4, even with the introduction of label noise, the reconstruction method demonstrated notable improvements compared to scenarios without reconstruction. Although label inconsistency led to a decline in performance relative to consistent-label conditions, the model still captured meaningful patterns, underscoring the resilience and effectiveness of the label reconstruction approach under suboptimal conditions.

Methods			Yelp		Book	
Recon	Item	User	R@20	N@20	R@20	N@20
X	**O**	**O**	0.0842	0.0372	0.1004	0.0417
O	**X**	**O**	0.0973	0.0377	0.1374	0.0718
O	**O**	**X**	0.1026	0.0382	0.1353	0.0711
O	**O**	**O**	0.1048	0.0406	0.1377	0.0741

Table 4: Effect of label inconsistency. X in Item and User column indicates that the label of dataset is switched.

Methods			Yelp		Book	
Recon.	Avg.	2-stage.	R@20	N@20	R@20	N@20
X	**X**	-	0.0755	0.0302	0.1049	0.0586
X	**O**	-	0.0842	0.0327	0.1004	0.0417
O	**X**	**O**	0.0969	0.0371	0.1309	0.0706
O	**O**	**X**	0.0972	0.0361	0.1242	0.0682
O	**O**	**O**	0.1048	0.0406	0.1377	0.0741

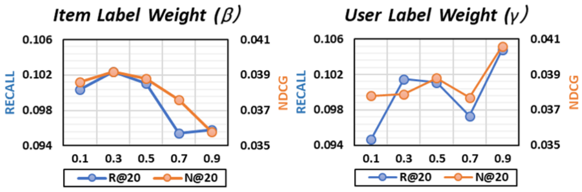
Table 5: Effect of average embedding initialization (**Avg.**) and 2-stage pseudo-label generation (**2-stage.**) for Yelp and Amazon-book datasets.

Pseudo-label Quality To examine the effect of pseudo-label quality on performance, we conducted experiments refining the label generation process with a two-stage method. In the first stage, user and item profiles were generated by aggregating relevant reviews to produce initial labels. In the second stage, these labels were refined by removing noise. Notably, even with the one-stage process, our reconstruction method showed significant improvements in overall performance despite the presence of noise. As shown in Table 5, applying the two-stage label generation further enhanced performance, demonstrating the added value of refined labels, with single-stage process already providing substantial gains.

Embedding initialization We initialized the user and item embeddings in our model with the average value of pre-trained LLM word embedding. Table 5 compares the effectiveness of this average initialization both with and without reconstruction loss. The results show that average initialization was only consistently effective when reconstruction loss was included. We believe this is because, as ReLRec trains on user/item textual labels, the

Metrics	Sequence Weight α			
	0.25	**0.50**	**0.75**	**1.00**
R@20	0.0856	0.1000	**0.1000**	0.0956
N@20	0.0337	0.0390	**0.0396**	0.0373

Table 6: Performance variation with different sequence weight α value on Yelp dataset.



(a) Item Recon. Loss Weight (b) User Recon. Loss Weight

Figure 3: Recall/NDCG@20 on change of β and γ.

average value of the pre-trained word embeddings provides a better initialization point compared to random initialization.

Weight parameter To evaluate the impact of each loss hyperparameters, we conducted experiments with different values of hyperparameters on Yelp dataset. Table 6 shows the performance variations with different values of next item prediction loss weight α. The results indicate that an α value of 0.75 yields the best performance. This suggests that a balanced emphasis on sequential interactions and label reconstruction is crucial in optimal performance. Increasing α beyond this point leads to a decline in performance, likely due to overemphasizing sequential data at the expense of label information.

Fig. 3 illustrates the performance changes as the item label weight β and user label weight γ vary, respectively. Initially, increasing β improves performance as the item label information integrates into the sequential pattern. However, beyond a certain point, further increases in β cause performance to drop significantly, likely because excessive weight on item labels hinders the item embedding's ability to predict the next item. This is evident from the decline in both Recall and NDCG metrics as shown in Figure 3a.

For γ, the performance trends differently. As shown in Figure 3b, the best performance is observed at the highest value of γ. This indicates that a higher weight on user labels effectively enhances the model's ability to capture user-specific preferences.

Dataset	#User	#Item	#Inter.	#Avg.	Sparsity
Steam	23,311	5,238	596,560	113.9	99.511%

Table 7: Dataset statistics of Steam dataset.

4 Conclusion

In this paper, we proposed ReLRec, a reconstruction based LLM recommendation model which integrates features of user/item text labels to the next item prediction. Our proposed model, ReLRec, effectively utilizes pseudo-labels generated from user reviews to capture nuanced information and enhance recommendation accuracy by label reconstruction. Experimental results demonstrates the effectiveness of ReLRec.

Acknowledgments

This work was partly supported by the National Research Foundation of Korea (NRF) [No. RS-2023-00243243] and Institute of Information & communications Technology Planning & Evaluation (IITP) [NO. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], both grant funded by the Korea government (MSIT). Also, the ICT at Seoul National University provided research facilities for this study.

Limitation

We conducted additional experiments on Steam (Kang and McAuley, 2018) dataset and analyzed our results. The experiments on Steam indicated that the performance of the label reconstruction-based approach may dependent on the quality of the labels, and it might not lead to performance improvements on certain conditions.

Experimental Settings

Statistics about Steam dataset are provided in Table 7. Similar to Yelp and Amazon-book, Steam dataset includes user reviews on items that users have interacted with. However, Steam features a much higher user-to-item ratio, a significantly larger average number of interactions per user, and a much higher total number of interactions. Moreoever, unlike Yelp and Amazon-book, Steam does not provide user ratings for each item. Therefore, we could not filter interactions based on ratings and only excluded users with fewer than 5 interactions. The evaluation metrics, baselines, and setup are

Models	Steam			
	R@10	N@10	R@20	N@20
STAMP	0.1613	<u>0.1317</u>	0.1924	0.1395
HGN	0.0840	0.0443	0.1261	0.0548
GRU4Rec	0.1732	0.1301	0.2176	0.1413
SASRec	**0.1814**	**0.1392**	**0.2277**	**0.1508**
BERT4Rec	0.1288	0.0871	0.1767	0.0991
NARM	<u>0.1775</u>	0.1311	<u>0.2256</u>	<u>0.1432</u>
Ours	0.1521	0.1135	0.2002	0.1255
Improvement	-16.2%	-18.5%	-12.1%	-16.8%

Table 8: Performance comparison of sequential recommendation models on Steam dataset.

Methods			Steam			
User	PL	Recon.	R@10	N@10	R@20	N@20
X	X	X	0.1443	0.1108	0.1842	0.1209
O	X	X	0.1424	0.1077	0.1811	0.1174
O	O	X	0.1398	0.1060	0.1789	0.1160
O	O	O	**0.1534**	**0.1143**	**0.2001**	**0.1260**
Improvement			6.3%	3.2%	8.6%	4.2%

Table 9: Comparing methods for using user embedding (**User**), projection layer (**PL**) and reconstruction loss (**Recon.**). All embeddings are randomly initialized.

identical to those used for Yelp and Amazon-book dataset.

Experiment Analysis

Table 8 shows the performance comparison between our model and baseline models on Steam dataset. We can observe that our model did not achieve the best or the second-best results in any of the metrics on Steam dataset.

We conducted a similar analysis to that of Yelp and Amazon-book datasets to evaluate the effectiveness of the reconstruction loss on Steam. The results are shown in Table 9. The performance improvement on Steam was much smaller compared to Yelp and Amazon-book.

Label quality and noises The inclusion of a high volume of interactions without filtering the dataset can introduce noises. Since Steam dataset does not have rating, all reviews were included in the training. This means the model may be learning from labels and interactions that are not truly indicative of user preferences, reducing the effectiveness of the label reconstruction-based approach.

Limitation Conclusion

The performance of ReLRec model on Steam dataset underscores the challenges of handling high interaction volumes and the need for mechanisms that can effectively differentiate and manage low-quality interactions. Future work should focus on enhancing the model's ability to understand and mitigate the impact of negative or irrelevant interactions. By improving the model's capacity to handle diverse interaction data, we can enhance its robustness and overall performance in high-interaction environments like Steam dataset.

References

Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. Llm based generation of item-description for recommendation system. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, 1204–1207.

Nabiha Asghar. 2016. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362* (2016).

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

Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*. 2172–2182.

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

Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 306–310.

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

Kirandeep Kaur and Chirag Shah. 2024. Efficient and Responsible Adaptation of Large Language Models for Robust Top-k Recommendations. *arXiv preprint arXiv:2405.00824* (2024).

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

Lei Li, Yongfeng Zhang, and Li Chen. 2023b. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1348–1357.

Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023a. E4SRec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443* (2023).

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

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1831–1839.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747* (2023).

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

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52.

Lin Ning, Luyang Liu, Jiaxing Wu, Neo Wu, Devora Berlowitz, Sushant Prakash, Bradley Green, Shawn O'Banion, and Jun Xie. 2024. User-LLM: Efficient LLM Contextualization with User Embeddings. *arXiv preprint arXiv:2402.13598* (2024).

Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3464–3475.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

Jianling Wang, Haokai Lu, James Caverlee, Ed H Chi, and Minmin Chen. 2024b. Large Language Models as Data Augmenters for Cold-Start Item Recommendation. In *Companion Proceedings of the ACM on Web Conference 2024*. 726–729.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2020. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *arXiv preprint arXiv:2012.15828* (2020).

Xinfeng Wang, Jin Cui, Yoshimi Suzuki, and Fumiyo Fukumoto. 2024a. RDRec: Rationale Distillation for LLM-based Recommendation. *arXiv preprint arXiv:2405.10587* (2024).

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 806–815.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933* (2023).

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *2022 IEEE 38th International Conference on*

Data Engineering (ICDE). IEEE Computer Society, 1259–1273.

Lanling Xu, Zhen Tian, Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2023. Towards a More User-Friendly and Easy-to-Use Benchmark Library for Recommender Systems. In *SIGIR*. ACM, 2837–2847.

Xiaohan Yu, Li Zhang, Xin Zhao, Yue Wang, and Zhongrui Ma. 2024. RA-Rec: An Efficient ID Representation Alignment Framework for LLM-based Recommendation. *arXiv preprint arXiv:2402.04527* (2024).

Jujia Zhao, Wenjie Wang, Chen Xu, Zhaochun Ren, See-Kiong Ng, and Tat-Seng Chua. 2024. LLM-based Federated Recommendation. *arXiv preprint arXiv:2402.09959* (2024).

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Adapting large language models by integrating collaborative semantics for recommendation. *arXiv preprint arXiv:2311.09049* (2023).

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*. 3162–3172.

A Appendix

A.1 Pseudo Labels

This section provides additional examples related to the generation and training of pseudo labels. Fig. 4 shows examples of user and item input prompts and their answer labels. A random prompt is selected from the list of prompts, and inserted into the model along with a user/item token. During evaluation, the similarity score between the reconstructed label and the answer label is measured using *cosine similarity*. The similarity in user and item label reconstruction was assessed using "all-MiniLM-L6-v2" model (Wang et al., 2020).

User Reconstruction
Input: Regarding *uid_1*, below are the details.
Answer: This user has a diverse palate and enjoys exploring new a nd unique dining experiences. They are drawn to traditional bakeri es and ice cream parlors, as well as casual and lively bars with a wi de selection of food and drinks....

Item Reconstruction
Input: Specifics about *iid_1* are provided here.
Answer: With a focus on speed and affordability, *iid_1* attracts cust omers who enjoy the efficiency of a fast-food joint but crave the a bility to customize their meals, even if it means accepting...

Figure 4: Examples of user/item reconstruction.

Methods	Yelp		Similarity	
	R@20	N@20	Item	User
w/o proj.	0.0995	0.0385	0.5261	0.7507
1-layer	**0.1048**	**0.0406**	0.4538	0.7211
2-layer	0.1039	0.0394	0.0875	0.3579

Table 10: Comparing methods for using linear projection attached to word token embedding.

A.2 Further Analyses

A.2.1 Sequential Prediction Results

Based on our experiments with the Yelp and Amazon-book datasets, most of our performance metrics exceeded those of the baseline models. We examined the user labels and the labels of the top N recommended items. Fig. 5 shows example labels for users, the correct answer items and candidate items, included in the top-5 recommendations.

From these examples, we can assume that the contextual information of both the user and the items is effectively integrated into the sequential prediction results. ReLRec suggests items that closely match the labels of the users and the correct answer items. The suggested candidates had labels that closely resembled those of the answer items and aligned well with the user's preferences.

This alignment indicates that ReLRec is capable of capturing and utilizing rich textual information to enhance recommendation accuracy. By integrating both user and item labels into the prediction process, ReLRec is able to provide recommendations that are more likely to match the user's preferences, achieving higher rankings compared to models that only utilize sequential data.

A.2.2 Text Projection Layer

To preserve the knowledge of LLMs, we froze the word embedding and transformer layers and examined the impact of adding a projection layer. As Table 3 demonstrates, adding projection layer was effective when ReLRec trains textual labels. Additionally, we investigated the optimal number of layers required for the model to effectively train on the contextual information of users/items. As detailed in Table 10, using more than one layer resulted in a decline in performance. This suggests that adding more layers can disrupt the model's ability to reconstruct textual information and degrade overall performance.

We also conducted experiment to determine the optimal projection layer dimension for ReLRec's

User-1: User enjoys Italian delis, American classics, high-quality burgers, creative soups, generous deli portions, seafood, and friendly service.
Answer item: Mamma Italia's is a casual restaurant in Eagle with diverse menu, generous portions, cozy ambiance, and friendly staff.

C1: Upscale grill with classic American dishes, efficient service, trendy atmosphere, and pet-friendly outdoor dining.
C2: Classic Italian dining in Philadelphia with unassuming ambiance, generous portions, and excellent service in a nostalgic atmosphere.
C3: Las Palmas offers casual, affordable Tex-Mex with friendly service, quick service, and popular queso, although some find food mediocre.
C4: *iid_9* in Safety Harbor offers high-end wines, build-your-own charcuterie boards, convenient credit card system, cozy atmosphere, and attentive staff.

User-2: User seeks high-quality, diverse dining experiences with fresh ingredients, variety of options, cozy ambiance, and attentive service.
Answer item: Modern, trendy atmosphere with diverse menu, quality food/drink, outdoor seating, attention to detail, budget-friendly, innovative brunc h, well-cooked meats/sauces.

C1: Cozy and lively atmosphere with varied menu, wide beer selection, great service, steak and prime rib specialties.
C2: *iid_99* offers diverse pizzas, extensive beer selection, cozy atmosphere, and excellent service for pizza and beer lovers.
C3: *iid_88* in New Orleans offers house beers, bar food, and a lively atmosphere, popular with locals and tourists.
C4: Versatile venue with good food and large portions, suitable for families, sports fans, and casual dining, but prices can be high.

User-3: Enjoys unique dining experiences, flavorful food, generous portions, attentive staff, diverse choices, and specialty/organic options.
Answer item: Offers diverse menu, attentive service, and pleasant atmosphere, suitable for casual or special occasions.

C1: *iid_2* offers interactive dining with a wide variety of food options, ideal for meat and seafood lovers in a casual upscale setting.
C2: Popular Philly spot offering delicious breakfast/brunch dishes, attentive service, endless coffee, and cozy atmosphere at reasonable prices.
C3: Casual Latin American restaurant specializing in Guatemalan food, with vegetarian options, variety of flavors, and friendly atmosphere.
C4: Popular comfort food spot in New Orleans' French Quarter with diverse crowd, offering cheap eats, breakfast, burgers, and friendly service.

Figure 5: Example labels for user, answer item and candidate item (**C**) labels from Yelp dataset. Underlined words are features that overlaps with the user and the answer item.

Dimension	Yelp R@10	Yelp N@10	Yelp R@20	Yelp N@20
64	0.0393	0.0198	0.0657	0.0264
128	0.0514	0.0255	0.0865	0.0344
256	0.0620	0.0300	0.1034	0.0401
512	0.0599	0.0292	0.1048	0.0406
1024	0.0530	0.0256	0.0899	0.0349
2048	0.0487	0.0233	0.0889	0.0334
4096	0.0455	0.0220	0.0793	0.0304

Table 11: Comparing dimensional size of linear projection on Yelp dataset.

Model	Inference Speed
Llama2-seq	0.2521 batch/s
Ours	0.2579 batch/s

Table 12: Comparison on inference speed of Llama2 and ReLRec.

embedding, ranging from 64 to 4096 dimensions. 11 presents the performance results on the Yelp dataset across different dimension sizes. The results demonstrate that a dimension size of 512 achieved the highest performance. While increasing the dimension size from 64 to 512 led to improved performance, further increasing it beyond 512 resulted in diminishing returns. This suggests that while larger dimensions provide more capacity, they may reduce model's ability to generalize effectively. Consequently, we selected 512 as the optimal dimension size for the projection layer in our final model configuration.

A.3 Inference

In ReLRec, the pre-trained word embeddings and transformer layers remain frozen, with the main modifications being the addition of a linear projection layer, user embedding, and item embedding. Consequently, as shown in Table 12,the inference speed of our model closely matches that of Llama2-seq, which refers to recommendation prediction using Llama2-7b (Touvron et al., 2023) with only the item embedding attached. The inference experiments were conducted on a single NVIDIA RTX A6000 with a batch size of 16. This minimal difference in inference speed highlights the efficiency of our approach, allowing for the potential application of existing techniques to further accelerate the process. Future work will focus on exploring additional methods to optimize inference without altering the core transformer architecture.