Item-Language Model: Improving Large Language Model for Recommendation via Item-Language Representation Learning

Li Yang[1 2], **Anushya Subbiah**[2], **Hardik Patel**[2], **Judith Yue Li**[2], **Yanwei Song**[2]
Reza Mirghaderi[2], **Vikram Aggarwal**[2], **Fuli Feng**[3], **Zenglin Xu**[4]
Dongfang Liu[5], **Qifan Wang**[1]*
[1]Meta AI, [2]Google Research, [3]University of Science and Technology of China
[4]Fudan University, [5]Rochester Institute of Technology

Abstract

Large Language Models (LLMs) have recently made significant advancements in tackling complex tasks, such as retrieving hard-to-find information and solving intricate problems. Consequently, various approaches have been proposed to integrate LLMs into recommender systems, primarily by embedding them within existing architectures or training them on the recommendation data. However, most existing methods fail to effectively incorporate user-item interaction signals into pretrained LLMs due to the modality gap between interaction data and the LLM's internal knowledge. To address this challenge, we propose the Item-Language Model (ILM) to enhance LLMs for recommendation. ILM consists of two main components: An item-language representation learning module, where an ILM encoder is pretrained to generate text-aligned item representations. And an item-language co-training module, where the ILM encoder is integrated into a pretrained LLM for the recommendation tasks. Extensive experiments demonstrate the superior performance of our approach over several state-of-the-art methods, validating the importance of text-aligned item representations in bridging this modality gap. Our ablation studies further reveal the effectiveness of our model design for integrating the interaction knowledge into LLMs for recommendation tasks. Our code is available at: `https://anonymous.4open.science/r/ILM-7AD4/`.

1 Introduction

Large Language Models (LLMs) (OpenAI, 2024a; Gemini Team, 2024b; Dubey et al., 2024), have demonstrated remarkable emergent capabilities, including complex reasoning (OpenAI, 2024b; DeepSeek-AI, 2025) and multimodal understanding, when scaled in data, parameters, and compute. These models have achieved and even surpassed human-level performance on various professional and academic benchmarks. In contrast, advancements in recommender systems have not seen similar breakthroughs despite the rapid progress in LLMs (Wu et al., 2023; Chen et al., 2024; Lin et al., 2024b), even though recommendation algorithms power a significant portion of online user activity. A key challenge is that user interactions in recommender systems are not naturally expressed in language. Instead, these signals are often implicit, such as item co-watch behavior. For example, in a video recommendation system, if many users watch both v_1 and v_2, a user who likes v_1 may also like v_2. However, these interactions are difficult to explain, and natural language descriptions of such patterns are often unavailable. As a result, LLMs pretrained on web-scale text data struggle to inherently understand user-item interaction signals, leading to suboptimal performance compared to traditional recommendation algorithms.

Various approaches have been proposed to tackle this issue, including 1) Using LLMs as agents and employing tool-use to integrate them with recommender systems (Zhang et al., 2024; Wang et al., 2024c). 2) Inputting different types of item content representations—such as text, images, audio,

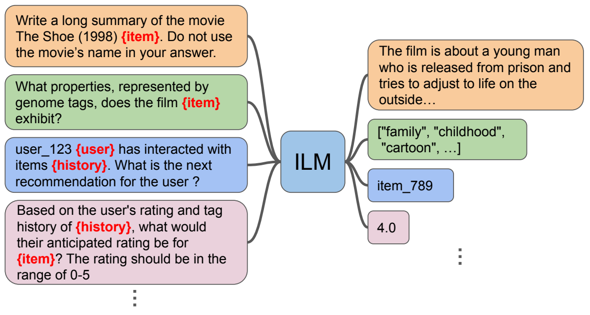
Write a long summary of the movie The Shoe (1998) {item}. Do not use the movie's name in your answer.
What properties, represented by genome tags, does the film {item} exhibit?
user_123 {user} has interacted with items {history}. What is the next recommendation for the user ?
Based on the user's rating and tag history of {history}, what would their anticipated rating be for {item}? The rating should be in the range of 0-5
ILM
The film is about a young man who is released from prison and tries to adjust to life on the outside…
["family", "childhood", "cartoon", …]
item_789
4.0

Figure 1: ILM is applicable in various types of recommendation tasks. User and item collaborative filtering embeddings, marked by placeholders in the input, are interleaved with text embeddings and fed to the model. {history} can be a sequence of item embeddings.

*Corresponding author

and content embeddings—into an LLM (Dai et al., 2023; Liao et al., 2024). 3) Providing item IDs as input to an LLM (Geng et al., 2022; Rajput et al., 2023). 4) Inputting item collaborative filtering (CF) embeddings into an LLM (Zhang et al., 2025; Tennenholtz et al., 2024). The first two approaches do not enable LLMs to directly understand user-item interaction signals. In contrast, the latter two approaches allow LLMs to incorporate both language understanding and user-item interaction signals. However, a modality gap exists between item CF signals and an LLM's internal knowledge. Most prior works address this gap using simple embedding lookups or MLP networks for item representation, but how item representations can effectively bridge this modality gap remains an open question.

To address this limitation, we propose the Item-Language Model (ILM), which effectively incorporates the interaction patterns with item representation learning, and can be directly used in various recommendation tasks, as shown in Figure 1. Specifically, ILM is a two-stage framework consisting of an item-language representation learning stage and an item-language co-training stage. In the first stage, the ILM encoder takes collaborative filtering (CF) embeddings as input and is pretrained using various contrastive learning objectives to generate language-aligned item representations. In the second stage, we integrate the ILM encoder into a pretrained LLM using a linear projection adaptor layer. The LLM receives an interleaved sequence of item and token embeddings as input, enabling it to incorporate both language understanding and user-item interaction signals. The main contributions of this work are summarized as follows:

- We reveal that item-language representation learning plays a crucial role in closing the modality gap between user-item interaction signals and language, and propose an item-language model to effectively bridge this gap.
- We design an interleaved format of item and token embeddings, allowing ILM to be able to perform arbitrary recommendation tasks, and therefore effectively integrates the interaction information into LLM through the co-training.
- We conduct extensive experiments and ablation studies across various recommendation tasks, demonstrating that our ILM approach consistently outperforms existing methods for integrating item representations into LLMs.

Paper Overview. In Section 2, we provide a brief literature survey of LLM for recommender systems, how item representations are used in existing approaches. In Section 3, we present the details of the proposed ILM approach, including model architecture and training. We present our experiment results in Section 4 with deep analysis and discussions in Section 5. We conclude in Section 6.

2 Related Work

LLM for Recommendation With LLMs showing remarkable emergent abilities and surpassing human-level performance across various domains, there have been explorations to apply LLMs to recommender systems (Wu et al., 2023; Chen et al., 2024; Zhao et al., 2024; Lin et al., 2024b; Li et al., 2025). In-Context Learning methods have been used as a straightforward way for this purpose (Dai et al., 2023; Gao et al., 2023; Kang et al., 2023; Zhang et al., 2021; Wang et al., 2023), which rely on LLMs' world knowledge. However, since user interaction data in recommender systems is largely not available during LLM pretraining, purely text-prompting based methods show suboptimal performance. Another line of work is by finetuning a language model on user interaction data. P5 (Geng et al., 2022; Xu et al., 2023) pretrains a unified language model for many different recommendation tasks by converting them into a common natural language sequence format, where user and item ids are represented as text strings. TALLRec (Bao et al., 2023) finetunes a LLM using LoRA (Hu et al., 2022) on user rating data, and the model outputs a binary label.

Item Representations in LLM4Rec Efficiently representing users and items in recommender systems is a rich field with years of work of traditional techniques such as Matrix Factorization (Koren et al., 2009; Rendle et al., 2022). When applying LLM to recommender systems, users and items are key objects, and it is critical for LLM to be able to understand them. Using text representation, such as the title of an item is a straightforward way. However, one issue is text representation of an item may not be informative enough. For example, it is quite common that a video title is unrelated to the video content, and different content can have a single title. We can always include more text features of an item, such as description, author and other content features, but this may introduce irrelevant information, which may confuse the model or

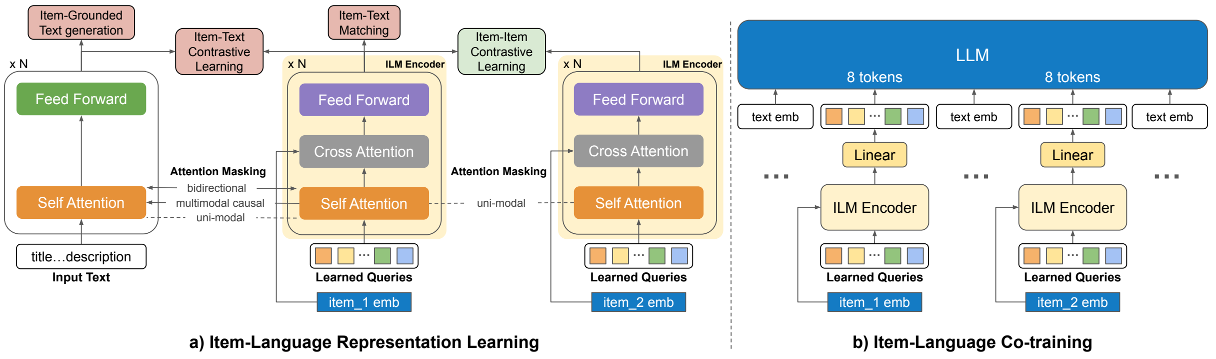

Figure 2: Overall model architecture of ILM. a) Item-Language Representation Learning. The ILM encoder takes CF embeddings as input and is pretrained using various contrastive learning objectives to generate language-aligned item representations. b) Item-Language Co-training. ILM encoder is integrated into a pretrained LLM using a linear projection adaptor layer. The LLM receives an interleaved sequence of item and token embeddings.

make the method inefficient due to very long input. Quantized id representations as item representation has been proposed in the methods of random indexing (Anderson et al., 2020; Xu et al., 2023), sequential or collaborative indexing (Xu et al., 2023; Hua et al., 2023) and semantic ids in the context of generative retrieval (Wang et al., 2024a,b; Liu et al., 2025). In addition to using quantized ids, ELM (Tennenholtz et al., 2024) demystifies the input embedding spaces by feeding semantic embeddings to LLM, and CoLLM (Zhang et al., 2025) enhances recommendation performance on rating prediction tasks by feeding user and item collaborative filtering embeddings to LLM. Recently, USER-LLM (Ning et al., 2024) contextualizes LLM with user history embeddings by integrating user embeddings to LLM through perceiver (Jaegle et al., 2021; Alayrac et al., 2022), projection, and cross-attention modules, where the LLM can be frozen to preserve the original ability.

3 Methodology

3.1 ILM Overview

In LLM-based recommendation models, the input and output typically consist of interleaved sequences of items and text, making the model's ability to understand items crucial. The key idea behind ILM is to develop an effective item-language learning framework that bridges the gap between item representation and LLM adaptation for recommendation tasks. The overall ILM architecture is illustrated in Figure 2. It comprises two main components: (1) Item-Language Representation Learning Module (Section 3.2): This module trains the ILM encoder using multiple contrastive learning objectives to generate language-aligned item representations. (2) Item-Language Co-training Module (Section 3.3): This module integrates the pretrained ILM encoder into an LLM via a linear projection adaptor layer. The LLM is then co-trained on recommendation tasks using an interleaved sequence of item and token embeddings. This structured approach ensures that the LLM effectively incorporates user-item interaction signals, enhancing its recommendation capabilities.

3.2 Item-Language Representation Learning

In the Item-Language Representation Learning stage, we pretrain the ILM encoder to generate language-aligned item representations from item collaborative filtering (CF) embeddings. To achieve this, we employ three item-text learning tasks: *item-text generation*, *item-text matching*, and *item-text contrastive learning*, as illustrated in Figure 2(a). Specifically, for item-grounded text generation, we apply an auto-regressive loss on top of the text tower, following the approach in (Li et al., 2023). For item-text matching, we use a binary cross-entropy loss on the CLS token output from the text tower. For item-text contrastive learning, given a positive item-text pair, we compute the output representations $[h_1, h_2, ..., h_N]$ for the N learnable queries $[q_1, q_2, ..., q_N]$ from the query tower f_q and use the CLS output representation from the text tower f_t as the text representation. The contrastive loss is then computed to align these representations effectively:

$$\begin{aligned} [h_1, h_2, ..., h_N] &= f_q([q_1, q_2, ..., q_N], e) \\ h^{\text{text}} &= f_t([t_{\text{cls}}, t_1, ..., t_L]), \end{aligned} \quad (1)$$

where e is the item input embedding and $[t_1, ..., t_L]$ are text tokens. We select the closest query representation to h^{text} as the item representation:

$$h^{\text{item}} = \max_{h_i} \text{ cosine\_similarity}(h_i, h^{\text{text}}) \tag{2}$$

Then the contrastive loss between h^{item} and h^{text} is computed using in-batch negative sampling following (Radford et al., 2021):

$$\begin{aligned}&\mathcal{L}_{\text{item}-\text{text}} \\ &= -\sum_{j=1}^{B} \log \frac{e^{s(h_j^{\text{item}}, h_j^{\text{text}})}}{e^{s(h_j^{\text{item}}, h_j^{\text{text}})} + \sum_k^{\mathcal{N}} e^{s(h_j^{\text{item}}, h_k^{\text{text}})}},\end{aligned} \tag{3}$$

where j is the index in the batch, B is the batch size, $\mathcal{N}$ is the number of in-batch negative samples, and s is the cosine similarity function with a learnable temperature parameter. The item-text pair data is processed from item descriptions and tags, more details can be found in Section 4.1.

In addition to item-text alignment tasks, we introduce a novel *item-item contrastive learning task* to mitigate overfitting on item-text data, especially when text labels are sparse. This task extends the item-text contrastive loss to directly align item representations. Specifically, given a positive item-item pair, we compute the output representations of the N learnable queries for both items. This ensures that items with similar interaction patterns are mapped closer in the learned representation space, improving model robustness and generalization.

$$\begin{aligned}[h_1^{[1]}, h_2^{[1]}, ..., h_N^{[1]}] &= f_q([q_1, q_2, ..., q_N], e^{[1]}) \\ [h_1^{[2]}, h_2^{[2]}, ..., h_N^{[2]}] &= f_q([q_1, q_2, ..., q_N], e^{[2]}),\end{aligned} \tag{4}$$

where $e^{[1]}$ and $e^{[2]}$ are embeddings for item_1 and item_2, respectively. Note that we use a single set of learnable queries for all items. We then select the pair of closest query representations as the representations for the two items h^{item_1} and h^{item_2}, and compute the contrastive loss between them:

$$\begin{aligned}&\mathcal{L}_{\text{item}-\text{item}} \\ &= -\sum_{j=1}^{B} \log \frac{e^{s(h_j^{\text{item}_1}, h_j^{\text{item}_2})}}{e^{s(h_j^{\text{item}_1}, h_j^{\text{item}_2})} + \sum_k^{\mathcal{N}} e^{s(h_j^{\text{item}_1}, h_k^{\text{item}_2})}},\end{aligned} \tag{5}$$

Combining item-item learning with item-text learning offers a key advantage: it enhances the ILM encoder's ability to generate higher-quality representations. These representations not only capture item-text similarity but also encode item-item similarity, allowing the model to infer text-related information for items without explicit text labels. However, due to the in-batch negative sampling used in contrastive loss, we cannot mix item-text and item-item examples within the same batch. To address this, we train the ILM encoder on separate item-text and item-item batches, optimizing $\mathcal{L}_{\text{item}-\text{text}}$ and $\mathcal{L}_{\text{item}-\text{item}}$ in an alternating manner. The impact of item-item contrastive loss is further analyzed through ablation studies in Section 5.3.

3.3 Item-Language Co-Training

After item-language representation learning, we integrate the learned ILM encoder with a pretrained LLM in a co-training module for recommendation tasks as shown in Figure 2(b). Specifically, For each text token x_i, we retrieve its embedding e_i from the embedding table. For each item y_j, the ILM encoder generates a sequence of N embeddings, corresponding to the query length in the ILM encoder. We then apply a linear MLP projection to align these embeddings with the token embedding space, resulting in $[f_j^1, f_j^2, ..., f_j^N]$. For example, given an interleaved sequence of items and text $[x_1, x_2, y_1, x_3, y_2, x_4]$, the corresponding input sequence of embeddings to the LLM becomes $[e_1, e_2, f_1^1, f_1^2, ..., f_1^N, e_3, f_2^1, f_2^2, ..., f_2^N, e_4]$. The LLM's output consists solely of text and is optimized using autoregressive learning.

$$\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{T} \log p\left(x_t \mid x_1, x_2, y_1, x_3, \ldots, x_{t-1}\right) \tag{6}$$

The LLM can be pretrained for various purposes, including general-purpose instruction-following LLMs (OpenAI, 2024a; Gemini Team, 2024b,a) and recommendation-specialized LLMs trained on recommendation data to support generative retrieval (Rajput et al., 2023; Sun et al., 2023). We evaluate both scenarios and present experimental results in the experiment section.

4 Experiments

4.1 Datasets

We use the widely adopted Embedding Language Model (ELM) (Tennenholtz et al., 2024) and OpenP5 (Xu et al., 2023) benchmarks to evaluate recommendation tasks.

ELM 24 Tasks are derived from the MovieLens 25M dataset, including single movie tasks, e.g. summarizing a movie, and movie pair tasks, e.g. comparing characteristics of movies. The training targets are generated by prompting the PaLM 2-L model with a movie's title and task-specific information. For training inputs, the same task-specific prompts are used, with the movie title replaced

Table 1: Performance comparison results of ILM with two SOTA baselines on ELM 24 Tasks.

Model	Sum.	Pos. Rev.	Neu. Rev.	5Pos Ch.	5Neg Ch.	Long Desc.	Fun.	Sad.	Scare	Imp.	M2V	Pitch	Crit.	Conv1	Conv2	Conv3	Diss1	Diss2	Sim.	Interp.	WhyNN	DiffNN	CommNN	All
ELM (Tennenholtz et al., 2024)	81.53	**88.12**	84.41	86.41	84.89	80.81	75.52	77.86	76.77	83.30	84.72	87.96	83.04	83.02	81.82	80.54	80.97	80.69	84.53	75.94	82.22	84.70	79.71	82.15
LLaRA (Liao et al., 2024)	80.32	86.65	81.87	85.34	85.26	78.15	73.68	76.33	75.16	79.51	81.39	86.54	78.75	81.25	80.10	78.56	78.22	79.34	83.72	74.26	80.21	85.24	80.18	80.46
CoLLM (Zhang et al., 2025)	78.44	85.25	81.47	85.06	86.77	77.83	73.42	76.06	75.24	77.94	80.70	85.26	79.30	80.74	79.62	77.69	79.72	80.22	83.51	73.86	79.33	85.09	80.85	80.27
ILM	**82.66**	87.89	**85.48**	**91.19**	**93.89**	**81.58**	**76.78**	**79.39**	**78.50**	**84.67**	**88.44**	**89.01**	**85.01**	**83.60**	**84.97**	**85.14**	**81.84**	**85.77**	**90.48**	**78.38**	**88.72**	**93.28**	**88.90**	**85.44**

by the movie embedding. We use semantic embeddings provided in the ELM dataset, which are generated by PALM2-XS LLM on item descriptions. We use behavioral embeddings trained on user ratings in the MovieLens 25M dataset with Matrix Factorization computed using Weighted Alternating Least Squares (WALS) (Hu et al., 2008).

OpenP5 is a dataset designed for LLM-based recommendation development, fine-tuning, and evaluation. It includes 10 preprocessed public datasets, each supporting two types of tasks: Sequential Recommendation and Straightforward Recommendation. For our benchmarks, we select the MovieLens-1M, Beauty, and Clothing datasets, using random indexing item representations, which can be viewed as a simple generative retrieval setup. The training target for each example is the ground truth item id. The training input consists of each item's random indexing ID, appended with its behavioral embedding, which is computed using the iALS matrix factorization algorithm (Rendle et al., 2021) on the user sequence training set. We follow the provided train, development, and test splits in OpenP5, where the last item is used for testing with the second-to-last item used for development.

4.2 Evaluation Metrics

For ELM 24 tasks, we report the Semantic Consistency (SC) (Tennenholtz et al., 2024) on the test set. For SC, we use the cosine similarity of semantic embeddings of the original and decode targets from the Sentence-T5 11B model (Ni et al., 2022). For OpenP5 tasks, we report top-k Hit Rate (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K) with K = 5, 10 to evaluate the recommendation performance. To compute those metrics, we use beam search to generate 10 outputs for each example, and remove invalid outputs that do not match the regular expression.

4.3 Results on ELM 24 Tasks

We compare ILM with the three SOTA baselines on ELM 24 tasks, including CoLLM (Zhang et al., 2025), LLaRA (Liao et al., 2024) and ELM (Tennenholtz et al., 2024). CoLLM uses a two layer MLP with intermediate size 10 times the input embedding size to map the input behavioral embedding to LLM token embedding space. During training, both the LLM and the MLP parameters are trained. LLaRA derives the textual and behavioral tokens from the ID-based item embedding learned by traditional recommender models and use them for item representations. There is no item-text representation stage. ELM uses a MLP adapter to adapt the item embeddings to language space. A two-stage training strategy is used. In stage-1 it trains only the adapter and keep the LLM frozen. In stage-2 it fully finetunes all the parameters in the LLM and the MLP adapter.

For ILM encoder, we use a Q-Former (Li et al., 2023) with 8 transformer layers. During item-language representation learning stage, we pair the item with a concatenation of a) the prompt original used by ELM to generate the target, i.e. title(s) and task specific information, and b) the target as the item-text pairs. For the ELM 24 tasks benchmark, we only train the ILM encoder with item-text data. In the item-language co-training stage, we fully finetune both the ILM encoder and the LLM. In all experiments, we use PaLM 2-S (Google, 2023) as the LLM backbone. We train the models for 100k steps using a batch size 32 and learning rate 5×10^{-4} with a cosine decay.

Table 2: Performance comparison results of ILM with two SOTA baselines on OpenP5 tasks.

Setting	Method	ML1M				Beauty				Clothing			
		HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10
Seen	OpenP5-R (Xu et al., 2023)	0.0688	0.0455	0.1033	0.0566	0.0208	0.0162	0.0254	0.0177	0.0015	0.0010	0.0030	0.0015
	LLaRA (Liao et al., 2024)	0.0723	0.0478	**0.1084**	0.0576	0.0206	0.0154	0.0259	0.0171	0.0026	0.0017	0.0038	0.0021
	CoLLM (Zhang et al., 2025)	0.0692	0.0459	0.1041	0.0572	0.0199	0.0150	0.0255	0.0168	0.0015	0.0010	0.0022	0.0012
	ILM	**0.0724**	**0.0485**	0.1064	**0.0595**	**0.0213**	**0.0164**	**0.0270**	**0.0182**	**0.0041**	**0.0025**	**0.0065**	**0.0033**
Unseen	OpenP5-R (Xu et al., 2023)	0.0696	0.0449	0.1041	0.0560	0.0206	0.0161	0.0253	0.0176	0.0016	0.0010	0.0034	0.0016
	LLaRA (Liao et al., 2024)	0.0710	0.0477	0.1033	0.0581	0.0206	0.0157	0.0263	0.0175	0.0023	0.0015	0.0037	0.0019
	CoLLM (Zhang et al., 2025)	0.0716	0.0470	0.1045	0.0576	0.0203	0.0151	0.0255	0.0168	0.0016	0.0011	0.0026	0.0014
	ILM	**0.0717**	**0.0481**	**0.1086**	**0.0600**	**0.0213**	**0.0162**	**0.0269**	**0.0181**	**0.0038**	**0.0024**	**0.0062**	**0.0032**

The comparison results are presented in Table 1. It can be seen that our approach consistently outperforms both strong baselines on almost all tasks. This observation validates the effectiveness of ILM in bridging the item-language gap in LLM learning for recommendation. We further report the mean and standard deviation results in Table 11.

4.4 Results on OpenP5

In OpenP5, each task contains 10 prompt templates used for training with one prompt template used for unseen testing. For the baselines, we compare ILM with an additional method, OpenP5-R (Xu et al., 2023). OpenP5-R stands for the OpenP5 random indexing method, i.e., using the backbone model directly without any embedding inputs, where same LLM backbone as our ILM approach is used.

For item-language representation learning stage, we generate the item-text pair data by extracting item metadata from (1) movie title and genres from the original Movielens-1M dataset (Harper and Konstan, 2015) for the ML1M task (2) product metadata including title, description, features, brand, etc. from the original Amazon Review 2014 Metadata (He and McAuley, 2016) for Beauty and Clothing tasks. Since the inputs of OpenP5 tasks contain both user id and item id, we generate user-item pairs using the training set of OpenP5's user sequence data, and conduct user-item contrastive learning [*]. For the co-training stage, we use an 8 layer transformer model as the LLM backbone, and pretrain the backbone on the OpenP5 data using random item indexing to enable the model generative retrieval (Geng et al., 2022; Xu et al., 2023; Rajput et al., 2023; Sun et al., 2023) ability. ILM approach can be integrated with any other type of item token id based encoding such as sequential indexing and collaborative indexing in the OpenP5 dataset as well as other more advanced semantic id based methods (Rajput et al., 2023; Sun et al., 2023). Training hyperparameters can be found in Section B. We present the statistics of data used in both stages in Table 6.

For each dataset, we select the checkpoint with the best NDCG@10 metric on the development set. The test results on both seen and unseen OpenP5 MovieLens-1M, Beauty, and Clothing tasks are reported in Table 2. From the results we can observe that our method consistently outperforms other baselines across all tasks and settings, demonstrating the effectiveness and generalization of ILM on various recommendation tasks. We report the mean and standard deviation results in Table 12.

5 Analysis and Discussion

5.1 Impact of Item Embedding Types

Items in a recommender system can have both semantic and behavioral embeddings, both of which can be used as inputs to the item encoder in our framework. In this study, we examine three types of embeddings and evaluation the ILM performance: a) *Semantic embedding* from the ELM dataset, which consists of content embeddings generated by the PaLM2-XS LLM on item descriptions. b) *Behavioral embedding*, generated through collaborative filtering trained with Alternating Least Squares (ALS) (Rendle, 2022) on the MovieLens-25M user-item interaction data. c) *Combined embedding*, which is generated by concatenating an item's semantic and behavioral embeddings.

The performance comparison results on a representative subset of the ELM 24 tasks are shown in Table 3 (full results are presented in Table 13 in the Appendix). It is unsurprising that the ILM performance of using semantic embeddings outperforms behavioral embeddings, as many of the ELM 24 tasks assess content understanding of items. However, when combining semantic and behavioral embeddings, ILM achieves the best performance, indicating that behavioral embeddings learned from

[*]Here we simply take user CF embedding as input to the ILM encoder, so user-item and item-item contrastive learnings are using exactly the same setup.

Table 3: ILM performance on ELM 24 tasks with different item embedding types.

	Sum.	Pos. Rev.	Long Desc.	Funnier	Sim.	Interp.	WhyNN	All
ILM-Semantic	82.15	87.70	81.15	76.10	90.16	77.85	87.61	85.08
ILM-Behavioral	74.06	79.09	72.58	69.43	80.87	71.92	80.57	78.43
ILM-Combined	**82.66**	**87.89**	**81.58**	**76.78**	**90.48**	**78.38**	**88.72**	**85.44**

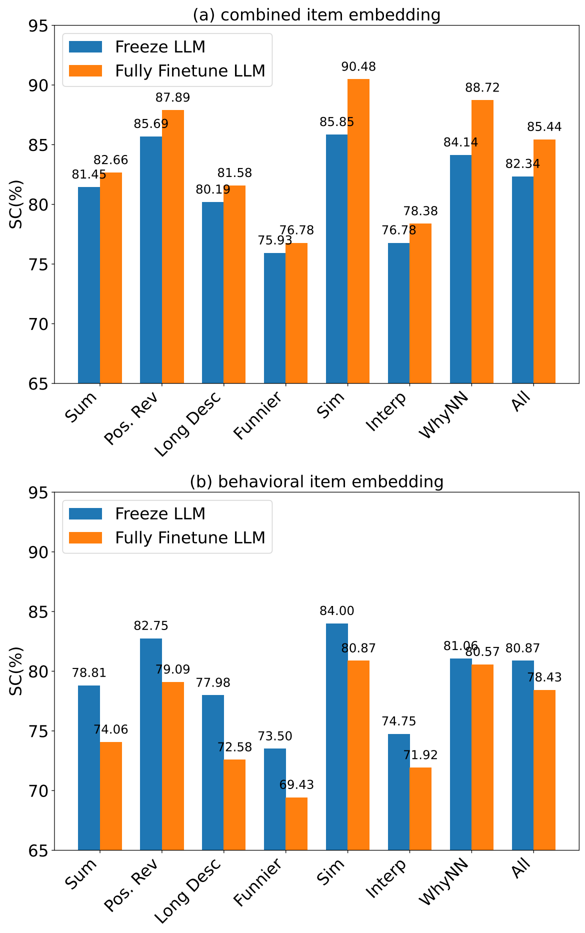

Figure 3: Effects of LLM training strategies in item-language co-training stage on ELM benchmark.

the interaction data contribute complementary information to the content embeddings.

5.2 Freezing v.s. Finetuning LLM

In the item-language co-training stage, we have the option to either freeze the LLM or fine-tune the LLM. To assess the effectiveness of LLM co-training, we conduct an experiment by freezing the LLM during this stage. The comparison results of freezing versus fine-tuning the LLM using combined item embeddings on the ELM 24 tasks are presented in Figure 3(a). It is evident that full fine-tuning the LLM consistently leads to better model performance. Similar patterns have been observed in previous works (Lin et al., 2024a; Tennenholtz et al., 2024). The reason is that fine-tuning the LLM facilitates better alignment between the item and language embedding spaces. Interestingly, we also observe that when using behavioral item embeddings alone, freezing the LLM results in better performance as shown in Figure 3(b). Our hypothesis is that the modality gap between behavioral embeddings and the LLM's knowledge is too large, making it difficult for the LLM to be effectively optimized during fine-tuning.

5.3 Impact of Item-Language Representation Learning

To evaluate the overall effectiveness of item-language representation learning, we conduct an ablation study by removing the ILM encoder training stage—i.e., directly using the original item embeddings in the item-language co-training. To isolate the effect of item encoder representation learning, we freeze the LLM in the co-training stage for both settings (with and without the first stage). The performance comparison results are presented in Figure 4. The results show a significant performance drop across all tasks when the item-language representation learning stage is removed. This highlights the importance of the ILM encoder in learning better item representations and effectively bridging the modality gap between user-item interaction signals and LLM knowledge.

Table 4: Impact of item-item and item-text losses on OpenP5 ML1M benchmark.

Setting	Methods	HR@5	NDCG@5	HR@10	NDCG@10
seen	ILM-IT	0.0719	0.0474	**0.1088**	0.0594
	ILM	**0.0724**	**0.0485**	0.1064	**0.0595**
unseen	ILM-IT	0.0700	0.0470	0.1071	0.0589
	ILM	**0.0717**	**0.0481**	**0.1086**	**0.0600**

5.4 Impact of Different Losses in ILM Encoder Learning

In item-language representation learning, the ILM encoder is trained using two types of training losses: item-text and item-item contrastive losses. To better understand the impact of these training tasks on ILM encoder learning, we conduct experiments

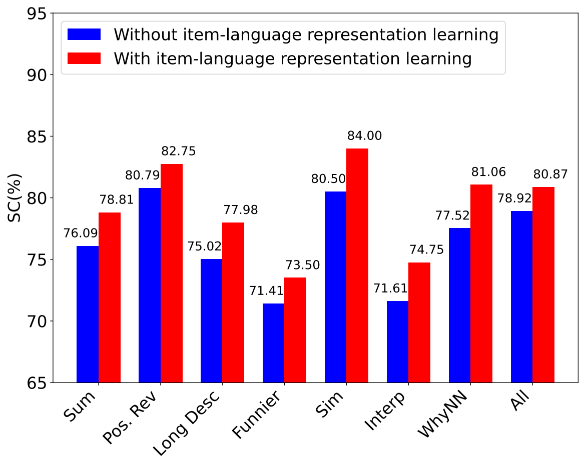

Figure 4: Impact of item-language representation learning stage on representative tasks on ELM benchmark.

Table 5: Effects of stage-1 item-item and user-item contrastive losses on OpenP5 stage-1 final train and eval item-grounded text generation losses.

Methods	ML1M		Beauty		Clothing	
	Train	Eval	Train	Eval	Train	Eval
ILM-IT	0.0000	4.1699	1.0441	4.2643	0.2114	2.0530
ILM	0.0089	4.0663	2.3420	3.3724	0.5498	1.6358

with different loss combinations: (1) Only using item-text losses (refer to as **ILM-IT**). (2) Combine item-text losses with the item-item contrastive loss, i.e. our (**ILM**) approach.

The comparison results on the OpenP5 ML1M benchmark are presented in Table 4. The results indicate that introducing item-item contrastive loss generally leads to performance improvements, highlighting its effectiveness in enhancing item representations. To further demonstrate the benefits of item-item contrastive loss, we examine the final training and evaluation losses for item-grounded text generation, as shown in Table 5. The results reveal that incorporating item-item contrastive loss indeed helps to reduce evaluation loss and narrow the train-eval gap, reinforcing its role in improving generalization during ILM encoder learning.

5.5 Impact of Item Token Numbers

Another key aspect of our ILM approach is the use of multiple learned queries to generate multiple embeddings as item representations as the ILM encoder output, which are then fed into the LLM. In contrast, existing methods (Tennenholtz et al., 2024; Zhang et al., 2025) typically use a single embedding to represent an item within the LLM. We analyze the impact of using different numbers of query tokens in Figure 5. To better understand the advantages of our approach, we also compare against an MLP baseline, where the input embedding is projected into the same number of embeddings as in our ILM method. For both approaches, performance initially improves as the number of query tokens increases, but then declines, with the best performance observed around 4-8 query tokens. Our hypothesis is that with an excessively large number of query tokens, the input sequence length to the LLM becomes too long (since each item in the input sequence is represented by this number of query tokens), making it difficult to effectively fine-tune the model during the second stage. Additionally, across different query lengths, our ILM approach consistently outperforms the MLP baseline in most cases, highlighting the effectiveness of our multi-query token strategy for item representation.

6 Conclusion

In this work, we propose ILM, a novel approach for integrating collaborative filtering knowledge into large language models (LLMs) for conversational recommendation tasks. Our method follows a two-stage training paradigm. We first trains an ILM encoder to generate item-language aligned representations from semantic and behavioral embeddings, which are then interleaved with text token embeddings and fed into an LLM for co-training on recommendation tasks. We conduct extensive benchmark evaluations across various recommendation datasets and tasks. Our results demonstrate that incorporating item-item and user-item interaction data significantly enhances performance, effectively integrating traditional interaction signals into LLMs. The findings show that ILM consistently outperforms existing methods for integrating collaborative filtering embeddings into LLMs, achieving state-of-the-art performance.

Limitations

We evaluate ILM on the ELM 24 tasks and OpenP5 dataset, showcasing its effectiveness as a unified item-language modeling approach for recommendation tasks with interleaved item-text inputs. This setup is widely applicable in real-world scenarios, such as conversational recommendation, where users engage in multi-turn interactions, and LLM-based agent systems, where models must reason over retrieved items while incorporating collabo-

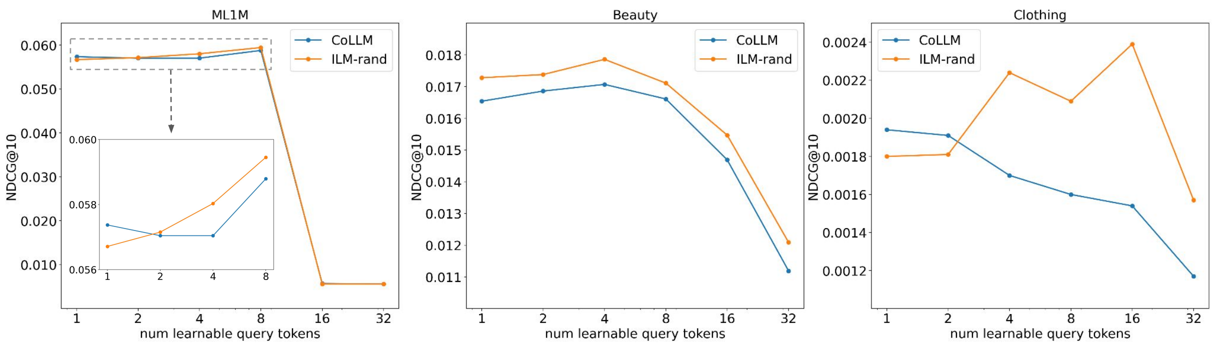

Figure 5: Impact of Number of Query Tokens on the ILM model performance on OpenP5 benchmark.

rative filtering information. One future direction is to benchmark ILM on additional recommendation tasks to assess its generalization, exploring more challenging recommendation scenarios with refined evaluation metrics. Another potential direction is to develop a unified framework that combines the two-stage training, which could help reduce the complexity of the training and improve the consistency of the learned representations.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, et al. 2022. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736. Curran Associates, Inc.

John Anderson, Qingqing Huang, Walid Krichene, Steffen Rendle, and Li Zhang. 2020. Superbloom: Bloom filter meets transformer. *Preprint*, arXiv:2002.04723.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23. ACM.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, Kai Zheng, Defu Lian, and Enhong Chen. 2024. When large language models meet personalization: perspectives of challenges and opportunities. *World Wide Web*, 27(4).

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 1126–1132, New York, NY, USA. Association for Computing Machinery.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiabao Fang, Shen Gao, Pengjie Ren, Xiuying Chen, Suzan Verberne, and Zhaochun Ren. 2024. A multi-agent conversational recommender system. *Preprint*, arXiv:2402.01135.

Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. 2023. A large language model enhanced conversational recommender system. *Preprint*, arXiv:2308.06212.

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

Google Gemini Team. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Google Gemini Team. 2024b. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

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*, RecSys '22, page 299–315, New York, NY, USA. Association for Computing Machinery.

Google. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4).

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 507–517, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. *SIGIR-AP*.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4651–4664. PMLR.

Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. 2023. Do llms understand user preferences? evaluating llms on user rating prediction. *Preprint*, arXiv:2305.06474.

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

Guanghan Li, Xun Zhang, Yufei Zhang, Yifan Yin, Guojun Yin, and Wei Lin. 2025. Semantic convergence: Harmonizing recommender systems via two-stage alignment and behavioral semantic tokenization. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 12040–12048. AAAI Press.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 1785–1795, New York, NY, USA. Association for Computing Machinery.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. 2024a. VILA: On Pre-training for Visual Language Models . In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26679–26689, Los Alamitos, CA, USA. IEEE Computer Society.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. 2024b. How can recommender systems benefit from large language models: A survey. *ACM Trans. Inf. Syst.*

Qidong Liu, Xian Wu, Wanyu Wang, Yejing Wang, Yuanshao Zhu, Xiangyu Zhao, Feng Tian, and Yefeng Zheng. 2025. Llmemb: Large language model can be a good embedding generator for sequential recommendation. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 12183–12191. AAAI Press.

Yuanxing Liu, Weinan Zhang, Yifan Chen, Yuchi Zhang, Haopeng Bai, Fan Feng, Hengbin Cui, Yongbin Li, and Wanxiang Che. 2023. Conversational recommender system and large language model are made for each other in E-commerce pre-sales dialogue. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9587–9605, Singapore. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

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. *Preprint*, arXiv:2402.13598.

OpenAI. 2024a. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024b. Learning to reason with llms. `https://openai.com/index/learning-to-reason-with-llms/`. [Accessed 19-09-2024].

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender systems with generative retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Steffen Rendle. 2022. *Item Recommendation from Implicit Feedback*, pages 143–171. Springer US, New York, NY.

Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. 2021. Revisiting the performance of ials on item recommendation benchmarks. *Preprint*, arXiv:2110.14037.

Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. 2022. Revisiting the performance of ials on item recommendation benchmarks. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 427–435, New York, NY, USA. Association for Computing Machinery.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. In *Advances in Neural Information Processing Systems*, volume 36, pages 46345–46361. Curran Associates, Inc.

Guy Tennenholtz, Yinlam Chow, ChihWei Hsu, Jihwan Jeong, Lior Shani, Azamat Tulepbergenov, Deepak Ramachandran, Martin Mladenov, and Craig Boutilier. 2024. Demystifying embedding spaces using large language models. In *The Twelfth International Conference on Learning Representations*.

Chen Wang, Liangwei Yang, Zhiwei Liu, Xiaolong Liu, Mingdai Yang, Yueqing Liang, and Philip S. Yu. 2024a. Collaborative alignment for recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, pages 2315–2325. ACM.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is ChatGPT a good NLG evaluator? a preliminary study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 1–11, Singapore. Association for Computational Linguistics.

Shirui Wang, Bohan Xie, Ling Ding, Xiaoying Gao, Jianting Chen, and Yang Xiang. 2024b. Secor: Aligning semantic and collaborative representations by large language models for next-point-of-interest recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*, pages 1–11. ACM.

Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu, Xiaojiang Huang, and Yingzhen Yang. 2024c. RecMind: Large language model powered agent for recommendation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364, Mexico City, Mexico. Association for Computational Linguistics.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A survey on large language models for recommendation. *Preprint*, arXiv:2305.19860.

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2023. Openp5: Benchmarking foundation models for recommendation. *arXiv:2306.11134*.

An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024. On generative agents in recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 1807–1817, New York, NY, USA. Association for Computing Machinery.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340.

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

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2024. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6889–6907.

A OpenP5 Training Data Statistics

In Table 6, we show the dataset statistics for OpenP5 stage-1 and stage-2 training. As can be seen, comparing with Amazon Review datasets, Beauty and Clothing, the ML1M dataset contains much fewer number unique users and items, but much larger number of user-item interactions. ML1M items also contain very limited text features, i.e. title and genres, while Amazon Review datasets contain much rich text features, i.e. title, descriptions, product features, etc. We classify ML1M dataset as user interaction rich, and Amazon Review datasets as item content feature rich. Behavioral embeddings play a more important role in our ILM approach for ML1M dataset.

Table 6: OpenP5 stage-1 and stage-2 dataset statistics.

Datasets	stage-1			stage-2			
	Item-text	Item-item	User-item	Train	Test	# Users	# Items
ML1M	3079	479664	888696	19629820	12080	6040	3416
Beauty	10879	103268	138521	2628260	44726	22363	12101
Clothing	20750	142427	180128	3210280	78774	39387	23033

B Hyperparameters

The hyper-parameters for stage-1 and stage-2 trainings on the ELM benchmark are presented in Table 7.

Table 7: Hyper-parameters for ELM 24 tasks.

stage-1	ILM encoder	8 layers, 168M params
	batch size	256
	learning rate	3×10^{-5}
	schedule	cosine decay
	optimizer	AdaFactor
	# steps	259K
	hardware	16 Cloud V5 TPUs
stage-2	LLM	PaLM 2-S
	batch size	32
	learning rate	5×10^{-4}
	schedule	linear decay
	optimizer	AdaFactor
	# steps	100K
	hardware	64 Cloud V5 TPUs

The hyper-parameters for stage-1 and stage-2 trainings on the OpenP5 benchmark are presented in Table 8. For OpenP5 benchmark, we use a 8 layer transformer decoder as the LLM backbone. We add an extra stage of pretraining the LLM backbone using text only OpenP5 data to enable generative retrieval. For the ML1M dataset, we pretrain for 100K steps. For the Beauty dataset, we pretrain for 20K steps. For the Clothing dataset, we pretrain for 10K steps.

Table 8: Hyper-parameters for OpenP5 tasks.

stage-1	ILM encoder	8 layers, 168M params
	batch size	256
	learning rate	3×10^{-5}
	schedule	cosine decay
	optimizer	AdaFactor
	# steps	40K for ML1M, 10K for Beauty, 15K for Clothing
	hardware	16 Cloud V5 TPUs
stage-2	LLM	Transformer decoder 8 layers, 128M params
	batch size	32
	learning rate	5×10^{-4}
	schedule	linear decay
	optimizer	AdaFactor
	# steps	50K for ML1M, 20K for Beauty, 20K for Clothing
	hardware	64 Cloud V5 TPUs

Table 9: Training and Inference costs of both stages on all tasks. All inferences are using 4 Cloud V5 TPUs.

	Stage	Training Time	Hardware	Inference Latency
ELM 24	stage-1	45 hours	16 Cloud V5 TPU	NA
	stage-2	115 hours	64 Cloud V5 TPUs	O(1s)
ML1M	stage-1	0.5 hours	16 Cloud V5 TPU	NA
	stage-2	0.5 hours	64 Cloud V5 TPU	O(100ms)
Beauty	stage-1	0.9 hours	16 Cloud V5 TPU	NA
	stage-2	0.2 hours	64 Cloud V5 TPU	O(100ms)
Clothing	stage-1	1.3 hours	16 Cloud V5 TPU	NA
	stage-2	0.2 hours	64 Cloud V5 TPU	O(100ms)

C Training and Inference Cost

We summarize the training time and inference latency in the Table 9. For ELM 24 tasks, the stage-1 computation is relatively lightweight compared with stage-2. Since in stage-1, a small sized ILM encoder is enough for the purpose of item-text alignment, while in stage-2, we need a LLM (PaLM-2S) to generate long and complex outputs. For OpenP5 tasks, the stage-1 and stage-2 computations are on-par. Since in stage-1, we need a similar sized ILM encoder for item-text alignment, while we found in stage-2, a larger LLM backbone will degrade the performance. And for Beauty and Clothing tasks, item descriptions are much longer than the ML1M task, so we observed more computation cost in stage-1 for Beauty and Clothing tasks.

The inference cost is a sum of ILM encoder cost and backbone LLM cost. For ELM 24 tasks, the ILM encoder cost is negligible compared with backbone LLM cost. For OpenP5 tasks, the ILM encoder cost is comparable with backbone LLM cost. Since most of the computation happens in matrix multiplications, we can estimate the inference FLOPs for backbone LLM by 2N*D, where N is the number of input and output tokens, and D is the number of model parameters, and estimate the inference FLOPs for ILM encoder by $2N_i$*D, where N_i is the number of input tokens.

Table 10: Model Generalization: performance results on LLaMa 3.3 and Qwen 2.5.

LLaMa 3.3 8B	ML1M	Beauty	Clothing
OpenP5-R	0.0631	0.0239	0.0036
CoLLM	0.0645	0.0266	0.0053
ILM	0.0673	0.0296	0.0087
Qwen 2.5 7B	**ML1M**	**Beauty**	**Clothing**
OpenP5-R	0.0647	0.0244	0.0042
CoLLM	0.0662	0.0271	0.0061
ILM	0.0695	0.0298	0.0094

D Model Generalization on More LLM Backbones

Our ILM encoder is a general method to align item CF embeddings with text, and our method can be integrated to a LLM trained for any purposes, including general dialogue (Llama 3.3, Qwen 2.5, GPT-4o, etc.), complex reasoning (OpenAI-O3, DeepSeek-R1, etc.), or for recommendation (OpenP5, Tigar, etc.), and inherit the LLM's original ability. To demonstrate the generalizability of our approach, we conduct additional experiments on LLaMa 3.3 and Qwen 2.5 and present the results in Table 10. It can be seen that the results are consistent with those on PaLM.

E Standard Derivation of ILM Result

We present the mean and standard error of the mean (SEM) of ILM results on ELM benchmark in Table 11 and on OpenP5 benchmark in Table 12. All results are computed using 3 runs with different random number seeds. As can be seen, the standard error of the mean is about 1-2 orders of magnitude smaller than gains by our results.

F Full Results on ELM 24 Tasks

We present the full results of ILM with fully fine-tuned LLM in co-training stage and different item embedding types on ELM benchmark in Table 13.

Table 11: The mean and standard error of the ILM metrics on ELM 24 tasks.

Tasks	SC(%)	Log pplx
summary	0.7318 ± 0.00359	0.6250 ± 0.00134
positive review	0.7832 ± 0.00246	0.5711 ± 0.00211
neutral review	0.7680 ± 0.00584	0.5759 ± 0.00345
five pos char.	0.8631 ± 0.00320	0.5960 ± 0.00556
five neg char.	0.9225 ± 0.00017	0.4029 ± 0.00845
long description	0.7258 ± 0.00411	0.6622 ± 0.00130
funnier	0.6891 ± 0.00358	0.5394 ± 0.00085
sadder	0.7181 ± 0.00267	0.5152 ± 0.00112
scarier	0.7241 ± 0.00261	0.5251 ± 0.00104
improve	0.7759 ± 0.00293	0.5123 ± 0.00264
movie to viewer	0.8008 ± 0.00313	0.6005 ± 0.00425
pitch	0.8370 ± 0.00341	0.5181 ± 0.00198
criticize	0.7894 ± 0.00356	0.5504 ± 0.00406
convince1	0.7881 ± 0.00353	0.6176 ± 0.00353
convince2	0.7957 ± 0.00523	0.7795 ± 0.00533
convince3	0.7949 ± 0.00421	0.8900 ± 0.00714
dissuade1	0.7909 ± 0.00203	0.6076 ± 0.00381
dissuade2	0.8386 ± 0.00167	0.7278 ± 0.00609
similarities	0.8422 ± 0.00585	0.3993 ± 0.00842
interpolation	0.7300 ± 0.00363	0.5310 ± 0.00238
why like nn	0.8030 ± 0.00367	0.6452 ± 0.00556
diff than nn	0.8891 ± 0.00198	0.5111 ± 0.00941
common with nn	0.8433 ± 0.00182	0.5393 ± 0.00817
all	0.7939 ± 0.00317	0.5813 ± 0.00415

Table 12: The mean and standard error of ILM metrics on OpenP5 datasets.

Setting	Dataset	HR@5	NDCG@5
seen	ml1m	0.0715 ± 0.00051	0.0476 ± 0.00047
	beauty	0.0210 ± 0.00018	0.0160 ± 0.00020
	clothing	0.0040 ± 0.000085	0.0025 ± 0.000014
unseen	ml1m	0.0720 ± 0.00044	0.0478 ± 0.00052
	beauty	0.0214 ± 0.00019	0.0162 ± 0.000090
	clothing	0.0038 ± 0.000037	0.0025 ± 0.000041
Set	**Dataset**	**HR@10**	**NDCG@10**
seen	ml1m	0.1072 ± 0.00043	0.0591 ± 0.00020
	beauty	0.0265 ± 0.00036	0.0177 ± 0.00025
	clothing	0.0061 ± 0.00022	0.0032 ± 0.000051
unseen	ml1m	0.1069 ± 0.00095	0.0591 ± 0.00049
	beauty	0.0266 ± 0.00020	0.0179 ± 0.000099
	clothing	0.0059 ± 0.00015	0.0032 ± 0.000029

We present full results of ILM with different stage-1 and stage-2 training strategies on ELM benchmark in Table 15.

We further present the task-level analysis on ELM 24 tasks in Table 14. We summarize the gains of our method v.s. the two baselines in the table. The task descriptions are from the original ELM paper. As can be seen, for tasks more related to user's references, e.g. 5Neg Ch., Conv3, etc., or tasks related to item relationships, e.g. Sim.,

Table 13: Full results on effects of embedding types. LLM is fully finetuned in co-training stage.

Tasks	ELM	ILM-Semantic	ILM-Behavioral	ILM-Combined
Sum.	81.53	82.15	74.06	**82.66**
Pos. Rev.	**88.12**	87.70	79.09	87.89
Neu. Rev.	84.41	85.10	79.44	**85.48**
5Pos Ch.	86.41	90.99	82.73	**91.19**
5Neg Ch.	84.89	93.64	84.70	**93.89**
Long Desc.	80.81	81.15	72.58	**81.58**
Fun.	75.52	76.10	69.43	**76.78**
Sad.	77.86	78.66	72.04	**79.39**
Scare	76.77	77.96	71.99	**78.50**
Imp.	83.30	84.34	79.50	**84.67**
M2V	84.72	88.01	79.38	**88.44**
Pitch	87.96	88.92	83.60	**89.01**
Crit.	83.04	84.78	80.21	**85.01**
Conv1	83.02	**83.66**	79.20	83.60
Conv2	81.82	**85.07**	78.00	84.97
Conv3	80.54	84.97	77.07	**85.14**
Diss1	80.97	81.77	78.57	**81.84**
Diss2	80.69	85.64	79.12	**85.77**
Sim.	84.53	90.16	80.87	**90.48**
Interp.	75.94	77.85	71.92	**78.38**
WhyNN	82.22	87.61	80.57	**88.72**
DiffNN	84.70	92.57	86.51	**93.28**
CommNN	79.71	88.32	80.01	**88.90**
All	82.15	85.08	78.43	**85.44**

CommNN., etc., our ILM approach can achieve larger gains. For tasks that are more about item content understanding, e.g. Sum., Long Desc., ILM achieves marginal gains compared with ELM, but still quite significant gain compared with CoLLM, which only uses item behavioral embedding.

G Preserving Pretrained Ability

In conversational recommendation, often the user and the system will conduct multiple turns of conversations, and for the system to achieve a certain goal, tool use may be employed (Feng et al., 2023; Gao et al., 2023; Liu et al., 2023; Fang et al., 2024). There is often no constraint on the topics of the conversations, so the pretrained abilities of the LLM could be important. For example, the user may ask the LLM to perform certain operations such as adding or removing certain items from the recommended items, or conducting certain filtering based on a criteria. Those operations require certain reasoning ability from LLM pretraining. For another example, to use LLMs as an automatic agents (Zhang et al., 2024; Wang et al., 2024c) for conversational recommendation, they may require certain broad knowledge to perform tool use (Schick et al., 2023) to achieve a task. If the LLM is later fully finetuned only using the task specific data, it is likely that those pretrained abilities will be lost or hidden. In our ILM approach, a frozen LLM can be used, when the inputs don't contain items, the behavior of the model will be exactly the same as the original LLM. This means all pretrained knowledge can be preserved, which is crucial for multi-turn conversations and tool use in automatic agents.

H More OpenP5 Results

For OpenP5, we experiment with different combinations of stage-1 training losses: (1) Only using item-text losses (**ILM-IT**). (2) Combine (1) with an item-item contrastive loss (**ILM-IT-II**). (3) Combine (1) with an user-item contrastive loss (**ILM**). We generate item-item pair data by using two consecutive items in the history sequence as a positive pair, then we perform de-duplication. The results are in Table 16. For ML1M, introducing II or UI contrastive losses can lead to performance gains, while for Beauty and Clothing there are no obvious gains. We hypothesize this is due to ML1M having richer user interactions and scarcer item text features than the other two datasets, Table 6. This supports our hypothesis, and suggests exploring user-interaction signals in the stage-1 representation learning can be beneficial for tasks like ML1M.

Table 14: Task-level analysis on ELM 24 tasks.

Task	Description	SC gain v.s. CoLLM	SC gain v.s. ELM
Sum.	One paragraph summarizing of movie plot.	4.22	1.13
Pos. Rev.	A positive review of the movie.	2.64	-0.23
Neu. Rev.	A negative review of the movie.	4.01	1.07
5Pos Ch.	Listing five positive characteristics of the movie.	6.13	4.78
5Neg Ch.	Listing five negative characteristics of the movie.	7.12	9.00
Long Desc.	A long exhaustive description of the movie plot.	3.75	0.77
Fun.	A plot for a funnier version of the movie.	3.36	1.26
Sad.	A plot for a sadder version of the movie.	3.33	1.53
Scare	A plot for a scarier version of the movie.	3.26	1.73
Imp.	An improved version of the movie (as generated by an LLM)	6.73	1.37
M2V	Describing a viewer that would like to watch this movie, including characteristics.	7.74	3.72
Pitch	A pitch for the movie.	3.75	1.05
Crit.	Criticizing the movie.	5.71	1.97
Conv1	Convincing to watch the movie.	2.86	0.58
Conv2	Convincing in detail to watch the movie.	5.35	3.15
Conv3	Convincing briefly to watch the movie.	7.45	4.60
Diss1	Dissuading to watch the movie (version 1 prompt).	2.12	0.87
Diss2	Dissuading in detail to watch the movie.	5.55	5.08
Sim.	List three similarities between the movies.	6.97	5.95
Interp.	Interpolate the plots of two movies.	4.52	2.44
WhyNN	Explain why someone would like a nearest neighbor movie.	9.39	6.5
DiffNN	Three major differences between two nearest neighbor movies.	8.19	8.58
CommNN	Three similarities between two nearest neighbor movies.	8.05	9.19

Table 15: Full results on ELM 24 tasks using semantic and behavioral embeddings.

Tasks	Item Semantic Embeddings					Item Behavioral Embeddings			
	ELM	CoLLM	ILM-RF	ILM-F	ILM	CoLLM	ILM-RF	ILM-F	ILM
Sum.	81.53	77.42	81.35	80.98	82.15	71.47	76.09	78.81	74.06
Pos. Rev.	88.12	84.67	86.12	86.14	87.70	76.39	80.79	82.75	79.09
Neu. Rev.	84.41	80.16	84.12	83.80	85.10	73.85	79.99	82.54	79.44
5Pos Ch.	86.41	85.02	85.58	86.17	90.99	80.20	83.26	84.98	82.73
5Neg Ch.	84.89	86.14	84.43	84.66	93.64	83.43	84.46	83.70	84.70
Long Desc.	80.81	76.76	80.37	80.21	81.15	70.71	75.02	77.98	72.58
Fun.	75.52	72.41	75.89	75.37	76.10	68.73	71.41	73.50	69.43
Sad.	77.86	74.90	78.17	77.82	78.66	70.32	73.73	75.90	72.04
Scare	76.77	74.61	77.15	77.01	77.96	70.26	73.31	75.21	71.99
Imp.	83.30	79.46	83.08	82.97	84.34	75.60	79.43	81.44	79.50
M2V	84.72	80.05	84.19	84.40	88.01	75.71	79.97	82.20	79.38
Pitch	87.96	85.35	88.24	88.17	88.92	80.52	84.51	86.29	83.60
Crit.	83.04	79.41	83.10	82.86	84.78	76.21	80.38	81.89	80.21
Conv1	83.02	79.86	83.31	83.23	83.66	75.60	80.87	82.69	79.20
Conv2	81.82	79.71	82.41	82.19	85.07	75.31	79.94	81.77	78.00
Conv3	80.54	77.57	81.20	80.60	84.97	73.88	78.47	80.35	77.07
Diss1	80.97	79.36	81.33	81.08	81.77	76.15	79.50	80.23	78.57
Diss2	80.69	80.17	81.25	81.03	85.64	77.36	80.58	80.92	79.12
Sim.	84.53	82.67	85.86	85.66	90.16	79.05	80.50	84.00	80.87
Interp.	75.94	73.68	76.79	76.74	77.85	71.14	71.61	74.75	71.92
WhyNN	82.22	76.95	84.15	83.97	87.61	75.76	77.52	81.06	80.57
DiffNN	84.70	82.68	84.38	85.47	92.57	80.59	81.89	84.10	86.51
CommNN	79.71	79.22	82.02	82.23	88.32	76.51	78.76	80.57	80.01
All	82.15	79.60	82.44	82.37	85.08	75.59	78.92	80.87	78.43

Table 16: More results on effects of stage-1 item-item and user-item contrastive losses on OpenP5 benchmarks.

Set	Methods	ML1M				Beauty				Clothing			
		HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10
Seen	ILM-IT	0.0719	0.0474	0.1088	0.0594	0.0212	0.0160	0.0262	0.0177	**0.0044**	**0.0029**	0.0061	**0.0035**
	ILM-IT-II	0.0712	0.0479	**0.1093**	**0.0602**	0.0210	0.0160	0.0261	0.0177	0.0040	0.0027	0.0060	0.0033
	ILM-IT	**0.0724**	**0.0485**	0.1064	0.0595	**0.0213**	**0.0164**	**0.0270**	**0.0182**	0.0041	0.0025	**0.0065**	0.0033
Unseen	ILM-IT	0.0700	0.0470	0.1071	0.0589	**0.0218**	**0.0163**	**0.0275**	**0.0182**	**0.0039**	**0.0025**	0.0056	0.0031
	ILM-IT-II	0.0701	0.0472	0.1078	0.0594	0.0216	0.0162	0.0269	0.0180	0.0037	0.0024	0.0054	0.0030
	ILM	**0.0717**	**0.0481**	**0.1086**	**0.0600**	0.0213	0.0162	0.0269	0.0181	0.0038	0.0024	**0.0062**	**0.0032**