Université Paris cité

UFR des Sciences Fondamentales et Biomédicales

PPD Report

A-LLMRec: Large Language Models meet Collaborative Filtering

MSc : Machine Learning for Data Science (MLDS)

Prepared by :
Mr. MAHMAHI Anis
Mr. HATTABI Noureddine Ilyes

Supervised by :
Ms. AFFELDT Séverine

Promotion : 2023/2024

Contents

Chapter 1

Literature Review

1 Introduction

Recommender systems play a crucial role in modern digital platforms by helping users discover relevant content based on their preferences and interactions. These systems are widely used in domains such as e-commerce, entertainment, and social media, enhancing user engagement and satisfaction. Over time, recommender systems have evolved from simple rule-based approaches to sophisticated machine learning-driven models. Among the most prominent methods, three main types have gained significant attention: *collaborative filtering*, *modality-aware recommender systems*, and *large language model (LLM)-based recommender systems*.

1.1 Collaborative Filtering

Collaborative filtering is one of the foundational approaches to recommendation and operates on the principle that users with similar past behaviors are likely to have similar future preferences. It is typically categorized into:

- **User-based collaborative filtering** – This method finds users with similar interaction patterns and recommends items that these similar users have liked.
- **Item-based collaborative filtering** – Instead of focusing on users, this approach identifies relationships between items. If two items frequently appear together in user interactions, one can be recommended when a user engages with the other.

This method has been widely implemented in platforms like Netflix, Amazon, and Spotify to suggest movies, products, and music. However, collaborative filtering faces challenges such as the *cold-start problem*, where new users and items lack sufficient interaction data, and *scalability issues*, as maintaining large similarity matrices becomes computationally expensive.

1.2 Modality-Aware Recommender Systems

Modality-aware recommender systems extend traditional recommendation approaches by incorporating multiple types of data (text, images, audio, and video). Instead of relying solely on user-item interaction data, these systems integrate multimodal information to enhance recommendation accuracy and personalization.

For example, in fashion retail, a recommendation system may use *product images, textual descriptions, and user purchase history* to generate more relevant suggestions. Similarly, video streaming services like YouTube and TikTok leverage *audio-visual content, metadata, and user interactions* to personalize recommendations.

By integrating multiple modalities, these systems can *mitigate the cold-start problem*, as recommendations are not solely dependent on past user interactions. Additionally, they improve content discovery by understanding richer representations of user preferences.

1.3 LLM-Based Recommender Systems

The emergence of *large language models (LLMs)* has introduced a new approach to recommender systems. These models, built on architectures like **GPT, BERT, and T5**, excel at processing and generating human-like text, making them highly effective for personalized recommendations.

Key advantages of LLM-based recommender systems include:

- **Contextual understanding** – LLMs can analyze complex user queries, reviews, and contextual information to provide more nuanced recommendations.
- **Zero-shot and few-shot learning** – Unlike traditional models that require extensive training data, LLMs can generalize well to new items and user preferences with minimal interaction data.
- **Conversational personalization** – Users can express their preferences in natural language (e.g., ”Recommend me a sci-fi novel similar to *Dune*”), allowing for dynamic and interactive recommendation experiences.

These systems are increasingly being explored in *chatbot-driven recommendations, conversational search engines, and AI-powered content discovery.* Platforms like OpenAI’s ChatGPT and Google’s Bard exemplify how LLMs can enable a more intelligent and personalized recommendation experience.

2 Collaborative Filtering Approach

2.1 Overview

Collaborative filtering (CF) is one of the most widely used and well-established techniques in recommender systems. At its core, CF operates on the principle of leveraging user-item interaction data, such as ratings, clicks, or purchases, to identify patterns and similarities among users or items. The underlying assumption is that users who have agreed in the past (e.g., liked similar items) are likely to agree again in the future. CF methods can be broadly categorized into two types: memory-based and model-based approaches.

- **Memory-based CF** relies on similarity metrics (e.g., cosine similarity or Pearson correlation) to find users or items that are similar to the target user or item. These similarities are then used to generate recommendations, either by predicting ratings for unrated items or by directly recommending items that similar users have liked.

- **Model-based CF** employs machine learning algorithms to learn latent representations of users and items from the interaction data. These models aim to capture the underlying structure of the data, enabling more accurate and scalable recommendations.

Despite its success, collaborative filtering faces challenges such as data sparsity, cold-start problems, and scalability issues. Recent advancements in deep learning and graph-based methods have addressed some of these limitations, pushing the boundaries of what CF can achieve. Below, we discuss two state-of-the-art methods from existing papers that have significantly advanced the field.

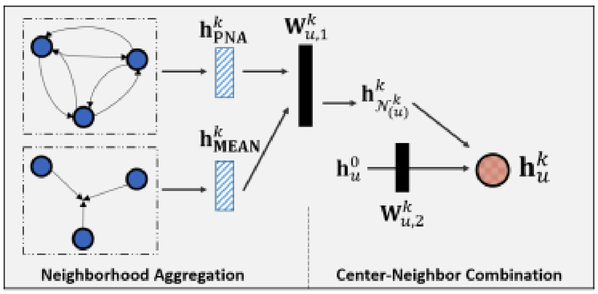
2.2 Previous Works

Neighbor Interaction Aware Graph Convolution Networks for Recommendation (NIA-GCN)

NIA-GCN represents a specialized architecture of graph neural networks (GNNs), Its main components as mentioned in the paper [9] by Sun et al. is firstly the Neighbor Interaction Awareness which means that the model doesn't just consider the direct connections (e.g., a user has rated an item) but also how these connections influence and interact with each other (i.e, the item-item and user-user relational structures), all this to leverage the limitation of traditional GCN approaches. Secondly, the Attention Mechanisms integrated allows the model to weigh the importance of different neighbor interactions differently. This way, the model can learn to focus more on the most informative parts of the graph structure. Also, according to the authors, NIA-GCN architecture is designed to be scalable and efficient, handling large-scale user-item graphs that are common in real-world recommendation systems, as online experiments were conducted to demonstrate that NIA-GCN outperforms the baseline by 10.19% at the time (2020) compared with recommendation models such as PinSAGE, NGCF, GIN ...etc, on Gowalla and 3 different amazon datasets.

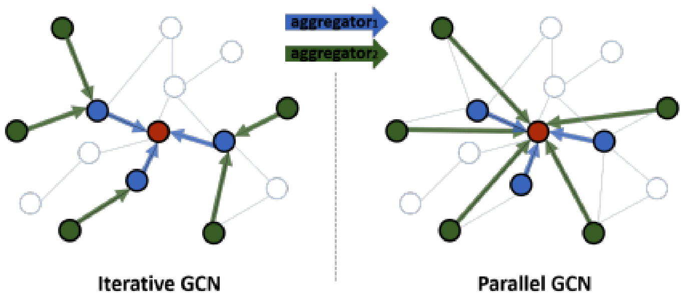
The NIA-GCN is based of three main mechanisms:

- **Pairwise Neighborhood Aggregation graph convolution layer (PNA layer)** This process splits into two steps: neighborhood aggregation, in which an aggregation function operates over a multiset of vectors to aggregate the embeddings of neighbors, and center-neighbor combination that combines the aggregated neighborhood vector with the central node embedding. We now elaborate upon these two steps but first here is a figure of what a PNA layer looks like:


$\mathbf{h}^k_{\mathrm{PNA}}$
$\mathbf{W}^k_{u,1}$
$\mathbf{h}^k_{\mathcal{N}^k(u)}$
$\mathbf{h}^k_{\mathrm{MEAN}}$
$\mathbf{h}^0_u$
$\mathbf{h}^k_u$
$\mathbf{W}^k_{u,2}$
Neighborhood Aggregation
Center-Neighbor Combination

The first step which is the neighborhood aggregation consists of applying element-wise multiplication on every neighbor-neighbor pair (the upper blue rectangle) as well as a sum aggregator (lower blue rectangle) then these two representations are concatenated and passed to a standard MLP. The second step (Center-neighbor aggregation) generates the new node embedding by passing the old one to a standard MLP and concatenate it with the aggregated neighborhood vector and then passed through a tanh activation function (red circle) to generate the new embedding of node u in the layer k

- **Parallel-GCNs** The idea behind this is that instead of recursively updating the node embedding at layer-(k−1) with the neighbors in layer-k, we learn multiple central node embeddings from neighbors at depths 1, ..., k directly, as shown in the figure



  Additionally, instead of employing the same aggregation and transformation function at each layer for every node and sharing aggregators for both user nodes and item nodes, we process them separately by learning two sets of aggregators for two different entities on the bipartite graph.

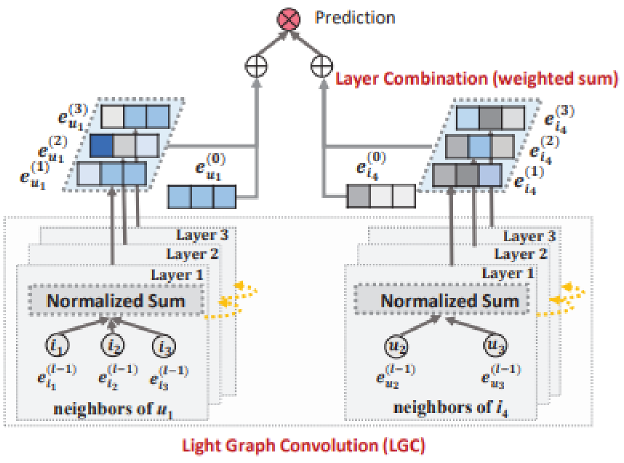
- **Cross-Depth Ensemble (CDE)**

  The final step of the training is derived from k PNA layer outputs and the outputs of the sum aggregator applied to each of the k GCN layers. They combine these 2k information vectors using pairwise multiplication to generate the final representation of user/item nodes

LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

LightGCN is a simplified and efficient version of the Graph Convolutional Network (GCN). The main idea of LightGCN according to [2] by He et al. is to simplify the design of traditional GCN by removing feature transformation and non-linear activation, which are common in standard GCN layers. This simplification leads to a lightweight model that is both easier to train and more efficient in inference. instead they aggregate the embeddings from all layers of the GCN. This means that the final representation of a user or an item is a combination of its embeddings from all the layers, capturing the collaborative signal at different levels of the graph structure. and what we mean by collaborative signal is the concept of Collaborative filtering which is a method used by recommendation systems to make predictions about the interests of a user by collecting preferences from many other users. Which LightGCN excels in according to the paper. Finally, the tests performed on the lightGCN concludes that not only did the model achieved superior performance compared to more complex models like NIA-GCN, especially in tasks like top-N recommendation. but it also has conciderably lower training time due to its simplicity which makes it highly scalable when trained with the pairwise BPR (Bayesian Personalized Ranking) loss or a pointwise mean squared error loss, depending on the specific application and the available data.

To understand in details the idea of lightGCN we can see this diagram



As we can see, LightGCN architecture works the same as GCN, we start by the normalized sum of feature vectors of all neighbors and do this for K layers (for both user and item node), then we combine these vectors using weighted sum, and contrary to the simple GCN we can see that many operation are missing in the lightGCN such as self-connection, feature transformation, and nonlinear activation which were explained in the previous report are missing from the lightGCN architecture, which makes the lightGCN simpler and weirdly enough more performant than previous work, as tested by the authors on

3 Modality-Aware Recommender Systems

3.1 Overview

Modality-aware recommender systems extend traditional recommendation techniques by incorporating multiple data modalities, such as text, images, audio, and video, to enhance the quality and relevance of recommendations. Unlike collaborative filtering, which primarily relies on user-item interaction data, modality-aware systems leverage rich content features to better understand the characteristics of items and the preferences of users. This approach is particularly useful in domains where item content plays a significant role, such as movies, music, fashion, and e-commerce.

The key idea behind modality-aware recommender systems is to learn representations of items and users by fusing information from multiple modalities. For example, in a movie recommendation system, textual data (e.g., plot summaries), visual data (e.g., posters or trailers), and audio data (e.g., soundtracks) can be combined to create a comprehensive representation of each movie. Similarly, user preferences can be inferred from their interactions with different modalities, such as liking certain types of images or engaging with specific textual content.

Recent advancements in deep learning have enabled the development of sophisticated models that can effectively process and integrate multimodal data. These models often employ techniques such as convolutional neural networks (CNNs) for images, recurrent neural networks (RNNs) or transformers for text, and graph neural networks (GNNs) for structured data. By combining these modalities, modality-aware recommender systems can provide more accurate, diverse, and contextually relevant recommendations.

3.2 Method 1: VBPR (Visual Bayesian Personalized Ranking)

Paper: *He & McAuley, 2016, "VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback"*
VBPR is a pioneering modality-aware recommendation model that integrates visual features into the recommendation process. It extends the traditional Bayesian Personalized Ranking (BPR) framework by incorporating visual content, making it particularly suitable for domains like fashion and e-commerce, where item appearance is a critical factor.

- **Visual Features**: VBPR uses pre-trained CNNs (e.g., AlexNet) to extract visual features from item images. These features are then used as additional input to the recommendation model.
- **Model Architecture**: The model learns latent representations of users and items, combining traditional collaborative filtering signals with visual features. The final score for a user-item pair is computed as a weighted sum of the collaborative and visual components.
- **Training**: VBPR is trained using the BPR optimization criterion, which maximizes the likelihood of observed user-item interactions over unobserved ones.
- **Results**: VBPR demonstrated significant improvements over traditional collaborative filtering methods on datasets like Amazon Beauty and Amazon Clothing, highlighting the importance of visual content in recommendations.

3.3 Method 2: MMGCN (Multi-Modal Graph Convolutional Network)

Paper: *Wei et al., 2019, "MMGCN: Multi-Modal Graph Convolution Network for Personalized Recommendation of Micro-Videos"*
MMGCN is a state-of-the-art modality-aware recommender system designed for micro-video recommendations. It leverages graph convolutional networks (GCNs) to model user-item interactions and integrate multiple modalities, such as visual, acoustic, and textual data.

- **Graph Construction**: MMGCN constructs separate graphs for each modality, where nodes represent users and items, and edges represent interactions. The model learns modality-specific embeddings by propagating information through these graphs.
- **Modality Fusion**: After obtaining modality-specific embeddings, MMGCN employs an attention mechanism to dynamically weigh the importance of each modality for different users and items. This allows the model to adaptively fuse multimodal information based on context.

- **Training**: The model is trained end-to-end using a combination of collaborative filtering loss and modality-specific losses, ensuring that both user-item interactions and content features are effectively utilized.

- **Results**: MMGCN achieved state-of-the-art performance on the TikTok micro-video dataset, outperforming methods that rely on single modalities or simple concatenation of multimodal features.

4 LLM-Based Recommender Systems

4.1 Overview

Large Language Models (LLMs), such as GPT, BERT, and their variants, have revolutionized the field of natural language processing (NLP) and are increasingly being applied to recommender systems. LLM-based recommender systems leverage the vast knowledge and language understanding capabilities of these models to provide highly personalized and context-aware recommendations. Unlike traditional collaborative filtering or modality-aware approaches, LLM-based systems excel at processing unstructured data, such as user reviews, product descriptions, and social media posts, enabling them to capture nuanced user preferences and item characteristics.

The key strength of LLM-based recommender systems lies in their ability to understand and generate natural language, which allows for more interactive and explainable recommendations. For example, these systems can engage in conversational recommendations, where users interact with the system in natural language to express their preferences or receive tailored suggestions. Additionally, LLMs can infer user preferences from textual data, such as past reviews or search queries, even in the absence of explicit interaction data.

Despite their potential, LLM-based recommender systems face challenges such as computational complexity, the need for large-scale training data, and the risk of generating biased or irrelevant recommendations. However, recent advancements in fine-tuning, prompt engineering, and efficient inference techniques have made these systems more practical and effective for real-world applications.

4.2 Method 1: P5 (Prompting for Personalized Recommendations)

Paper: *Geng et al., 2022, "P5: A Unified Framework for Personalized Recommendations Using Prompted Language Models"*
P5 is a groundbreaking framework that reformulates recommendation tasks as natural language processing tasks using prompts. By leveraging the generative capabilities of LLMs, P5 can handle a wide range of recommendation scenarios, including rating prediction, item ranking, and explanation generation.

- **Task Formulation**: P5 converts recommendation tasks into text-based prompts. For example, a rating prediction task might be framed as: *"Given user U and item I, predict the rating on a scale of 1 to 5."* This allows the LLM to generate recommendations in a natural language format.

- **Model Architecture**: P5 fine-tunes a pre-trained LLM (e.g., T5 or GPT) on a dataset of user-item interactions and associated textual data. The model learns to generate recommendations by predicting the next token in the sequence based on the input prompt.
- **Multi-Task Learning**: P5 supports multi-task learning, enabling the model to perform various recommendation tasks simultaneously, such as item recommendation, review generation, and user preference inference.
- **Results**: P5 achieved state-of-the-art performance on benchmark datasets like Amazon Books and MovieLens, demonstrating the effectiveness of prompt-based recommendation frameworks.

4.3 Method 2: RecLLM (Recommendation with Large Language Models)

Paper: *Li et al., 2023, "RecLLM: A Unified Framework for Recommendation with Large Language Models"*
RecLLM is a unified framework that integrates LLMs with traditional recommendation techniques to provide highly personalized and explainable recommendations. The framework combines the strengths of LLMs in understanding unstructured data with the structured reasoning capabilities of traditional recommender systems.

- **Hybrid Architecture**: RecLLM uses an LLM to process unstructured data, such as user reviews and item descriptions, and extracts latent representations of users and items. These representations are then combined with collaborative filtering signals to generate recommendations.
- **Explainability**: One of the key features of RecLLM is its ability to generate natural language explanations for recommendations. For example, the model can explain why a particular item is recommended based on the user's past behavior and preferences.
- **Training**: RecLLM is trained using a combination of recommendation loss (e.g., BPR loss) and language modeling loss, ensuring that the model learns both user-item interactions and textual semantics.
- **Results**: RecLLM outperformed traditional recommendation models on datasets like Yelp and Netflix, particularly in scenarios requiring explainability and context-aware recommendations.

Conclusion

Recommender systems have seen remarkable advancements, with collaborative filtering (CF), modality-aware methods, and LLM-based approaches each contributing unique strengths to the field. However, each approach faces specific limitations that hinder their effectiveness. Collaborative filtering excels at leveraging user-item interaction data but struggles with challenges like data sparsity, cold-start problems, and limited scalability. Modality-aware systems address some of these issues by incorporating rich, multimodal data (e.g., text, images, audio), but they often require complex fusion mechanisms and extensive multimodal datasets. LLM-based methods leverage natural language understanding to provide personalized and explainable recommendations, yet they face challenges such as computational complexity, the need for large-scale training data, and the risk of generating biased or irrelevant recommendations.

These limitations formed the basis on which the authors of the paper *"Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System"*, where they propose a novel framework that combines the strengths of Collaborative filtering (CF= and LLMs to overcome these challenges. By integrating CF's ability to model user-item interactions with LLMs' capacity to process unstructured data and generate natural language explanations.

Chapter 2

A-LLMRec : Detailed approach

1 A-LLMRec

In this section we detailed the method presented in the paper "Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System."

1.1 Overview and Motivation

A-LLMRec is designed to combine the strengths of traditional collaborative filtering (CF) with the language understanding capabilities of large language models (LLMs) in a single efficient framework. The core idea is to enable a frozen, pre-trained CF recommender system to transfer its *collaborative knowledge* (i.e., high-quality user/item embeddings learned from historical interactions) into a frozen LLM. This is accomplished via a lightweight trainable alignment network. The primary advantages are:

1. **Model-Agnostic Integration:** Any CF model can be integrated without requiring its fine-tuning.
2. **Efficiency:** Only the alignment network is trained, avoiding the heavy computational cost of fine-tuning either the CF model or the LLM.

These advantages lead to faster training and inference times, as demonstrated in the paper [3].

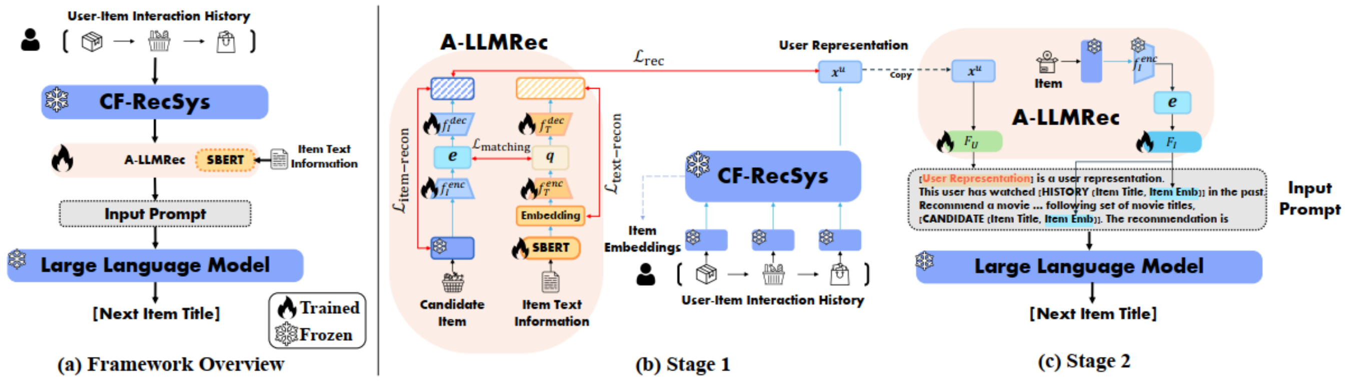

User-Item Interaction History
CF-RecSys
A-LLMRec
SBERT
Item Text Information
Input Prompt
Large Language Model
[Next Item Title]
Trained
Frozen
(a) Framework Overview
A-LLMRec
$\mathcal{L}_{\text{rec}}$
User Representation
x^u
Copy
f_I^{dec}
f_T^{dec}
$\mathcal{L}_{\text{matching}}$
e
q
f_I^{enc}
f_T^{enc}
Embedding
SBERT
$\mathcal{L}_{\text{item-recon}}$
$\mathcal{L}_{\text{text-recon}}$
Candidate Item
Item Text Information
Item Embeddings
CF-RecSys
User-Item Interaction History
(b) Stage 1
Item
F_u
F_I
[User Representation] is a user representation. This user has watched [HISTORY (Item Title, Item Emb)] in the past. Recommend a movie ... following set of movie titles, [CANDIDATE (Item Title, Item Emb)]. The recommendation is
Input Prompt
Large Language Model
[Next Item Title]
(c) Stage 2

Figure 2.1: (a) is the overview of A-LLMRec. (b) and (c) are the detailed architecture of Stage 1 and Stage 2, respectively.

1.2 Stage 1: Alignment of Collaborative and Textual Knowledge

Objective and Input Modalities

In Stage 1, the goal is to align two sources of information:

- **Collaborative knowledge:** Obtained from a frozen CF recommender system that produces item embeddings E_i, where the embedding matrix is $E \in \mathbb{R}^{|I|d}$.
- **Textual information:** Item descriptions (titles and descriptions) are converted into text embeddings using a pre-trained Sentence-BERT (SBERT) model. The SBERT model outputs a 768-dimensional vector $Q_i \in \mathbb{R}^{768}$ for each item, where the input is a concatenated string (e.g., `"Title:` t_i`, Description:` d_i`"`) .

Dual-Encoder Architecture

To bridge the gap between these modalities, two single-layer Multi-Layer Perceptrons (MLPs) are introduced:

- The **item encoder** $f^I_{\text{enc}} : \mathbb{R}^d \to \mathbb{R}^{d'}$ maps the CF-RecSys item embedding E_i to a latent space:
$$e_i = f^I_{\text{enc}}(E_i).$$
- The **text encoder** $f^T_{\text{enc}} : \mathbb{R}^{768} \to \mathbb{R}^{d'}$ projects the SBERT text embedding Q_i into the same latent space:
$$q_i = f^T_{\text{enc}}(Q_i).$$

Both encoders aim to align the representations so that the collaborative and textual knowledge can be jointly exploited .

Latent Space Matching and Reconstruction Losses

The alignment is enforced by minimizing the mean squared error (MSE) between the two latent embeddings:

$$L_{\text{matching}} = \mathbb{E}_{S_u \in S}\left[\sum_{i \in S_u} \text{MSE}\left(f^I_{\text{enc}}(E_i), f^T_{\text{enc}}(Q_i)\right)\right].$$

However, direct minimization of this loss can lead to over-smoothed representations (i.e., trivial solutions). To avoid this, the model includes decoders:

- A decoder f^I_{dec} reconstructs the original item embedding from $f^I_{\text{enc}}(E_i)$:
$$L_{\text{item-recon}} = \mathbb{E}_{S_u \in S}\left[\sum_{i \in S_u} \text{MSE}\left(E_i, f^I_{\text{dec}}\left(f^I_{\text{enc}}(E_i)\right)\right)\right].$$
- Similarly, a decoder f^T_{dec} reconstructs the SBERT embedding:
$$L_{\text{text-recon}} = \mathbb{E}_{S_u \in S}\left[\sum_{i \in S_u} \text{MSE}\left(Q_i, f^T_{\text{dec}}\left(f^T_{\text{enc}}(Q_i)\right)\right)\right].$$

These reconstruction losses preserve the distinctive information from each modality.

Incorporating Collaborative Recommendation Loss

To further embed collaborative knowledge, a recommendation loss is defined. Let x_u be the user representation extracted from the CF-RecSys based on the interaction sequence S_u. The loss compares the dot product between x_u and the latent embedding of a positive item $E_{i_u^+}$ versus a negative sample $E_{i_u^-}$:

$$L_{\text{rec}} = -\sum_{S_u \in S} \left[\log \sigma \left(s \left(x_u, f_{\text{dec}}^I \left(f_{\text{enc}}^I(E_{i_u^+}) \right) \right) \right) + \log \left(1 - \sigma \left(s \left(x_u, f_{\text{dec}}^I \left(f_{\text{enc}}^I(E_{i_u^-}) \right) \right) \right) \right) \right],$$

where $s(\cdot, \cdot)$ denotes the dot product and σ is the sigmoid function .

Overall Loss for Stage 1

The final loss for Stage 1 is a weighted sum:

$$L_{\text{stage-1}} = L_{\text{matching}} + \alpha \, L_{\text{item-recon}} + \beta \, L_{\text{text-recon}} + L_{\text{rec}},$$

where α and β are coefficients that balance the reconstruction terms. Once trained, the output $e_i = f_{\text{enc}}^I(E_i)$ becomes the *joint collaborative-text embedding.* For cold or unseen items, the alternative embedding $q_i = f_{\text{enc}}^T(Q_i)$ may be used .

1.3 Stage 2: Alignment between Joint Collaborative-Text Embedding and the LLM

Projection into the LLM Token Space

Stage 2 bridges the joint embedding space (dimension d') and the LLM token embedding space (dimension d_{token}). Two 2-layer MLPs are introduced:

- The user projection network $F_U : \mathbb{R}^d \to \mathbb{R}^{d_{\text{token}}}$ maps the user representation x_u to:

$$O_u = F_U(x_u).$$

- The item projection network $F_I : \mathbb{R}^{d'} \to \mathbb{R}^{d_{\text{token}}}$ maps the joint item embedding e_i to:

$$O_i = F_I(e_i).$$

These projected embeddings, O_u and O_i, are now compatible with the LLM's input space.

Prompt Design and Language Modeling Objective

A carefully structured prompt is constructed by embedding:

- The projected user representation O_u is injected as a “soft prompt” at the beginning to infuse user-specific collaborative signals.
- The candidate item information is provided by appending the projected item embeddings O_i alongside the natural language titles.

A typical prompt is formatted as follows:

```
[User Representation] is a user representation. This user has watched
[HISTORY (Item Titles, Item Emb)] in the past. Recommend a movie
for this user from the following set of movie titles, [CANDIDATE
(Item Titles, Item Emb)]. The recommendation is …
```

The LLM, which remains frozen, generates the recommendation token-by-token. The learning objective is to maximize the log-probability of the ground truth next item title:

$$\max_{\theta} \sum_{S_u \in S} \sum_{k=1}^{|y_u|} \log P_{\theta,\Theta}(y_{u,k} \mid p_u, y_{u,<k}),$$

where θ are the parameters in the projection networks, Θ are the frozen LLM parameters, p_u is the constructed prompt, and $y_{u,<k}$ are the preceding tokens.

2 Key Advantages and Practical Considerations

The two-stage alignment approach of A-LLMRec provides several benefits:

1. **Model-Agnostic Integration:** Since only the alignment networks are trainable, any state-of-the-art CF model can be integrated. This modular design allows for straightforward updates as new CF models emerge.

2. **Efficiency:** By avoiding fine-tuning of the large CF-RecSys and LLM components, the system achieves faster training and inference speeds, making it viable for large-scale and real-time applications.

3. **Robustness Across Scenarios:** The joint embedding encapsulates both collaborative and textual signals, which improves recommendation performance in warm, cold, and cross-domain settings.

In summary, the method leverages a two-stage process: Stage 1 aligns collaborative and textual modalities into a joint embedding, while Stage 2 projects this embedding into the LLM token space, enabling natural language prompts that guide the LLM to generate precise, personalized recommendations.

3 Critical analysis of A-LLMRec

While A-LLMRec presents an innovative integration of collaborative filtering with large language models, several potential disadvantages and limitations can be identified:

1. Suboptimal Cross-Modal Alignment

Issue: We see the alignment might really be a good idea but the use of MSE loss blindly aligns embeddings without preserving the relative relationships between items. For example : If two items (e.g., Avengers and Barbie) have similar CF embeddings (e.g., both are popular), but very different text, MSE will still try to align them, leading to meaningless overlaps.

Impact: This could lead to poor alignment, Avengers and Barbie (with similar CF embeddings but different text) might collapse into the same latent space. The model loses the ability to distinguish between them, leading to poor recommendations. A contrastive loss (e.g., InfoNCE) or cosine similarity might better preserve relative similarities (Avengers stays close to Guardians (similar in CF/text) and far from Barbie (dissimilar in text))

2. Heavy Dependence on Pre-trained Models

A-LLMRec relies on pre-trained components such as the CF recommender system and Sentence-BERT (SBERT) for obtaining item embeddings and text representations. If these models yield suboptimal or noisy embeddings, the joint collaborative-text embedding may suffer, potentially degrading recommendation quality.

3. Dependence on Prompt Engineering

Stage 2 of the method relies heavily on a well-designed prompt to inject collaborative knowledge into the LLM. While the paper provides a structured prompt format, the overall performance is sensitive to prompt engineering. Poorly designed prompts or inconsistencies across domains might lead to suboptimal recommendations, thereby necessitating additional manual tuning.

4 Our Contribution

1. Enhancing the Alignment Network Architecture

The simplicity of the alignment network, utilizing only shallow MLPs (1 NN layer), also presents a potential area for scrutiny. While this design choice contributes to the efficiency of the model by reducing the number of trainable parameters, it raises concerns about the capacity of these networks to effectively capture the complex, non-linear relationships between the different embedding spaces involved. Specifically, the following alignments might be limited:

- The alignment between CF-RecSys embeddings and SBERT text embeddings.
- The mapping of joint collaborative-text embeddings to LLM token embeddings.
- The projection of user representations to the LLM's token space.

It is possible that these simple networks might lead to a bottleneck in the transfer of knowledge, preventing the full richness of the collaborative signals from being effectively transfer to the LLM. Our new configuration consisted of 4 hidden layers, each different hidden units, and incorporated a dropout rate of 0.2 to avoid overfitting and trained with AdamW (learning rate 3e-4, weight decay 0.1). Additionally, ReLU activations were applied across all layers to introduce necessary non-linearity.

This approach improved the network's capacity to transfer rich collaborative signals and gave a better performance.

2. Replacing MSE with InfoNCE Loss for Better Embedding Alignment

A second significant contribution is to consider replacing the MSE loss with the InfoNCE loss for aligning item embeddings from the CF model with their corresponding SBERT text embeddings. Unlike MSE, which forces embeddings to match in absolute value, the InfoNCE loss preserves the relative structure of the embedding space by explicitly separating negatives.

InfoNCE Loss Details:

- **Anchor:** The item embedding e_i obtained from the CF-RecSys (e.g., SASRec).
- **Positive:** The text embedding q_i from SBERT corresponding to the same item i.
- **Negatives:** The text embeddings q_j from SBERT of other items where $i \neq j$.

The goal is to maximize the similarity between e_i and q_i while minimizing the similarity between e_i and q_j.

Negative Sampling Considerations:

- **False Negatives:** Random negatives may include items with semantically similar text (e.g., two sci-fi movies with nearly identical SBERT embeddings), potentially penalizing the model even when such items share similarities.

- **Mitigation via Semantic Filtering:** To counter this, one can apply semantic filtering by excluding items with SBERT similarity above a threshold θ from being considered as negatives. Specifically, for a given anchor i, discard any item j for which $\text{sim}(q_i, q_j) > \theta$, and use the remaining in-batch items as negatives.

This InfoNCE-based approach, with explicit negative sampling which means we can avoid adding $f^T_{\text{dec}}(E_i)$ and $f^I_{\text{dec}}(E_i)$ which they were just to overcome the over-smoothing problem.

Replacing the Model with a Larger OPT-13B Model

Our third key contribution is the **replacement of the base model with OPT-13B**, a significantly larger variant within the same model family. The motivation behind this change is to leverage the increased capacity of a larger language model for enhanced representation learning, better generalization, and improved recommendation quality.

Advantages of Using OPT-13B:

- **Improved Representation Learning:** The 13-billion-parameter model captures more complex interactions between collaborative filtering (CF) and text embeddings, leading to richer feature representations.
- **Better Alignment with CF-RecSys:** The larger model provides a more expressive latent space, improving the alignment between CF embeddings and textual embeddings derived from SBERT.
- **Enhanced Generalization:** The increased model capacity helps avoid overfitting to specific patterns in training data and leads to better generalization on unseen recommendations.

Impact on Performance: The use of OPT-13B resulted in **notable improvements in NDCG@ 10 and HR@10**, confirming that a larger model can better capture user-item relationships. However, this performance gain came at the cost of **increased training time and resource consumption**, highlighting a trade-off between accuracy and computational efficiency.

Chapter 3

Conclusion

In summary, while A-LLMRec innovatively aligns collaborative and textual information through a two-stage process, there are notable technical challenges. These include issues related to the sensitivity of MSE-based losses, the limited capacity of shallow projection networks, and the difficulties of aligning frozen components with inherently different embedding distributions. The contributions proposed above—enhancing the alignment network with deeper architectures and replacing MSE with an InfoNCE loss augmented by semantic filtering—offer promising directions for addressing these limitations and improving the overall transfer of collaborative knowledge into the LLM framework.

Bibliography

[1] Aman Chadha and Vinija Jain. Graph neural networks. *Distilled AI*, 2020. https://aman.ai.

[2] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[3] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system, 2024.

[4] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[5] Wanqi Ma, Weike Pan, and Zhong Ming. SCF: Structured collaborative filtering with heterogeneous implicit feedback. *Knowledge-based systems*, 258:109999, 12 2022.

[6] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. Ultragcn: Ultra simplification of graph convolutional networks for recommendation. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[7] Chau Pham. Graph convolutional networks (gcn). https://www.topbots.com/graph-convolutional-networks/, 2022.

[8] Jure Leskovec (Stanford). Excellent slides on graph representation learning. https://drive.google.com/file/d/1By3udbOt10moIcSEgUQ0TR9twQX9Aq0G/.

[9] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and M. Coates. Neighbor interaction aware graph convolution networks for recommendation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[10] Junhua Sun, Wei Guo, Dengcheng Zhang, Yingxue Zhang, Florence Regol, Yaochen Hu, Huifeng Guo, Ruiming Tang, Han Yuan, Xiuqiang He, and Mark Coates. A Framework for Recommending Accurate and Diverse Items Using Bayesian Graph Convolutional Neural Networks. *Virtual Event*, 8 2020.

[11] WelcomeAIOverlords. Video graph convolutional networks (gcns) made simple. https://www.youtube.com/watch?v=2KRAOZIULzw.