Sparse Autoencoders in Collaborative Filtering Enhanced LLM-based Recommender Systems

Xinyu He
University of Illinois at Urbana-Champaign
Champaign, IL, USA
xhe34@illinois.edu

Jose Sepulveda
Amazon
Seattle, WA, USA
joseveda@amazon.com

Fei Wang
Amazon
Sunnyvale, CA, USA
feiww@amazon.com

Hanghang Tong
University of Illinois at Urbana-Champaign
Champaign, IL, USA
htong@illinois.edu

Abstract

Large language models (LLM) have demonstrated remarkable capability in recommendation tasks. Recently, efforts have been made to further enhance LLM performance with collaborative knowledge learned from traditional recommender systems. One approach is to inject learned embeddings into LLM prompts through a trainable projector, yet these embeddings could carry noisy or irrelevant information. In this paper, we propose using sparse autoencoders to improve input prompts. We show that sparse autoencoders can learn highly interpretable embeddings and extract key collaborative features in the case of recommender systems. With the help of sparse autoencoders, we are able to extract collaborative features to augment input prompts. By capturing Top$K$ features of each item, we mitigate noisy information from item embeddings, therefore sparse autoencoders can also help with denoising embeddings in prompts. We develop two methods that utilize sparse autoencoders to augment or denoise input prompts. We evaluate the proposed methods on three real-world datasets and both show promising performance improvements.

CCS Concepts

• **Information systems → Recommender systems**.

Keywords

Recommender Systems; Large Language Models; Collaborative Filtering; Sparse Autoencoders

ACM Reference Format:
Xinyu He, Jose Sepulveda, Fei Wang, and Hanghang Tong. 2025. Sparse Autoencoders in Collaborative Filtering Enhanced LLM-based Recommender Systems. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746252.3760957

1 Introduction

Collaborative filtering based recommender systems have been widely explored in the past decades, including matrix factorization [6], sequential recommendation [9, 20], and neural graph collaborative filtering [5, 25]. Recently, inspired by the remarkable capabilities of large language models (LLMs), researchers are exploring the potential of LLMs in recommendation tasks, e.g., conversational recommendation [4, 7], item recommendation [1, 13], cold-start recommendation [19, 23], etc. Although LLMs are empowered with extensive external world knowledge and advanced reasoning ability, they are still outperformed by traditional collaborative filtering based recommender systems due to the lack of collaborative knowledge [10]. Previous work [26] finds that those LLM-based recommender systems heavily rely on textual information. Consequently, LLM-based models might be outperformed by collaborative filtering models when abundant user-item interactions are available. To leverage advantages from both LLM and collaborative filtering, more recent studies design ways to seamlessly incorporate collaborative knowledge learned from traditional recommender systems into LLMs. One solution that has been proved effective involves leveraging user and item embeddings learned from collaborative filtering based recommender systems. These embeddings are injected into input prompts as token embeddings through a trainable projector [10, 12]. Fine-tuned with interaction data, trainable projector aligns the embedding space of collaborative filtering recommender systems with the token space of LLMs. Hence, collaborative knowledge encoded in these embeddings is seamlessly incorporated into the input prompts of LLMs. As these embeddings identify the unique identities of users and items, they are also referred to as ID embeddings[13].

Although ID embeddings are proved to be helpful, they fall short of illustrating the information in a text-like format, which may not align optimally with LLMs. Furthermore, ID embeddings often contain noisy information that stems from noisy interaction datasets. For recommender systems that utilize text information (or other modality information), e.g. [10], ID embeddings might be further affected by irrelevant messages in texts. In this paper, we delve into the prompt engineering for collaborative filtering enhanced LLM-based recommender systems, especially focusing on (1) augmenting prompts with extracted collaborative features and (2) denoising ID embeddings to improve model performance.

Previous works have demonstrated the remarkable capability of sparse autoencoders (SAEs) for interpretable feature extraction. To name a few, [30] extracts low-dimensional features for more efficient and effective matrix factorization, [2, 3] explain the activation patterns in LLMs with sparse autoencoders; while sparse

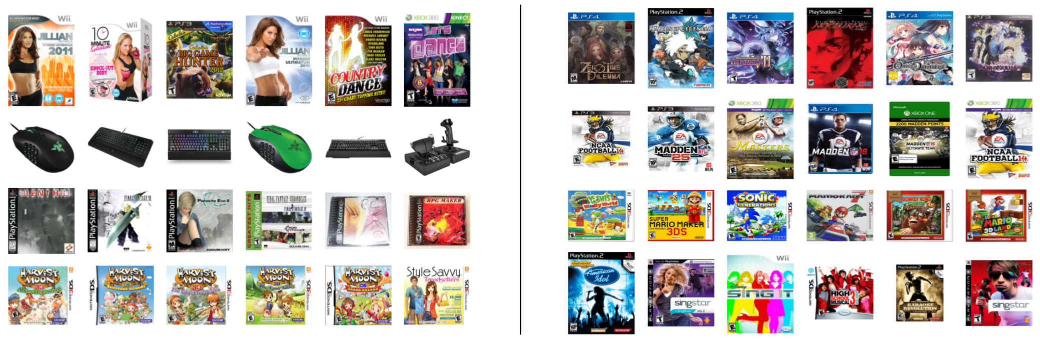


Figure 1: Examples of extracted features from Amazon Games dataset[1] with SAE and SimGCL. Each line corresponds to one generated feature. Each line displays items that are activated in the corresponding latent.

autoencoders are also applied to fetch high quality data for surface microseismic data [11]. For collaborative filtering, we train a sparse autoencoder with embeddings learned from SimGCL [25] and visualize the extracted features in Fig. 1. We can clearly observe that sparse autoencoders are indeed capable of generating interpretable features for collaborative filtering recommendation. For example, features in the right column can be interpreted as 'japanese anime game', 'football video games', 'child anime games', 'music singing games'. Inspired by this observation, we seek to leverage sparse autoencoders to extract Top$K$ collaborative features from ID embeddings and use such extracted features to reconstruct ID embeddings to improve the quality of prompts.

Our main contributions are summarized as follows.

- **Insights.** We show that sparse autoencoders are capable of learning highly interpretable features in the case of collaborative filtering recommendation.
- **Methods.** We propose two methods for leveraging sparse autoencoders to improve prompts of collaborative filtering enhanced LLM-based recommender systems.
- **Evaluations.** Proposed methods are evaluated on three real-world datasets and they both outperform state-of-the-art collaborative filtering and LLM-based methods.

2 Related Works

LLMs for recommendation. With the superb knowledge and reasoning ability of LLMs, researchers discover that LLMs are state-of-the-art zero-shot recommenders. [4, 7, 24] apply the in-context learning technique to adapt LLMs to recommendation by adding interaction and task description contexts in input prompts.[1] points out that the performance of LLMs is suboptimal due to the inadequate recommendation data during pretraining. Therefore, the authors propose to fine-tune LLMs with Alpaca tuning[21] and with LoRA[8] to fit recommendation datasets. Researchers further explore ways to incorporate collaborative information into LLMs. [10, 12, 29] use ID embeddings and map them into LLM token space. Moreover, [12] applies curriculum learning strategy with LoRA and [10] aligns ID embeddings with textual information, [29] separately finetunes LoRA and collaborative information learning modules in two steps. [31] expands the LLM vocabulary and designs a mutually-regularized strategy to pretrain the new token embeddings. [28] binarize ID embeddings to enable bitwise operations of LLMs.

Sparse autoencoders (SAEs). Sparse autoencoders serve as a powerful tool for interpretable machine learning and dictionary learning. Multiple types of regularizations have been studied to ensure the sparsity of latent states, including L_1 penalty [2], KL divergence [15] and Top$K$ activation [14]. Sparse dictionary learning [17] is closely related to our work, which aims to find an over-complete 'atom' feature set with SAEs, so that embeddings can be decomposed into embeddings of 'atoms'. Embeddings of those 'atoms' are also referred to as base vectors.

3 Proposed frameworks

In this section, we first introduce the sparse autoencoder for LLM-based recommendation (SAULREC) framework in Section 3.1, then introduce our design of sparse autoencoder in Section 3.2.

3.1 SAULREC Framework

For a collaborative filtering enhanced LLM-based recommender system that utilizes ID embeddings (e.g., [10, 12]), we generally formulate the model as three modules: (1) a pretrained collaborative filtering model $\mathcal{R}$, (2) an additional embedding processing module $\mathcal{F}$, and (3) a LLM-based recommender system which takes ID embeddings as input.

The pretrained collaborative filtering model $\mathcal{R}$ is implemented with SASRec [9] which learns user and item embeddings from sequential interaction data. We denote embeddings of user u and item i from collaborative filtering model as x_u and x_i respectively.

Following [10], an additional item embedding processing module $\mathcal{F}$ is added to align item embeddings from collaborative filtering with textual embedding encoded with SBERT [18]. For an item i with item embedding x_i and textual embedding of its description c_i, $\mathcal{F}$ is implemented with 2 autoencoders, $f^{\text{text}}(c_i)$ and MLP1 = $f^{\text{CF}}(x_i)$,

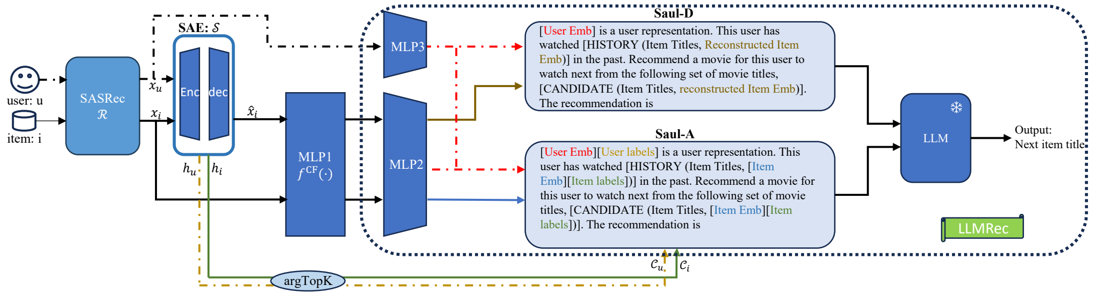
$$f^{\text{CF}}(x_i) = f^{\text{CF}}_{\text{dec}}(f^{\text{CF}}_{\text{enc}}(x_i)) \tag{1}$$

$$c_i = \text{SBERT}(\{\text{textual description of item i}\}) \tag{2}$$

$$f^{\text{text}}(c_i) = f^{\text{text}}_{\text{dec}}(f^{\text{text}}_{\text{enc}}(c_i)). \tag{3}$$

$f^{\text{CF}}_{\text{enc}}, f^{\text{CF}}_{\text{dec}}$ are the encoder and decoder of f^{CF}, $f^{\text{text}}_{\text{enc}}, f^{\text{text}}_{\text{dec}}$ are the encoder and decoder of f^{text}, which are all implemented with fully-connected layers. To align item embeddings with textual description, MSE loss is applied to minimize the distance between $f^{\text{CF}}_{\text{enc}}(x_i)$

[1] https://mcauleylab.ucsd.edu/public_datasets/data/amazon_2023/benchmark/5core/rating_only/Video_Games.csv.gz



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

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

Figure 2: Framework of SaulRec. Solid lines represent item information, dashed lines represent user information.

and $f^{\text{text}}_{\text{enc}}(c_i)$. Therefore, trained with a combination of reconstruction loss $\|f^{\text{CF}}(x_i) - x_i\|_2^2$, alignment loss $\|f^{\text{CF}}_{\text{enc}}(x_i) - f^{\text{text}}_{\text{enc}}(c_i)\|_2^2$ and cross entropy recommendation loss $-\log\sigma(x_u, f^{\text{CF}}(x_i)) + \log(1 - \sigma(x_u, f^{\text{CF}}(x_i)))$, the reconstructed item embedding from MLP1 incorporates textual information while retaining information from recommendation data. These reconstructed item embeddings are then fed into LLM-based recommender system as ID embeddings.

To project ID embeddings into the LLM token space, MLP2 and MLP3 serve as user and item ID embedding projectors respectively. With carefully designed input prompt p_u, these two projectors are optimized to maximize the next token probability in the LLM output

$$\max_{\theta} \sum_{j=1}^{\|y^u\|} \log P_{\theta,\Theta}(y^u_j | p_u, y^u_{<j}) \tag{4}$$

where θ is parameters of MLP2 and MLP3, y^u is ground truth next item title, p_u is input prompt corresponding to user u, Θ is LLM parameters, $P_{\theta,\Theta}$ is the output next token probability from LLM, $y^u_j, y^u_{<j}$ denote the j-th token and the tokens before the j-th token.

However, this framework might fall short in fully illuminating collaborative knowledge in a text-like format, which may not align optimally with LLMs. Therefore, we resort to sparse autoencoders to extract collaborative features from $\mathcal{R}$ and seek to augment LLM prompts with extracted item labels. Furthermore, as there might be noisy and irrelevant information encoded in ID embeddings, we leverage reconstructed embedding from SAE to help LLM focus on the extracted user/item Top$K$ features only.

3.2 Sparse Autoencoder

To extract the collaborative features from recommender systems, we apply Top$K$ activation based SAEs because this activation allows us to control the number of active latents. Here, a latent refers to one dimension in the embedding and the active latents denote the dimensions that have non-zero values. Top$K$ activation keeps only the k largest values in an embedding and zeros the rest. An SAE with Top$K$ activation is formulated as

$$\begin{aligned} x &= \frac{x-\mu}{\sigma}, \quad \mu, \sigma = x.mean(), x.std() \\ h &= \text{Top}K(h_0), h_0 = W_{\text{enc}}(x - b_{\text{pre}}) + b_{\text{enc}} \\ \hat{x} &= \mu + \sigma x_{\text{norm}}, x_{\text{norm}} = W_{\text{dec}} h + b_{\text{pre}} \end{aligned} \tag{5}$$

where $x \in \mathcal{R}^d$ is the input embedding and $\hat{x} \in \mathcal{R}^d$ is the reconstructed embedding. First, we normalize the input embedding x. Next, we subtract bias in dataset $b_{\text{pre}} \in \mathcal{R}^d$ and pass the embedding through an encoder with the weight $W_{\text{enc}} \in \mathcal{R}^{d \times d_h}$ and the bias $b_{\text{enc}} \in \mathcal{R}^{d_h}$. Then Top$K$ activation is applied to ensure the sparsity of the hidden embedding h. Lastly, the hidden embedding is passed through a decoder with parameters $W_{\text{dec}} \in \mathcal{R}^{d_h \times d}, b_{\text{dec}} \in \mathcal{R}^d$ and rescaled back with the mean and the standard deviation to reconstruct x. Note that SAEs for sparse dictionary learning are over-complete autoencoders with $d_h > d$ to learn a comprehensive feature set. However, the reconstruction error is not zero since we have the Top$K$ sparsity constraint on hidden embeddings.

In this structure, the latents h_0 before Top$K$ activation tend to be near zero if W_{enc} has much smaller values compared to W_{dec}. This makes the training process unstable, and it is hard to tell whether the selected Top$K$ latents are really important because they have close values compared to non-Top$K$ latents. We resort to the tied decoder [3] to resolve this issue, where $W_{\text{dec}} = W^T_{\text{enc}}$ is enforced during the whole training process. Another common problem in Top$K$ SAEs is dead latents, where some latents are rarely or never activated during training. Furthermore, disentanglement between extracted features are required, as repeated features are meaningless. Therefore, we add a uniformity regularization [22] on hidden embeddings h for SAE training.

$$\mathcal{L}_U = \frac{1}{|\mathcal{B}|^2} \sum_{i,i' \in \mathcal{B}} \exp(-2\|\frac{h_i}{\|h_i\|_2} - \frac{h_{i'}}{\|h_{i'}\|_2}\|_2^2) \tag{6}$$

where $\mathcal{B}$ is a sampled item batch, $h_i, h_{i'}$ are hidden embeddings of items i and i' respectively. Together with the MSE reconstruction loss, the training loss is defined by

$$\mathcal{L} = \mathcal{L}_{MSE} + \beta\mathcal{L}_U, \quad \mathcal{L}_{MSE} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \|x_i - \hat{x}_i\|_2^2 \tag{7}$$

where β is a hyperparameter.

Finally, we train the proposed SAE $\mathcal{S}$ with only item embeddings from the pretrained collaborative filtering model $\mathcal{R}$. Note that SAE trained with only item embeddings is sufficient because base vectors in the user space but not in the item space will not affect the result of the dot product. Therefore, it will not affect item ranking for recommendation. The Top$K$ collaborative labels for each user u and item i are then defined as

$$\mathcal{C}_u = \text{argTopK}(h_u), \quad \mathcal{C}_i = \text{argTopK}(h_i) \tag{8}$$

where argTopK() returns the indices of the TopK largest values in the hidden embeddings of user u and item i. Note that h_i has only K nonzero values, therefore $\hat{x}_i$ is reconstructed based on only the extracted Top$K$ collaborative features in $\mathcal{C}_i$. Thus, noises and less important information are removed in reconstructed embeddings.

Extracted collaborative features and reconstructed item embeddings are then integrated into LLM prompts to augment (Saul-A) or denoise (Saul-D) as illustrated in Fig. 2. To augment LLM prompts with extracted user/item labels C_u/C_i (Saul-A), the list of numerical values are appended after user/item embeddings as a textual form of user/item representations. To denoise LLM prompts (Saul-D), reconstructed item embeddings are inserted after item titles.

4 Experiments

4.1 Experimental Setting

Datasets. To evaluate the performance of SaulRec, we follow [10] to use three Amazon datasets [16] for experiments. Data statistics are summarized in Table 1.

Table 1: Data Statistics

Dataset	#Users	#Items	Avg.Len
Luxury (Luxury Beauty)	11690	6534	4.15
Toys (Toys and Games)	42597	75377	11.10
Movies (Movies and TV)	172229	64226	8.14

Baselines. We compare our proposed frameworks with three collaborative filtering recommender systems (SimGCL [25], NCF [6], SASRec [9]) and two collaborative filtering enhanced LLM-based recommender systems (LLARA[12], A-LLMRec[10]).

Metric and Evaluation Protocol. User purchase sequences are divided into train, validation and test sets. For user sequence $[i_1, i_2, ..., i_n]$, i_n is considered as the next item in the test set, i_{n-1} is the next item in the validation set, $i_1 \sim i_{n-2}$ are used as the training set. To evaluate the performance of all models, 19 randomly selected items for each user with no previous interactions are selected. Models make the next item prediction based on the candidate set composed of selected 19 negative candidates and the ground truth next item. Hit@1 is applied as the evaluation metric.

Implementation. Following [10], we adopt OPT-6.7B [27] as the backbone LLM model. We remove users and items with fewer than 4 interactions in the Luxury dataset. We remove users and items with fewer than 5 interactions and interactions with rating less than 4 in the Movies dataset. We remove users and items with fewer than 10 interactions and interactions with rating less than 4 in the Toys dataset. For SaulRec, we set $d = 50, d_h = 2,048, K = 5$ to train Toys and Movies datasets, and set $d = 50, d_h = 256, K = 5$ to train the Luxury dataset. For LLARA, we set number of accumulated batches to 4 and batch size to 2 due to memory constraints given our large datasets. Experiments are conducted with NVIDIA A100-SXM4-80GB GPU. Our codes are available at Github.

Table 2: Experimental results (Hit@1). Best results are in bold, second best results are underlined.

Method	Luxury	Toys	Movies
SimGCL	0.3490	0.2690	0.4461
NCF	0.4191	0.3357	0.5577
SASRec	0.5211	0.2815	<u>0.5990</u>
LLARA	0.4336	0.3495	0.5705
A-LLMRec	0.5578	0.3609	0.5767
Saul-D	**0.5715**	**0.3967**	**0.6055**
Saul-A	<u>0.5579</u>	<u>0.3736</u>	0.5798

4.2 Overall Performance

Experimental results are shown in Table 2. Our proposed methods outperform all baselines, including the base model A-LLMRec, while Saul-D performs better than Saul-A. Collaborative filtering enhanced LLM-based recommender systems generally perform better than collaborative filtering models, except for the Movies dataset where movie titles include many non-english words that affects the reasoning performance of LLMs. As we represent extracted features as index numbers in Saul-A, it might lose the similarity information between features, which is the potential reason of why Saul-D reaches a better performance than Saul-A.

4.3 Training Efficiency

Given the long training time of many LLM-based recommender systems, it is important to explore the learning efficiency of LLM-based recommender systems. In this experiment, we report the performance of our models and base model A-LLMRec after 1 epoch of training in Table 3. We can see that Saul-A reaches the best performance when trained with limited rounds especially for larger datasets, in contrast to the overall performance where Saul-D reached the best. This implies that explicitly representing collaborative knowledge as natural languages better aligns with the knowledge of LLM, making LLM easier to understand the input prompts and to be tuned.

Table 3: Hit@1 after training 1 epoch.

Method	Luxury	Toys	Movies
A-LLMRec	0.4583	<u>0.3559</u>	0.5519
Saul-D	**0.5435**	0.3499	<u>0.5632</u>
Saul-A	<u>0.5320</u>	**0.3685**	**0.5768**

4.4 Case Studies

For a user who has sequentially watched "The Lord Of The Rings: Trilogy", "Following-Complete Series 1", "The Day the Earth Stood Still VHS", "Nightbreed", "Your Inner Fish", "Dawn of the Planet of the Apes"," Nature: Fabulous Frogs", "Gattaca", "Matinee VHS", "Beyond the Myth", A-LLMRec predicted "Nightingale" as the next item. However, the proposed SaulRec 'knows' that user labels are *[30, 565, 743, 48, 49]*, labels for "Nightingale" and "Game of Thrones: Season 5" are *[684, 562, 405, 47, 52]* and *[48, 442, 576, 565, 88]* respectively, "Nightbreed", "Gattaca", and "Beyond the Myth" in the watch history are also labeled as *48*, and "Dawn of the Planet of the Apes" in the watch history is also labeled *565*. Therefore, both Saul-A and Saul-D successfully predicted "Game of Thrones: Season 5" as the next item because it has labels *48, 565* which overlap with user labels and labels of previously watched movies, while "Nightintgale" does not share any common labels with them.

5 Conclusion

In this paper, we propose a novel collaborative filtering enhanced LLM-based recommender system SaulRec. The main idea is to leverage the power of sparse autoencoders (SAEs) to improve LLM prompts. With extracted collaborative features from SAEs, input prompts are augmented with knowledge from collaborative filtering in the form of natural language. By only retaining Top$K$ features of items to reconstruct item embeddings, noisy information are mitigated and quality of input prompts are improved. Experiments demonstrate the effectiveness and the efficiency of our methods.

Acknowledgments

This work is partially supported by NSF (2316233) and AFOSR (FA9550-24-1-0002). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

6 GenAI Usage Disclosure

We know that the ACM's Authorship Policy requires full disclosure of all use of generative AI tools in all stages of the research (including the code and data) and the writing. No GenAI tools were used in any stage of the research, nor in the writing.

References

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

[2] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[3] Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[5] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[7] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 720–730, 2023.

[8] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[9] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206, 2018.

[10] 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. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1395–1406, 2024.

[11] Xuegui Li, Shuo Feng, Nan Hou, Ruyi Wang, Hanyang Li, Ming Gao, and Siyuan Li. Surface microseismic data denoising based on sparse autoencoder and kalman filter. *Systems Science & Control Engineering*, 10(1):616–628, 2022.

[12] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1785–1795, 2024.

[13] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. Llm-rec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780*, 2023.

[14] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[15] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[16] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[17] Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

[18] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[19] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*, pages 890–896, 2023.

[20] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1441–1450, New York, NY, USA, 2019. Association for Computing Machinery.

[21] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[22] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 1816–1825, 2022.

[23] Jianling Wang, Haokai Lu, James Caverlee, Ed H Chi, and Minmin Chen. Large language models as data augmenters for cold-start item recommendation. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 726–729, 2024.

[24] Lei Wang and Ee-Peng Lim. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*, 2023.

[25] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 1294–1303, 2022.

[26] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2639–2649, 2023.

[27] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[28] Yang Zhang, Keqin Bao, Ming Yan, Wenjie Wang, Fuli Feng, and Xiangnan He. Text-like encoding of collaborative information in large language models for recommendation. *arXiv preprint arXiv:2406.03210*, 2024.

[29] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[30] Yihao Zhang, Chu Zhao, Mian Chen, and Meng Yuan. Integrating stacked sparse auto-encoder into matrix factorization for rating prediction. *IEEE Access*, 9:17641–17648, 2021.

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