Recommender Systems Amid the Rise of DL and LLMs

Abstract

Recommender Systems have been of interest because they not only satisfy users' thoughts and needs better but also have direct impacts on business revenue. During lectures, the professor briefly talked about both Memory-based Collaborative Filtering(CF) and Model-based Collaborative Filtering approaches along with their applications and difficulties in real-world scenarios. In this paper, I delved into Model-based CF approaches because they scale well for most real-world scenarios and huge amounts of customers-item interactions, the model developed can be optimized and updated periodically to keep up with the latest patterns, and it offers the opportunity for a variety of techniques to realize its potentials. Specifically, I will focus on recent developments in recommender systems with deep neural networks and large language models and illustrate their techniques and implications. Possible future directions and trends from my perspective are pointed out in the end.

1 Introduction

A system for filtering information that matches users' preferences with relevant items is called the Recommender System. It offers solutions for identifying pertinent items or topics from huge amounts of information[13]. Recommender systems are omnipresent in the current world, social media news and advertisements, videos and music recommendations, and online stores and electronic commerce all rely on efficient and robust recommendation systems. As a result, recommender systems have been an attractive topic for both research and industry Fig.1. Large-scale service providers and platforms provide personalized and catered services along with intelligent and manageable recommendations to customers based on their private data and customers' historical behaviors. Before diving into technical details, let's look at the overall pipeline of a recommendation system2. Starting from raw data sources which include explicit feedback (ratings, likes and comments, etc) and implicit feedback(purchase and browse history, watching time, etc), the algorithm engineer will develop descriptive features based on raw data information, these features are often stored in a centralized data warehouse to facilitate experiment and deployment[2]. Recommendation Models fall into three categories: hybrid systems, content-based filtering, and collaborative filtering. Collaborative filtering(CF) leverages collaboration among users to make recommendations, whereas Content-based filtering uses an item-context matrix in the recommendation process. CF is further subdivided into model-based CF and memory-based CF, which encompasses user-based and item-based. The differences between user-based and item-based CF lie in the primary similarities being emphasized. Model-based CF is becoming popular because of its scalability and better capability[24, 9]. A hybrid System combines the advantages of both methods offers more flexibility and delivers better results[29].

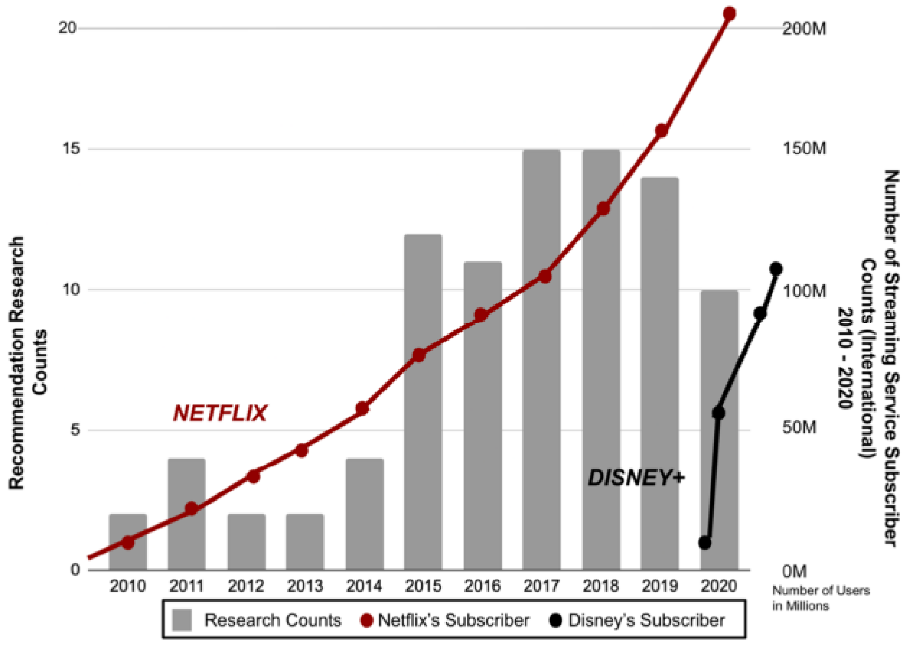


Figure 1: Recommendation research trend aligns with the development of Content Providers[13]

Recommender systems play a crucial role in improving click-through and conversion rates. However, their widespread applications and real scenarios also pose several challenges:

Cold Start : Insufficient information when recommending to new users or for new items

Data Sparsity : Most users have limited feedback, which leads to sparse estimation

Scalability : Imperceptible latency for large-scale recommendation with millions of objects

Diversity : User preference might lead to a narrower scope and stale recommendation

In industry, recommender systems are often divided into several stages Fig.3 such as retrieval, pre-ranking, ranking, and re-ranking serve different purposes with different techniques and strategies[17]. In the following sections, we cover a few well-established and classical models as well as recent developments with DL and LLMs in this area, including state-of-the-art research outcomes.

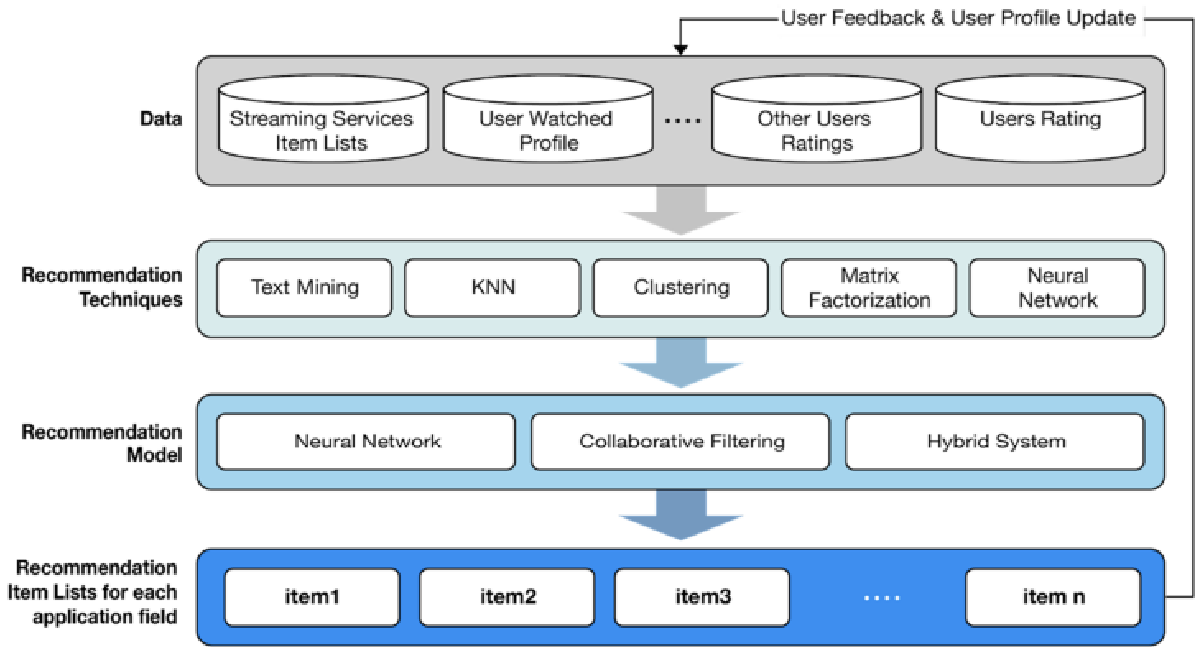


Figure 2: Recommendation Models and Techniques[13]

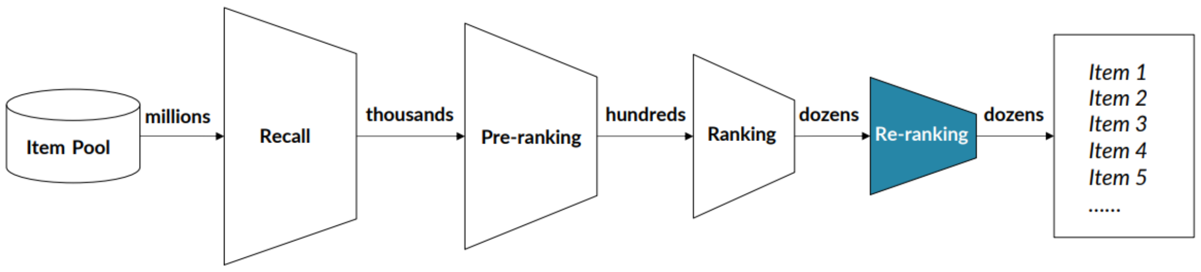


Figure 3: Industrial RS pipeline[17]

2 DL Based Algorithms

2.1 Matrix Factorization

Matrix Factorization[15] makes the assumption that the interaction matrix can be factorized to yield latent factors that represent user and items. Suppose the interaction(user-item) matrix $M \in R^{m\times n}$, latent matrix for the user $U \in R^{m\times k}$, and latent matrix for items$V \in R^{n\times k}$. The reconstructed interaction matrix is:

$$\hat{M} \approx UV^T$$

In addition, to express the average response, terms for item bias and user bias are included. To minimize the difference between true and reconstructed interaction matrix, we can optimize the objective function with SGD and Adam.

$$J = \underset{U,V,b}{\arg\min} \sum_{(u,i)\in\mathcal{K}} \|M_{ui} - \hat{M}_{ui}\|^2 + \lambda(\|U\|^2 + \|V\|^2 + b_u^2 + b_i^2)$$

Matrix factorization addresses sparsity and scalability issues to some extent but relies solely on the interaction matrix, which lacks information and integration of user characteristics, item attributes, and contextual features.

2.2 Factorization Machines

Factorization Machines[22] is a generalization and extension of matrix factorization and linear regression model, higher interaction between features are incorporated based on their inner product. The FM has excellent computational efficiency and scalability, and it can be utilized across different stages of the recommendation pipeline. The formula is:

$$\hat{y}(X) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=i+1}^{d} \langle v_i, v_j\rangle x_i x_j$$

feature embeddings $V \in \mathbb{R}^{n\times k}$, row v_i of V represents the latent vector associated with x_i, and $\langle v_i, v_j\rangle$ is the interaction between x_i and x_j. If we rearrange the third term, the computational complexity decreases from $O(kd^2)$ to $O(kd)$ [29].

$$\sum_{i=1}^{d}\sum_{j=i+1}^{d} \langle v_i, v_j\rangle x_i x_j = \frac{1}{2}\sum_{l=1}^{k}\left((\sum_{i=1}^{d} v_{i,l} x_i)^2 - \sum_{i=1}^{d} v_{i,l}^2 x_i^2 \right)$$

To overcome the complexity and inflexibility of feature engineering and the over-generalization problem that appears in deep neural networks, Google[5] proposed Wide & Deep learning to combine the advantages of generalization and memorization for recommendation systems. The Wide component captures explicit and co-occurrence patterns and is optimized with FTRL[19] algorithm while the deep component generalizes well to unexplored feature interactions and is optimized with

Adagrad.

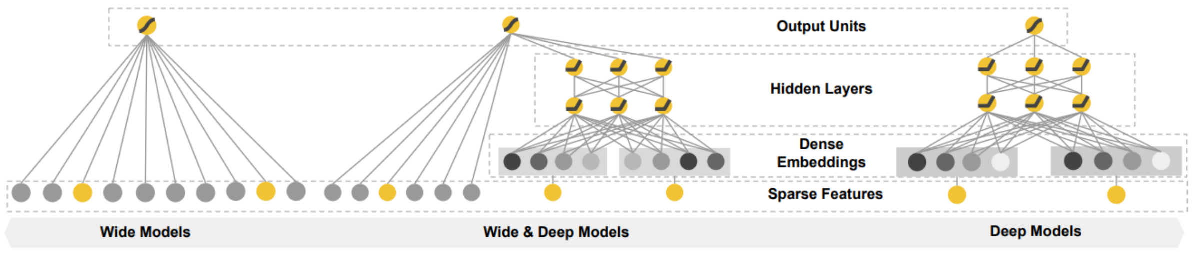


Figure 4: The spectrum of Wide & Deep models[5]

A further improvement over Wide & Deep eliminates the absolute separation of the two components in the early stages. Instead, they use the same set of input raw feature vectors to extract interactions between low- and high-order features. DeepFM[8] leverages the capability of DNN to extract feature representation and learn intricate feature interactions. It consists of two parallel components, an FM component and a deep component. The outputs from two separate modules are concatenated as the final prediction.

$$\hat{y}_{FM}(x) = w_0 + \sum_{i=1}^{N} w_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} v_i^T v_j x_i x_j$$

$$\hat{y}_{DNN}(x) = \sigma(W_L z_{L-1} + b_L) \qquad z_0 = [e_1, e_2, ..., e_f]$$

$$\hat{y} = \sigma(\hat{y}_{FM}(x) + \hat{y}_{DNN}(x))$$

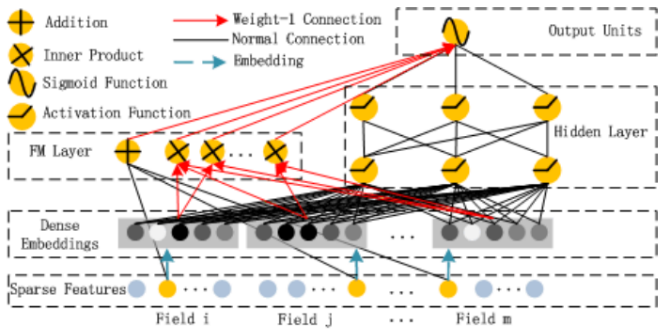


Figure 5: Wide & deep architecture of DeepFM [8]

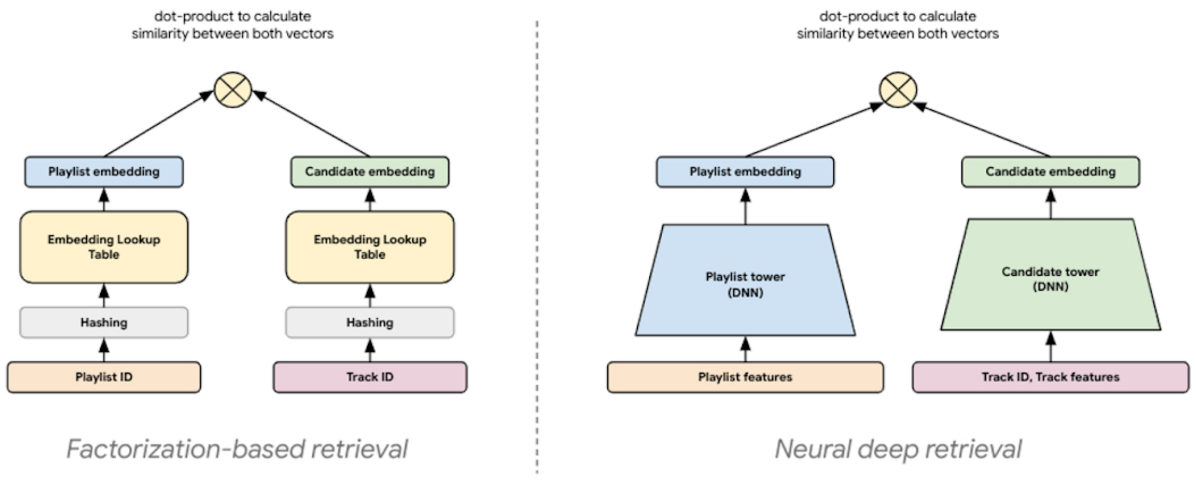


Figure 6: Two tower model for retrieval[12]

2.3 Two-Tower Model

The idea of the Two-Tower model first comes from the Deep Structured Semantic Model[11] which encodes both queries and documents into a similar low-dimensional space and computes relevance as the distance measured in this space. From the illustration Fig 6, we can see that embeddings from user features(queries) and item features(documents) are generated separately from two subsets(towers), and the relevance score is measured as the inner product or cosine similarity between embeddings. There is no direct interaction between user features and item features in this architecture, which increases computational speed at the expense of accuracy.

One of the most famous and valuable industrial examples utilizing such architectures is the Youtube DNN recommendation[6]. In Fig.7, the video watches (video input embeddings) and search tokens are embedded with word2vec[20] and concatenated with user information such as age, gender, and location as the input of a three-layers DNN model. The user embedding is the DNN model's output, and it is utilized to calculate similarities with the video output embeddings. After that, a probability distribution across all candidate videos is obtained by applying Softmax. In order to optimize the training for millions of videos, negative sampling, and importance weighting techniques are used to speed up the computation process. Following the first step of the video candidate generation model, another ranking model performs more fine-grained ranking with abundant and informative features to make more accurate recommendations[25].

2.4 DIN& DEIN

All the methods we discussed above transform input features to a lower dimensional embedding vector and use neural networks to learn nonlinear relations. Next, we look at two different techniques which take the historical behaviors of users into consideration. An ordered, time-stamped list of previous user actions is frequently used as the input for these sequence-ware recommender

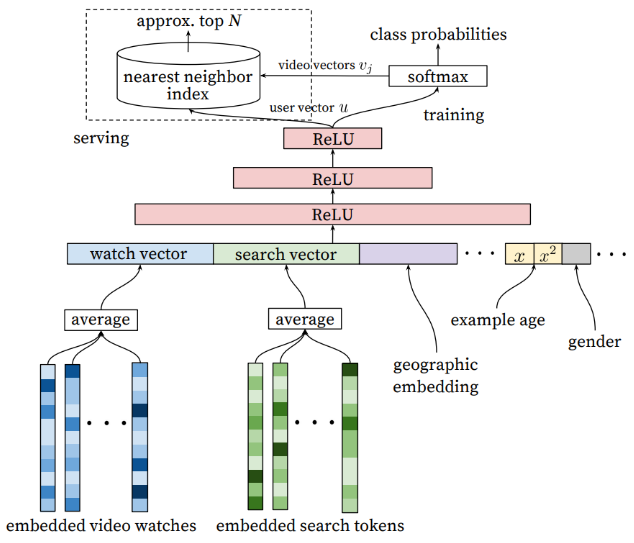


Figure 7: Youtube Candidate Generation Model[6]

systems. DIN[33] was first proposed by Alibaba to optimize their online advertising system, it makes use of users' past browsing history to learn how to represent their interests in relation to specific advertisements in an adaptive manner.. Compared to base models, the DIN model introduced an attention mechanism between users' historical behaviors and candidate advertisements, emphasizing the importance of historical items relevant to current ads. In Fig.8, a local activation unit that contains a feed-forward neural network computes the relevance between behaviors of a user and candidate ads.

$$v_U(A) = f(v_A, e_1, e_2, ..., e_H) = \sum_{j=1}^{H} a(e_j, v_A)e_j = \sum_{j=1}^{H} w_j e_j$$

$\{e_1, e_2, ..., e_H\}$ is the embedding vector of behaviors of user U with a length of H, and v_A is the embedding vector of ad A. It is worth mentioning that the out product between user behaviors and current ads is also included to add explicit knowledge for the modeling.

Further improvements over DIN use more complicated architecture to model the embedding vector of underlying and dynamic user interests. DIEN[32] develops an interest extractor layer followed by an interest involving layer to capture temporal interests and interests evolving process from history behavior sequence, and this system brings 20.7% CTR improvement for the advertising system of Taobao. The interest extractor layer utilizes a GRU module with an auxiliary loss to guide the learning process of user interests by monitoring users' next actions. The embedding vectors of user interests are fed into the Interest Evolving Layer to model the variations and shifts of interests as well as capture the evolving process associated with the target Ad. Finally, the AUGRU(GRU with attentional update gate) is used to combine the outcomes of the attention mechanism and GRU seamlessly.

$$\tilde{u}'_t = a_t * u'_t \qquad h'_t = (1 - \tilde{u}'_t) \circ h'_{t-1} + \tilde{u}'_t \circ \tilde{h}'_t$$

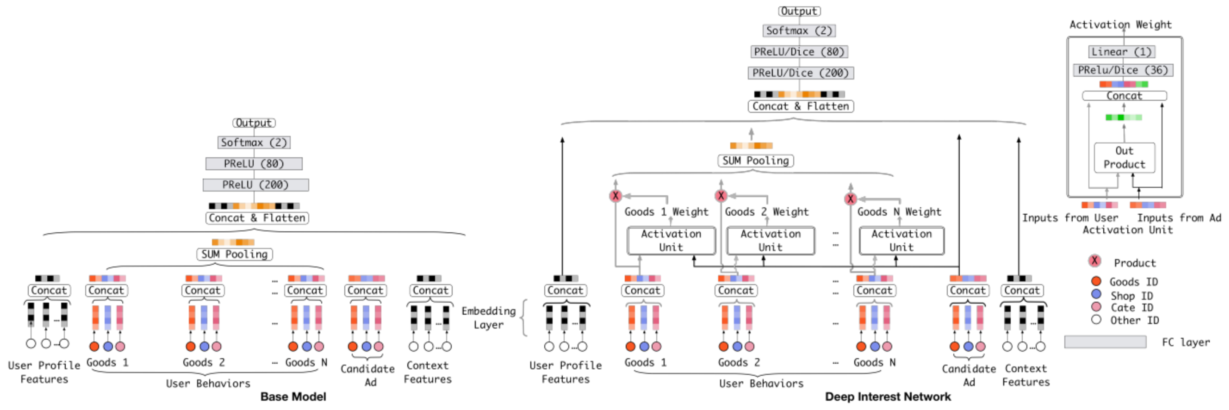


Figure 8: DIN Network Architecture[33]

Less related interest would have less of an impact on the hidden unit in the equation above, which uses the attention score a_t to scale the update gate's dimension. ANGRU pushes the relative interest to develop smoothly and offers more detailed context information.

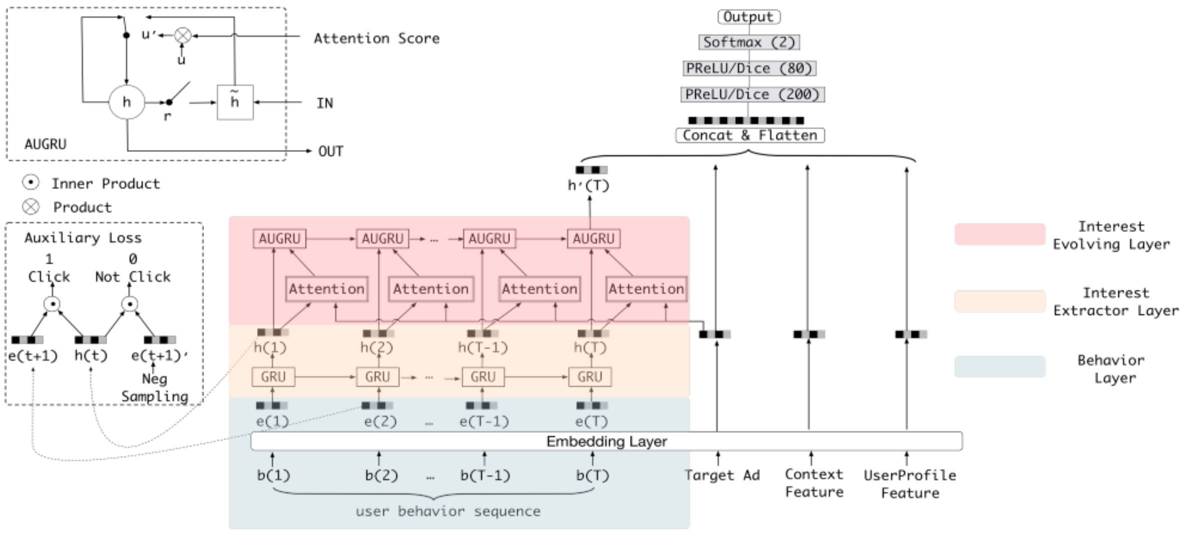


Figure 9: DIEN Network Architecture[32]

2.5 MMOE

Taking it a step further, we review one prominent work in neural-based multi-task learning in the field of movie recommendations. When training neural networks for different purposes simultaneously, balancing the trade-offs between inter-task relationships and task-specific objectives is

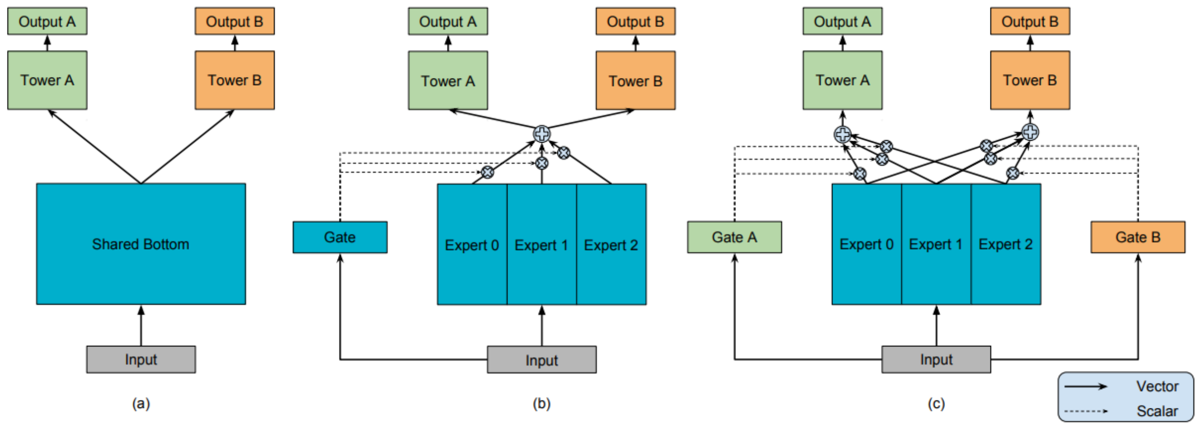


Figure 10: (a) Shared-Bottom Model (b) One-gate MoE model (c) Multi-gate MoE model[18]

essential. The common expert submodels are learned across all tasks using MMoE[18], which uses a mixture of experts (MoE) and optimizes each task using a specialized gating network. The performance and efficiency of this method have been demonstrated by Google's large-scale content recommendation system.

To capture the task heterogeneity without introducing computation overhead, a separate gating network g^k is added for specific task to capture relevant information from each expert.

$$y_k = h^k(f^k(x)) \qquad f^k(x) = \sum_{i=1}^{n} g^k(x)_i f_i(x) \qquad g^k(x) = softmax(W_{gk}x)$$

where $W_{gk} \in \mathbb{R}^{n\times d}$, d is the feature dimension, and n is the number of experts. The idea is insightful and effective, and the flexibility and trainability for multiple tasks are significantly enhanced. The model architecture's prior knowledge, combined with the representations learned by various experts, collaboratively contributes to the final excellent performance.

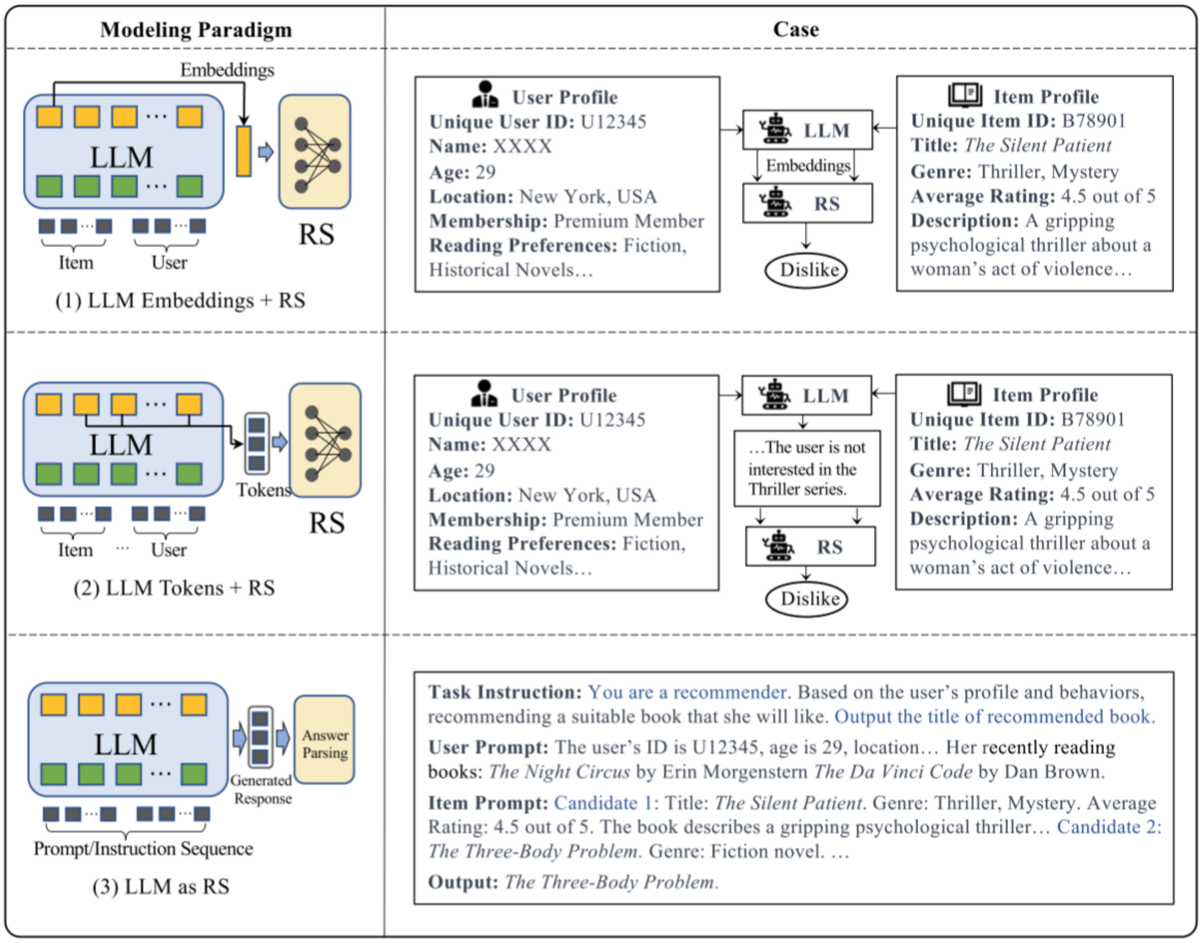
2.6 Summary

In the previous section, we discussed several influential and insightful approaches from both academia and industry. Although most of the work was published before 2019, the architecture and philosophy they proposed continue to influence current research publications, and some of them have been adopted as the backbone of the recommendation architectures in major tech companies. There are many other diverse and promising approaches to model development in this field. Examples include content-based methods[21], collaborative filtering techniques with temporal dynamics[14], and Item2Vec[1]. Additionally, systems have been developed using various advanced techniques, such as Autoencoders[23], graph[7], CNN[34], LSTM[35], Attention[3], DRL[4] and multi-modal tasks[26]. How to better leverage Neural Networks in recommender systems is an ongoing popular research topic in recent decades, innovative techniques and emerging models are published frequently. Limited by the length of the paper, I encourage interested readers to refer to [30] for a more comprehensive overview.

3 LLM for Recommendation

LLMs have received paramount attention in recent years. Even though training large language models from scratch demands significant resources and effort, open-source large language models are continuously being released at an impressive speed. Their capabilities and performance on various benchmarks have steadily enhanced, demonstrating notable progress with each iteration. Naturally, one would ask "Could we harness the powerful capabilities of LLMs to enhance recommendation systems(RS)?". Compared with pattern matching and feature representation extraction in DNN networks, LLMs have superior ability for context understanding and information summarization, and we also aspire they can reason about their recommendations and handle problems such as unseen recommendation scenarios or cold start [31]. Although the deployment of LLMs in recommendation architecture is in its early stages, many researchers have taken the first step to summarize and organize relevant LLM-empowered recommender systems attempts. They aim to provide a systematic, informative, and in-depth overview for researchers and practitioners. [28] suggested that contemporary LLM research on recommendation systems can be summed up in three paradigms and further divided into two primary groups: recommendation generative LLMs and discriminative LLMs. Discriminative LLMs leverage the embedding layers for a range of downstream applications and serve as the backbone for recommendation tasks, aligning with Paradigm (1) illustrated in Fig.11. Conversely, generative LLMs convert recommendation tasks into natural language tasks and produce recommendation results directly by using methods like prompt tuning, instruction tuning, and in-context learning, which is demonstrated by Paradigm (2) and (3) in Fig.11.

Being slightly different from the previous work, [31] thoroughly examines the three main ways that LLMs can improve recommender systems: pre-training, fine-tuning, and prompting. We highlight this work because it is detail-oriented and well-organized. We start by reviewing how LLM can be applied for different recommendation tasks and the development timeline of LLMs and RecSys. From Fig.12, we observe that different queries are designed to support the in-context learning processes of different LLMs. For instance, the possible rating for a specific movie can be inferred by analyzing a user's movie rating history along with the information about the movie being rated. Not only can LLMs assist traditional recommendation tasks such as Top-K recommendation, rating prediction, and conversational recommendation, but multi-modal information such as audio and images can be effectively leveraged during the recommendation process. Moreover, the impressive generalization and deductive abilities of LLMs naturally offer insights and explanations. From the timeline of milestones in RecSys and LLMs domains13, it is intriguing to see that the two domains not only complement but also enhance each other. Traditional N-gram and word2vec models stimulate YoutubeDNN[6] to incorporate negative sampling techniques and treat target and context embeddings differently. There is also a hybrid phase where both PLM-



User Profile
Unique User ID: U12345
Name: XXXX
Age: 29
Location: New York, USA
Membership: Premium Member
Reading Preferences: Fiction, Historical Novels…

Item Profile
Unique Item ID: B78901
Title: *The Silent Patient*
Genre: Thriller, Mystery
Average Rating: 4.5 out of 5
Description: A gripping psychological thriller about a woman's act of violence…

…The user is not interested in the Thriller series.

Task Instruction: You are a recommender. Based on the user's profile and behaviors, recommending a suitable book that she will like. Output the title of recommended book.

User Prompt: The user's ID is U12345, age is 29, location… Her recently reading books: *The Night Circus* by Erin Morgenstern *The Da Vinci Code* by Dan Brown.

Item Prompt: Candidate 1: Title: *The Silent Patient*. Genre: Thriller, Mystery. Average Rating: 4.5 out of 5. The book describes a gripping psychological thriller… Candidate 2: *The Three-Body Problem*. Genre: Fiction novel. …

Output: *The Three-Body Problem*.

Figure 11: Three Paradigms for LLMs on RS[28]

based and DNN-based methods represent the forefront of RecSys research. Stepping into the era of LLMs and Agents, several powerful LLMs are combined with RecSys to deliver significant performance improvement and intelligent personalized suggestions. After getting an overview of the taxonomy and timeline of LLMs empowered recommendation systems, it is worth investigating the similarities and differences between different paradigms and the essential workflow for each paradigm.

3.1 Pre-training LLMs for RecSys

Due to the technological challenges and immense computing resources required to train LLMs from scratch, application developers often finetune pre-trained LLMs using their own data. Masked Language Modeling is used in the training of encoder-only and encoder-decoder Transformer architectures, while Next Token Predicting is adopted for the training of decoder-only architectures. In correspondence with MLM and NTP tasks, PTUM [27] proposes the Masked Behavior Prediction (MBP) and Next K Behavior Prediction (NBP) tasks. The task pairs defined above resemble each other, except that the context is transitioned to recommendation tasks. Fig.14a illustrates the definition of these two recommendation tasks in more depth, we can basically substitute the corpus with datasets in the field of recommendation systems, and training transformer architectures from scratch. If we consider LLMs trained on webpages and Wikipedia as mastering world knowledge of various concepts, then LLMs trained on recommendation corpora have great potential to excel in recommendation tasks.

3.2 Fine-tuning LLMs for RecSys

The two most mainstream methods for fine-tuning are parameter-efficient fine-tuning and full-model fine-tuning. The latter method only makes minor adjustments to the model weights or creates trainable segments for specific tasks. Directly fine-tuned LLMs suffer from the problem of bias, and one possible solution is to introduce masking during training time and testing neutralization to mitigate this problem. Other research efforts have been put into utilization and privacy protection. Regarding parameter-efficient fine-tuning, LoRA[10] has been popular due to its simplicity and efficiency. Taking the immediacy and response time of recommendation tasks into consideration, most papers have focused on effectively fine-tuning 7B or 8B LLMs and minimizing latency for different tasks.

3.3 Prompting LLMs for RecSys

Prompting LLMs for recommendation tasks can be further categorized into three categories according to the function of LLMs: LLMs act as recommenders, Bridge LLMs and RecSys, and LLM-based autonomous agents. From Fig.14c, method one refers to the in-context learning where there is a parameter update of LLMs, we simply outline the task and requirement to retrieve the expected solution. Prompt tuning optimizes the input prompt with trainable prompt tokens to enhance the performance for prompts concentrated on specific tasks. At last, instruction tuning finetunes LLMs with multiple instructions associated with various tasks and extends the applicability to multiple tasks.

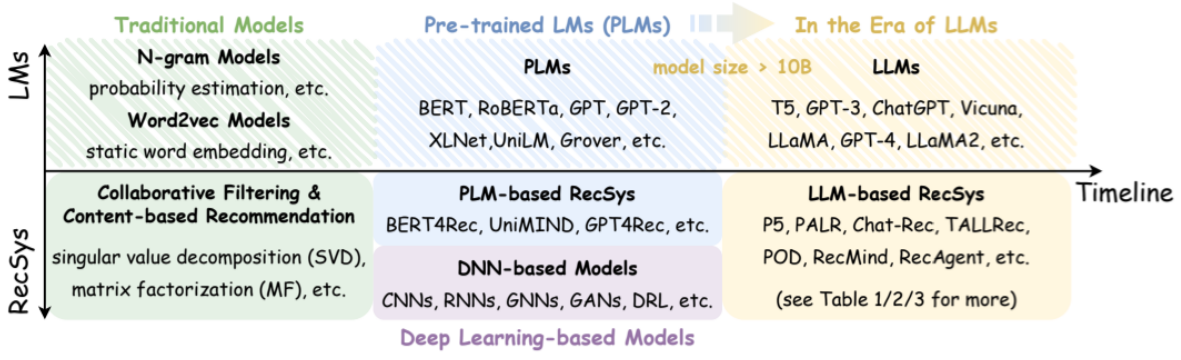

Traditional Models
Pre-trained LMs (PLMs)
In the Era of LLMs
LMs
N-gram Models
probability estimation, etc.
Word2vec Models
static word embedding, etc.
PLMs
model size > 10B
LLMs
BERT, RoBERTa, GPT, GPT-2,
XLNet,UniLM, Grover, etc.
T5, GPT-3, ChatGPT, Vicuna,
LLaMA, GPT-4, LLaMA2, etc.
Timeline
RecSys
Collaborative Filtering &
Content-based Recommendation
singular value decomposition (SVD),
matrix factorization (MF), etc.
PLM-based RecSys
BERT4Rec, UniMIND, GPT4Rec, etc.
DNN-based Models
CNNs, RNNs, GNNs, GANs, DRL, etc.
LLM-based RecSys
P5, PALR, Chat-Rec, TALLRec,
POD, RecMind, RecAgent, etc.
(see Table 1/2/3 for more)
Deep Learning-based Models

Figure 13: Timeline of milestones in RecSys and LLMs[31]

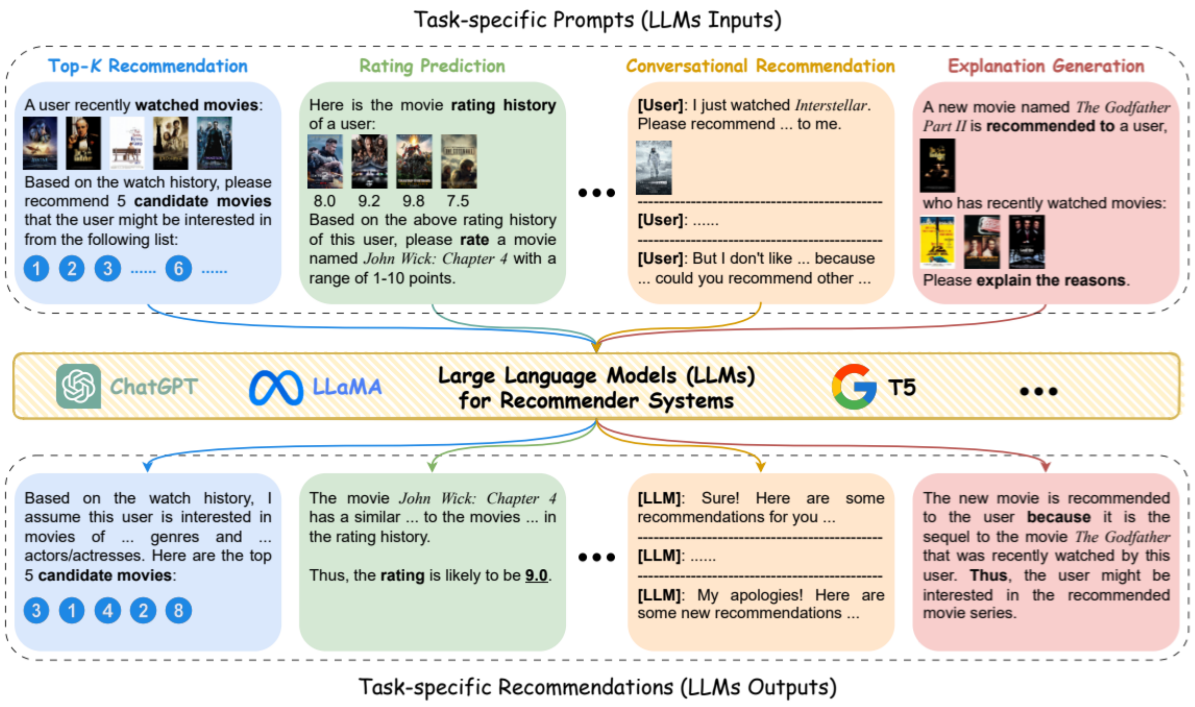
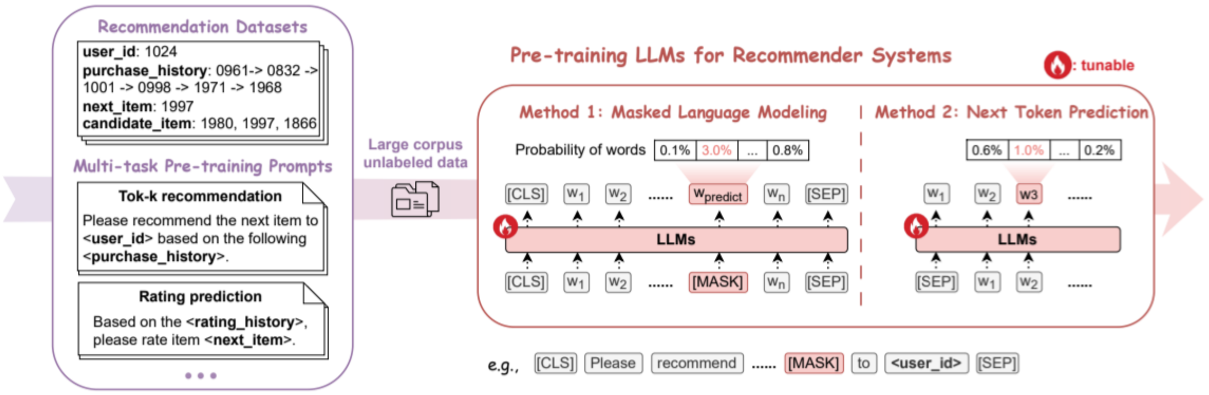
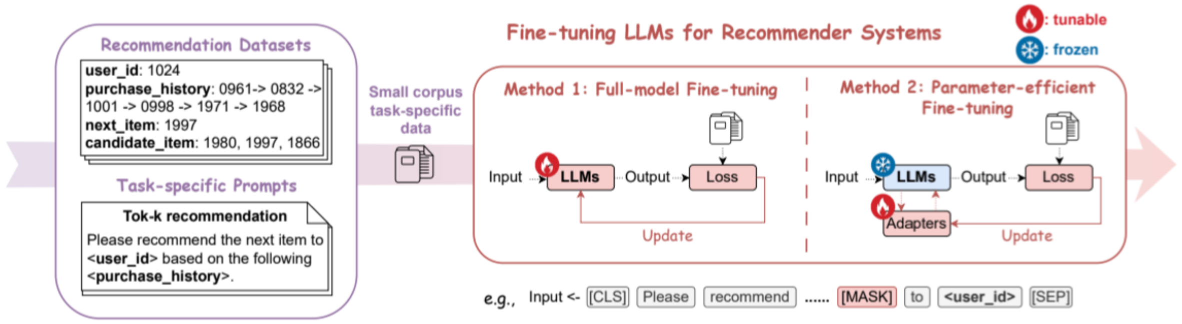

Task-specific Prompts (LLMs Inputs)
Top-K Recommendation
Rating Prediction
Conversational Recommendation
Explanation Generation
A user recently watched movies:
Based on the watch history, please recommend 5 candidate movies that the user might be interested in from the following list:
1 2 3 ...... 6 ......
Here is the movie rating history of a user:
8.0 9.2 9.8 7.5
Based on the above rating history of this user, please rate a movie named John Wick: Chapter 4 with a range of 1-10 points.
[User]: I just watched Interstellar. Please recommend ... to me.
[User]: ......
[User]: But I don't like ... because ... could you recommend other ...
A new movie named The Godfather Part II is recommended to a user,
who has recently watched movies:
Please explain the reasons.
ChatGPT
LLaMA
Large Language Models (LLMs) for Recommender Systems
T5
Based on the watch history, I assume this user is interested in movies of ... genres and ... actors/actresses. Here are the top 5 candidate movies:
3 1 4 2 8
The movie John Wick: Chapter 4 has a similar ... to the movies ... in the rating history.
Thus, the rating is likely to be 9.0.
[LLM]: Sure! Here are some recommendations for you ...
[LLM]: ......
[LLM]: My apologies! Here are some new recommendations ...
The new movie is recommended to the user because it is the sequel to the movie The Godfather that was recently watched by this user. Thus, the user might be interested in the recommended movie series.
Task-specific Recommendations (LLMs Outputs)

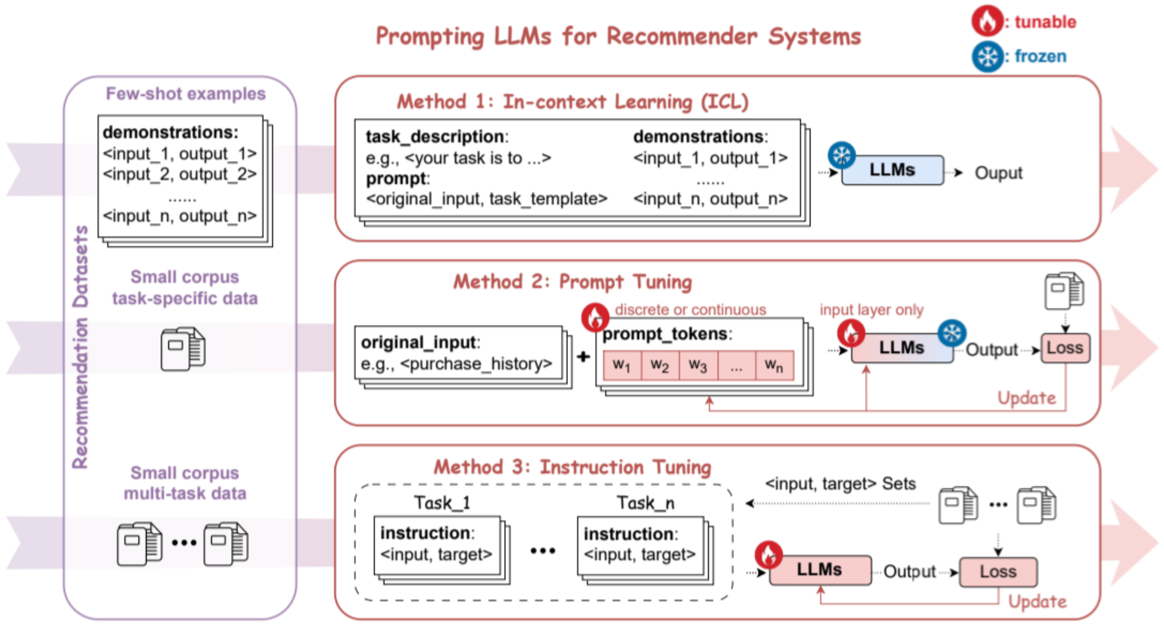
Figure 12: The application of LLMs for different recomendation tasks[31]


Recommendation Datasets
user_id: 1024
purchase_history: 0961-> 0832 -> 1001 -> 0998 -> 1971 -> 1968
next_item: 1997
candidate_item: 1980, 1997, 1866
Multi-task Pre-training Prompts
Tok-k recommendation
Please recommend the next item to <user_id> based on the following <purchase_history>.
Rating prediction
Based on the <rating_history>, please rate item <next_item>.
Large corpus unlabeled data
Pre-training LLMs for Recommender Systems
: tunable
Method 1: Masked Language Modeling
Method 2: Next Token Prediction
Probability of words
0.1% 3.0% ... 0.8%
0.6% 1.0% ... 0.2%
[CLS] w1 w2 ...... wpredict wn [SEP]
w1 w2 w3 ......
LLMs
[CLS] w1 w2 ...... [MASK] wn [SEP]
[SEP] w1 w2 ......
e.g., [CLS] Please recommend ...... [MASK] to <user_id> [SEP]

(a) A workflow of pre-training LLMs for recommender systems


Recommendation Datasets
user_id: 1024
purchase_history: 0961-> 0832 -> 1001 -> 0998 -> 1971 -> 1968
next_item: 1997
candidate_item: 1980, 1997, 1866
Task-specific Prompts
Tok-k recommendation
Please recommend the next item to <user_id> based on the following <purchase_history>.
Small corpus task-specific data
Fine-tuning LLMs for Recommender Systems
: tunable
: frozen
Method 1: Full-model Fine-tuning
Method 2: Parameter-efficient Fine-tuning
Input
LLMs
Output
Loss
Update
Adapters
e.g., Input <- [CLS] Please recommend ...... [MASK] to <user_id> [SEP]

(b) A workflow of fine-tuning LLMs for recommender systems



(c) A workflow of prompting LLMs for recommender systems

Figure 14: Workflow of three paradigms[31]

3.4 summary

Although the impressive generation and reasoning skills of LLMs have attracted plenty of attention and interest in expanding their capability across many areas, they still demand substantial effort and modifications to perform effectively in diverse domains. Once the challenges of adapting LLMs for general recommendation tasks are tackled, high-quality and personalized suggestion services would be the next interesting and compelling research topic. In summary, research outcomes and applications on applying LLMs for RecSys are still in their early stages and it is anticipated to see more practical and groundbreaking ideas and applications emerge in the near future. To get a more comprehensive and systematic review of recently published research papers in this domain, I encourage readers to explore Fig.15 which presents a dissection of research papers addressing the problem of each component in the recommendation cycle.

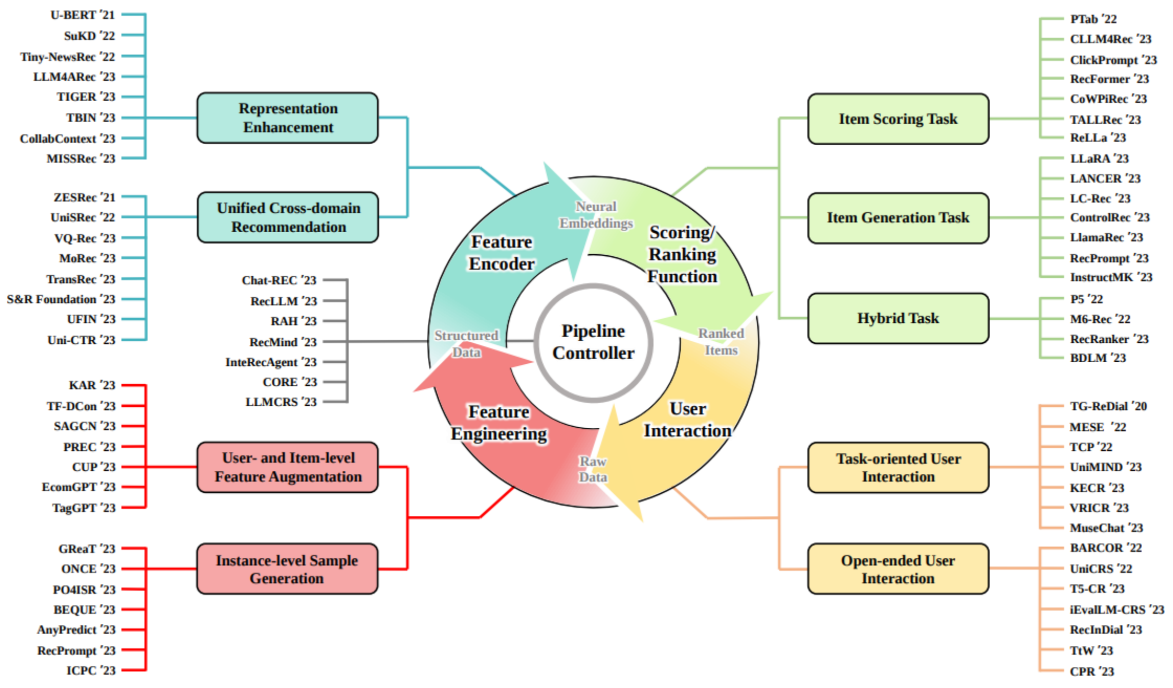


Figure 15: The illustrative dissection of research questions regarding LLMs in RS[16]

4 Outlook

DL-based methods and LLMs techniques for RecSys face diverse challenges to be overcome and aspects to be improved. For DL-based methods, deep composite models generally bring higher accuracy and improve the modeling of various important factors. However, the sensible way to fuse and complement each model to really enhance the overall performance is still worth investigating, and this also demands consideration for real-world applications. Regarding temporal dynamics in recommendation systems, the user history behavior seems to be a time series data, direct sequential modeling breaks the intention of recommendation tasks and introduces redundant and useless information in this process. The relationship between user and item features can evolve together or independently in different time periods which makes it hard to adaptively model and discover the underlying temporal structure. The cross-domain recommendation is common but under-explored in many real-world scenarios, it requires the model to be able to recognize the generalizations and variations across various domains and generate recommendations. For several methods we discussed above, it is worth noting that their efficiency and efficacy are directly examined and ascertained by services provided by large technology companies. The usefulness of a recommendation system is directly linked to its scalability and complexity, incremental learning and inference optimization are the keys to ensuring these properties[30].

The improvement of LLMs for RecSys comes from two aspects, the development of LLMs themselves and better adaptability between LLMs and recommender systems. Well-known hallucination phenomena can have a larger impact on recommendation systems, especially in fields like medical and legal advice. Safety, fairness, and privacy are also crucial factors to be considered when combining LLMs with RecSys. To prevent malicious and inappropriate content outputs, both pre-processing of recommendation task prompts and adversarial training techniques can potentially improve the safety and robustness of LLMs-empowered RecSys. Discrimination inherited from LLMs is exacerbated when putting under the context of recommendations, some items can be promoted or disregarded due to unintended bias. Because customized or personalized suggestion services make extensive use of individual information, this identifiable information leakage becomes a major concern. Both prompt encryption and systems-level protection are studied to guarantee users' privacy, but there still lacks universal and scalable ways to manage users' privacy with LLMs for RecSys. End-to-end RecSys demands low latency and re-calibration efficiency, the effects of tuning for multi-modal RecSys are a promising future direction[31].

Apart from industrial-level recommendation systems benefiting from the rise of DL and LLMs, I notice that many RAG applications or personal assistants based on LLMs are another form of recommendation systems. These systems allow users to customize and manage the use of their personal information and private databases to prompt LLMs for recommendations or information. Although they're in their primitive form and we haven't seen killer applications originating from such workflow, more intelligent and insightful recommendations are always desired. Broadly speaking, we can imagine that home assistants, driving agents, and life agents are all making recommendations to some degree. Not only can recommendation systems increase the revenue for technology companies on their search, advertisement, and item recommendations business, but intelligent agents can be embedded into various aspects of daily lives to make accurate and tailored suggestions and recommendations which consumes a lot of time and energy nowadays. The global digital transformation and web development of twenty years ago made knowledge and information accessible for everyone all over the world, I believe advancements in recommender systems and LLMs over the next twenty years would contribute to the transformation of more intelligent and efficient lifestyles again.

References

[1] Oren Barkan and Noam Koenigstein. “Item2vec: neural item embedding for collaborative filtering”. In: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2016, pp. 1–6.

[2] Ciprian Borodescu. *The anatomy of high-performance recommender systems*. 2023. URL: https://www.algolia.com/blog/ai/the-anatomy-of-high-performance-recommender-systems-part-1/.

[3] Jingyuan Chen et al. “Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention”. In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2017, pp. 335–344.

[4] Xiaocong Chen et al. “Deep reinforcement learning in recommender systems: A survey and new perspectives”. In: *Knowledge-Based Systems* 264 (2023), p. 110335.

[5] Heng-Tze Cheng et al. “Wide & deep learning for recommender systems”. In: *Proceedings of the 1st workshop on deep learning for recommender systems*. 2016, pp. 7–10.

[6] Paul Covington, Jay Adams, and Emre Sargin. “Deep neural networks for youtube recommendations”. In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 191–198.

[7] Chantat Eksombatchai et al. “Pixie: A system for recommending 3+ billion items to 200+ million users in real-time”. In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 1775–1784.

[8] Huifeng Guo et al. “DeepFM: a factorization-machine based neural network for CTR prediction”. In: *arXiv preprint arXiv:1703.04247* (2017).

[9] Xiangnan He et al. “Neural collaborative filtering”. In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[10] Edward J Hu et al. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021).

[11] Po-Sen Huang et al. “Learning deep structured semantic models for web search using clickthrough data”. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 2333–2338.

[12] Jeremy Wortz Jeremy Wortz. *Scaling deep retrieval with TensorFlow Recommenders and Vertex AI Matching Engine*. 2023. URL: https://cloud.google.com/blog/products/ai-machine-learning/scaling-deep-retrieval-tensorflow-two-towers-architecture.

[13] Hyeyoung Ko et al. “A survey of recommendation systems: recommendation models, techniques, and application fields”. In: *Electronics* 11.1 (2022), p. 141.

[14] Yehuda Koren. “Collaborative filtering with temporal dynamics”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 447–456.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

[16] Jianghao Lin et al. “How can recommender systems benefit from large language models: A survey”. In: *arXiv preprint arXiv:2306.05817* (2023).

[17] Weiwen Liu et al. “Neural Re-ranking for Multi-stage Recommender Systems”. In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 698–699.

[18] Jiaqi Ma et al. “Modeling task relationships in multi-task learning with multi-gate mixture-of-experts”. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1930–1939.

[19] Brendan McMahan. “Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 525–533.

[20] Tomas Mikolov. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* 3781 (2013).

[21] MJ Pazzani. *Content-Based Recommendation Systems*. 2007.

[22] Steffen Rendle. “Factorization machines”. In: *2010 IEEE International conference on data mining*. IEEE. 2010, pp. 995–1000.

[23] Suvash Sedhain et al. “Autorec: Autoencoders meet collaborative filtering”. In: *Proceedings of the 24th international conference on World Wide Web*. 2015, pp. 111–112.

[24] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. “Collaborative deep learning for recommender systems”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1235–1244.

[25] Zhe Wang. *Deep Learning Recommender Systems*. Electronics Industry Press, 2020.

[26] Yinwei Wei et al. “MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video”. In: *Proceedings of the 27th ACM international conference on multimedia*. 2019, pp. 1437–1445.

[27] Chuhan Wu et al. “PTUM: Pre-training user model from unlabeled user behaviors via self-supervision”. In: *arXiv preprint arXiv:2010.01494* (2020).

[28] Likang Wu et al. “A survey on large language models for recommendation”. In: *World Wide Web* 27.5 (2024), p. 60.

[29] Aston Zhang et al. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023.

[30] Shuai Zhang et al. “Deep learning based recommender system: A survey and new perspectives”. In: *ACM computing surveys (CSUR)* 52.1 (2019), pp. 1–38.

[31] Zihuai Zhao et al. “Recommender systems in the era of large language models (llms)”. In: *arXiv preprint arXiv:2307.02046* (2023).

[32] Guorui Zhou et al. "Deep interest evolution network for click-through rate prediction". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 5941–5948.

[33] Guorui Zhou et al. "Deep interest network for click-through rate prediction". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1059–1068.

[34] Xiaokang Zhou, Yue Li, and Wei Liang. "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 18.3 (2020), pp. 912–921.

[35] Yuwen Zhou et al. "Personalized learning full-path recommendation model based on LSTM neural networks". In: *Information sciences* 444 (2018), pp. 135–152.