# Recommender Systems

## Lecture 3: Sequential and Large Language Model-based Recommender Systems

David Vos, Chuan Meng
University of Amsterdam

3 June 2025

d.j.a.vos@uva.nl, c.meng@uva.nl

## Where are we?

- **Lecture 1**
  - Introduction to RecSys
- **Lecture 2**
  - Evaluation in RecSys
- **Lecture 3**
  - a. Sequential RecSys
  - b. Large Language Model-based RecSys
- **Lecture 4**
  - Generative RecSys

# Part 1
# Sequential Recommender Systems

## Acknowledgements

This lecture is based on an ECIR 2024 Tutorial on Transformers for Recommender Systems [Petrov and Macdonald, 2024] and a number of published papers.

- **Introduction to Sequential Recommendations**
  - A Brief Recap
  - When does Matrix Factorization fail?
- **Sequential Recommendation Methods**
  - Fully Parametrized Markov Chains [Shani et al., 2005]
  - Factorized Personalized Markov Chains [Rendle et al., 2010]
  - GRU4Rec [Hidasi et al., 2015]
  - SASRec [Kang and McAuley, 2018]
  - BERT4Rec [Sun et al., 2019]
- **Conclusions & Challenges**

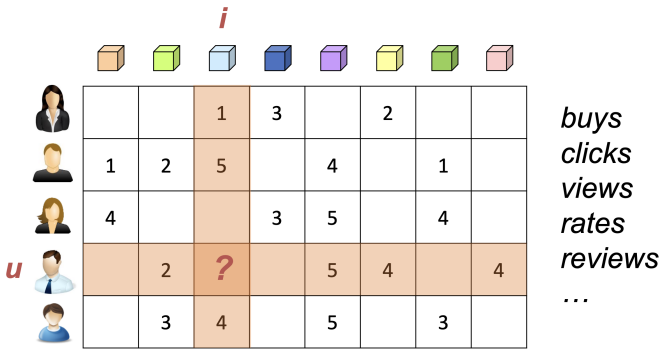
# A Brief Recap

# Recommendations as Matrix Completion

- User-based Collaborative Filtering
  - *How do similar users to the target user like item* $\mathbf{i}$*?*
  - Find neighbours for the target users.
- Item-based Collaborative Filtering
  - *How does the target user like items similar to* $\mathbf{i}$*?*
  - Find neighbours for items the target user liked.
- Item-based Collaborative Filtering is usually preferred as it is more stable.
- We use *Matrix Factorization (MF)* methods to model the recommendation task.
  - The CF matrix is factorized into smaller matrices, allowing for generalization and efficiency during inference.

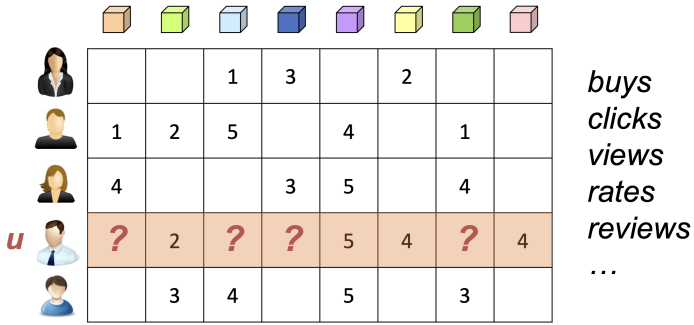
# Recommendations as Matrix Completion

- *User-based Collaborative Filtering*: How do similar users to user **u** like item **i**?



# Recommendations as Matrix Completion

- *Item-based Collaborative Filtering*: How does user **u** like items similar to item **i**?

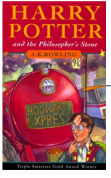


# When does Matrix Factorization Fail?

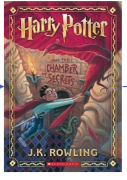
- MF does not consider the order of interactions. In some (real) scenarios, the order may be very important:
  - **Natural Sequence Patterns**
    - Choose a case for the phone after buying the phone, not the other way around.
  - **Series of Items**
    - Star Wars IV $\rightarrow$ Star Wars V $\rightarrow$ Star Wars VI $\rightarrow$ Star Wars I
  - **Evolving User Interests**
    - 10 years ago, a user mostly listened to pop music, but now they prefer rock.
  - **Geographical Proximity**
    - Amsterdam $\rightarrow$ Almere $\rightarrow$ Zwolle $\rightarrow$ Groningen OR
    - Amsterdam $\rightarrow$ Groningen $\rightarrow$ Almere $\rightarrow$ Zwolle
  - **Repeating Interactions**
    - A user buys a pack of coffee every month.

# Sequential Recommendations

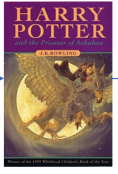
**Goal:** Predict the next item in a chronologically ordered sequence of (historical) user-item interactions.



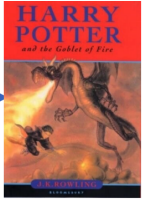

Harry Potter and the Philosopher's Stone




Harry Potter and the Chamber of Secrets




Harry Potter and the Prisoner of Azkaban




**Recommended: Harry Potter and the Goblet of Fire**

# Sequential Recommendation Paradigms

- Instead of items, we can recommend *baskets*, *bundles*, *playlists*, *etc.*
- *Session-based* recommendations and *user-based* recommendations.
- We focus on collaborative signals in this lecture, but item and user representations can be augmented with available content.

# Early works on Sequential Recommendations

# Markov Chains

- Given a sequence of $n$ interactions $< i_1, i_2, \ldots i_n >$, find the probability of the user interacting with the item $i_{n+1}$: $P(i_{n+1}|i_1, \ldots i_n)$
- If the model considers the last $k$ interactions, then it is a k-order Markov Chain: $P(i_{n+1|i_{n-k} \ldots i_n})$.
- Early approaches use an n-gram model, by counting observations in training data Shani et al. [2005].
- Data can become very sparse. This can be alleviated by:
  - Skipping: $< x_1, x_2, x_3 >$ lends some likelihood to $< x_1, x_3 >$
  - Clustering: $< x, y, z > \approx < w, y, z >$
  - Mixture modeling: Combine different values of n

# Next Basket Recommendations with Markov Chains

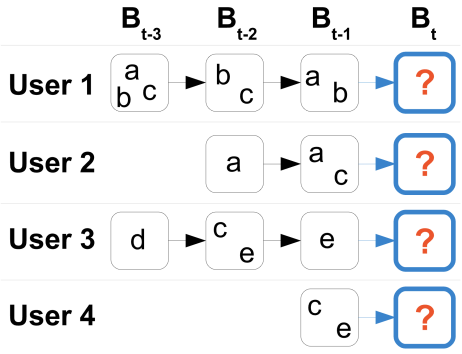
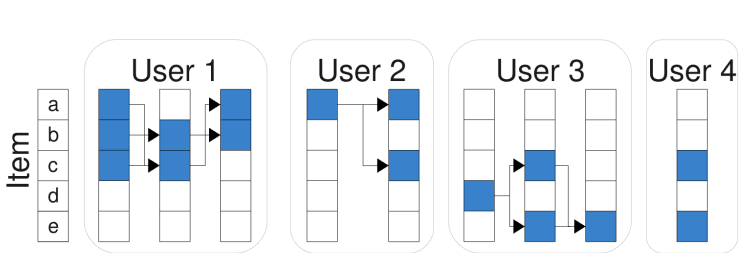



**Figure 1:** Sequential basket data with four users and five items (a, b, c, d, e). The task is to recommend items at time t given a basket history.




Transition Matrix

| from \ to | a | b | c | d | e | # |
|---|---|---|---|---|---|---|
| a | 0.5 | 0.5 | 1 | 0 | 0 | 2 |
| b | 0.5 | 1 | 0.5 | 0 | 0 | 2 |
| c | 0.3 | 0.7 | 0.3 | 0 | 0.3 | 3 |
| d | 0 | 0 | 1 | 0 | 1 | 1 |
| e | 0 | 0 | 0 | 0 | 1 | 1 |

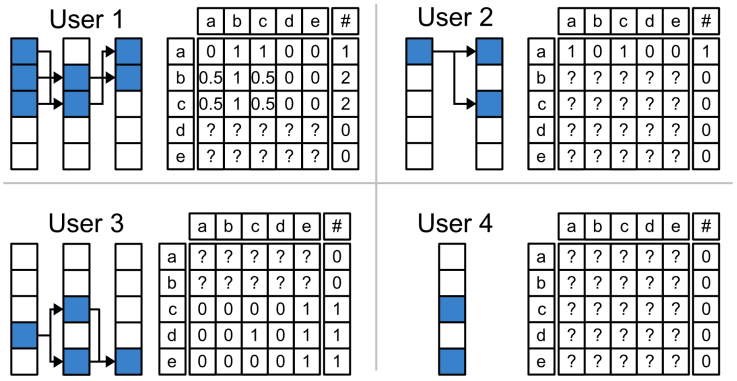
**Figure 2:** The transition matrix contains the estimates for the transition probabilities using the data of users 1, 2, 3 and 4.

# Personalized Markov Chains

- **Limitations of Naive Markov Chains**
  - Use a global transition matrix $T \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$
  - Same transition behavior for all users (no personalization)
- **FPMC: Factorized Personalized Markov Chains** [Rendle et al., 2010]
  - Introduces user-specific transition matrices $T^{(u)} \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$
  - Direct modeling is infeasible due to sparsity $\Rightarrow$ use matrix factorization.
- **Factorize the Transition Cube**
  - Predicted score:

$$\widehat{x}_{u,i,j} = P_u^\top Q_j + R_i^\top S_j$$

  - *Long-term preference:* $P_u^\top Q_j$ (user $u$'s affinity for item $j$)
  - *Short-term transition:* $R_i^\top S_j$ (likelihood of transitioning from item $i$ to $j$)
  - Optimized using a ranking loss with stochastic gradient descent.




| User 1 | a | b | c | d | e | # |
|---|---|---|---|---|---|---|
| a | 0 | 1 | 1 | 0 | 0 | 1 |
| b | 0.5 | 1 | 0.5 | 0 | 0 | 2 |
| c | 0.5 | 1 | 0.5 | 0 | 0 | 2 |
| d | ? | ? | ? | ? | ? | 0 |
| e | ? | ? | ? | ? | ? | 0 |

| User 2 | a | b | c | d | e | # |
|---|---|---|---|---|---|---|
| a | 1 | 0 | 1 | 0 | 0 | 1 |
| b | ? | ? | ? | ? | ? | 0 |
| c | ? | ? | ? | ? | ? | 0 |
| d | ? | ? | ? | ? | ? | 0 |
| e | ? | ? | ? | ? | ? | 0 |

| User 3 | a | b | c | d | e | # |
|---|---|---|---|---|---|---|
| a | ? | ? | ? | ? | ? | 0 |
| b | ? | ? | ? | ? | ? | 0 |
| c | 0 | 0 | 0 | 0 | 1 | 1 |
| d | 0 | 0 | 1 | 0 | 1 | 1 |
| e | 0 | 0 | 0 | 0 | 1 | 1 |

| User 4 | a | b | c | d | e | # |
|---|---|---|---|---|---|---|
| a | ? | ? | ? | ? | ? | 0 |
| b | ? | ? | ? | ? | ? | 0 |
| c | ? | ? | ? | ? | ? | 0 |
| d | ? | ? | ? | ? | ? | 0 |
| e | ? | ? | ? | ? | ? | 0 |

**Figure 3:** Personalized Markov chains: For each user, an individual transition matrix is given.

# FPMC Results

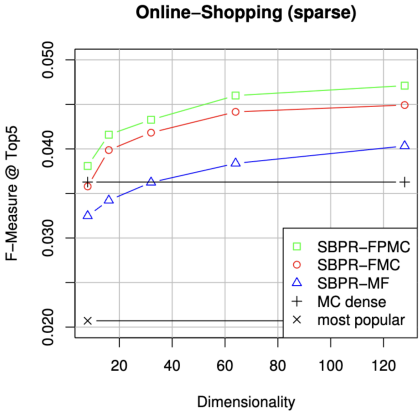



**Figure 4:** FPMC results on an online-shopping dataset. Baselines are Factorized Markov Chains, Matrix Factorization, (fully parametrized) Markov Chains, and a recommender based on item popularity.

# The Era of Deep Learning

# Sequential Recommendations with RNNs

**GRU4Rec**, introduced by Hidasi et al. [2015].

- Among the first deep learning models for sequential recommendations.
- Designed originally for **session-based recommendations**.
- Uses a Gated Recurrent Unit (GRU), a type of Recurrent Neural Network (RNN).

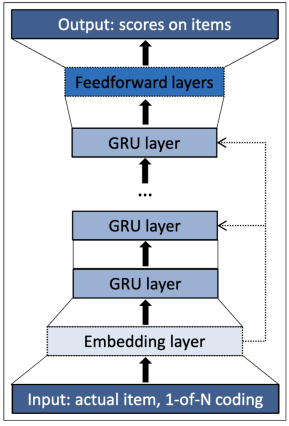



**Figure 5:** GRU4Rec architecture.

# GRU4Rec: Architecture

Overview of the GRU4Rec architecture:

- Each item has its own dense embedding.
- GRU layers capture sequential dependencies from previous interactions.
- Optimized with Bayesian Personalized Ranking (BPR) loss, a pairwise loss between a positive sample and a negative sample.

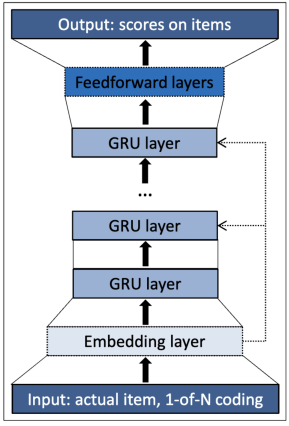



**Figure 6:** GRU4Rec architecture.

# GRU4Rec: When to use it?

- GRU4Rec outperforms methods like FPMC, especially when more data is available.
- GRU4Rec allows for more complex modeling and modeling of longer sequences than FPMC.
- When data is sparse, the model is required to be simple, or computational budget is limited, FPMC can outperform GRU4Rec.

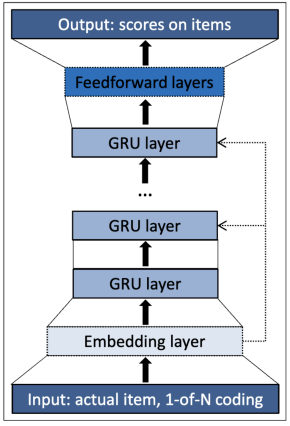



**Figure 7:** GRU4Rec architecture.

# GRU4Rec: Optimization

Optimized with pairwise **Bayesian Personalized Ranking (BPR) loss**:

$$\mathcal{L}_{\mathrm{BPR}} = -\frac{1}{N_S} \sum_{j=1}^{N_S} \log \sigma\left(\hat{r}_{s,i} - \hat{r}_{s,j}\right)$$

- $N_S$ Number of negative samples per positive instance
- $\hat{r}_{s,i}$ Score for the true next item $i$
- $\hat{r}_{s,j}$ Score for negative sample $j$
- $\sigma(\cdot)$ Sigmoid function

**SASRec**, proposed by [Kang and McAuley, 2018].

- First sequential recommender relying solely on self-attention.
- The input embedding is a combination of item embedding with positional embedding.
- Self-attention produces a contextual representation of the sequence.

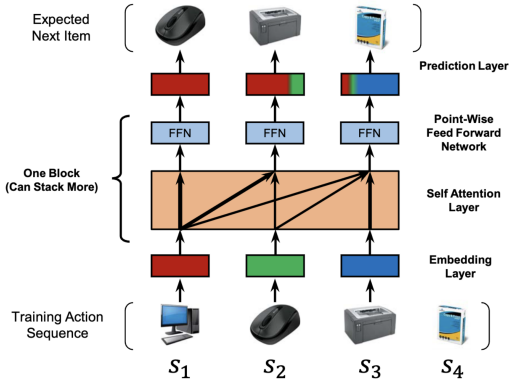



**Figure 8:** SASRec architecture.

# SASRec

- Uses a **causal mask** to prevent attending to future items.
- During training, computes scores for one positive and several negative samples.
- During inference, computes scores for all items by multiplying the last token's representation with the item embedding matrix.
- Optimized with binary cross-entropy (BCE) loss and negative sampling.

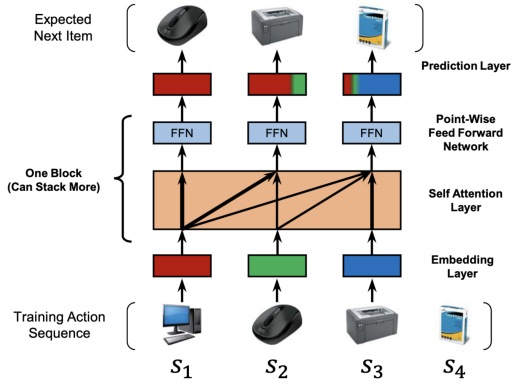



**Figure 9:** SASRec architecture.

# SASRec: Optimization

Optimized with **Binary Cross-Entropy (BCE) loss**:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N_S} \sum_{i=1}^{N_S} [y_{s,i} \log \hat{y}_{s,i} + (1 - y_{s,i}) \log(1 - \hat{y}_{s,i})]$$

$N_S$ Number of samples (positive + negative) per sequence

$y_{s,i} \in \{0, 1\}$ Ground-truth label for sample $i$

$\hat{y}_{s,i}$ Predicted score from SASRec's output

# SASRec Results

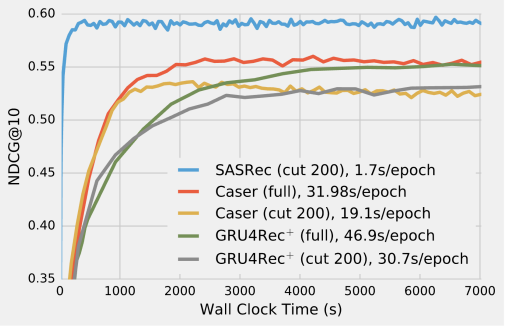



Figure 3: Training efficiency on *ML-1M*. SASRec is an order of magnitude faster than CNN/RNN-based recommendation methods in terms of training time per epoch and in total.

# Sequential Recommendations with Bidirectional Attention

**BERT4Rec**, proposed by Sun et al. [2019].

- Applies bidirectional self-attention to sequential recommendation.
- Causal (unidirectional) attention may miss patterns in loosely ordered data.
- Randomly masks about a certain percentage of items in the input sequence:

$$S = [v_1, \ldots, v_n] \quad \longrightarrow \quad S_{\text{masked}} = [v_1, \ldots, [\text{MASK}], \ldots, v_n]$$

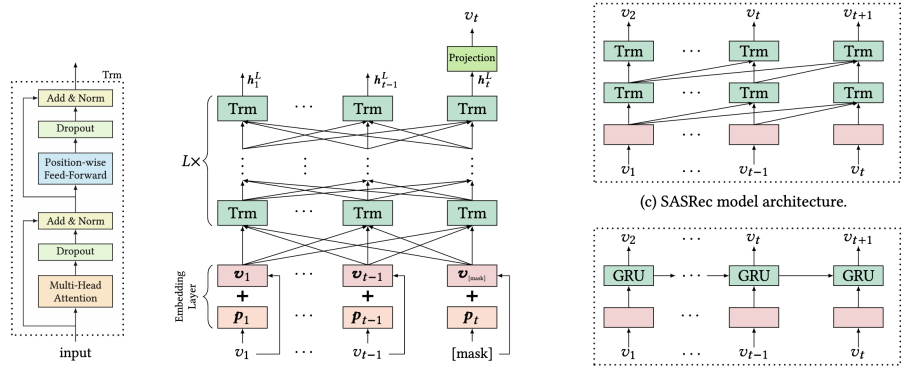
# BERT4Rec

- During inference, a masked token is appended to the user history.

$$S = [v_1, \ldots, v_n] \quad \longrightarrow \quad S_{\text{masked}} = [v_1, \ldots, v_n, [\text{MASK}]]$$

- When performing this 'mask at the end' task as a second training stage, performance of BERT4Rec increases.

# BERT4Rec: Architecture




(a) Transformer Layer. (b) BERT4Rec model architecture. (c) SASRec model architecture. (d) RNN based sequential recommendation methods.

**Figure 1: Differences in sequential recommendation model architectures. BERT4Rec learns a bidirectional model via Cloze task, while SASRec and RNN based methods are all left-to-right unidirectional model which predict next item sequentially.**

**Figure 10:** BERT4Rec architecture.

# BERT4Rec: Optimization

Optimized with **Cross-Entropy (CE) loss** over masked positions:

$$\mathcal{L}_{\text{MLM}} = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \log P(v_i \mid S_{\text{masked}}; \theta)$$

- $\mathcal{M}$ Set of masked positions in the sequence
- $v_i$ True item at position $i$
- $P(v_i \mid \cdot)$ Predicted probability from the Transformer + softmax

# BERT4Rec Results

- BERT4Rec outperforms SASRec. [Sun et al., 2019] claim that this is because of the bidirectional objective.
- Reproducibility studies show that the loss function has a large impact, especially the role of negatives, which can cause overconfidence.

| Dataset | Model | HR@10 | HR@100 | NDCG@10 | NDCG@100 |
|---|---|---|---|---|---|
| ML-1M | BPR-MF | 0.0762 | 0.3656 | 0.0383 | 0.0936 |
| | GRU4Rec (our) | 0.2811 | 0.6359 | 0.1648 | 0.2367 |
| | BERT4Rec | 0.2843 | 0.6680 | 0.1537 | 0.2322 |
| | SASRec | 0.2500 | 0.6492 | 0.1341 | 0.2153 |
| | SASRec+ (Full CE Loss) | 0.3152 | 0.6743 | 0.1821 | 0.2555 |
| | SASRec+ (BCE with 3000 negatives) | **0.3159** | **0.6808** | **0.1857** | **0.2603** |

**Figure 11:** Results reported by [Klenitskiy and Vasilev, 2023].

# Wrapping up

# Methods

In this lecture, we had an introduction to sequential recommenders:

- **Markov Chains**
  - Naive Markov Chains
  - Factorized Personal Markov Chains (FPMC)
- **Deep Learning Methods**
  - GRU4Rec
  - SASRec
  - BERT4Rec

# Discussion: Architectures

| Model | Strengths | Limitations |
|---|---|---|
| **FPMC** | Lightweight and interpretable for simple datasets with short interaction sequences. | Captures only first-order dependencies. |
| **GRU4Rec** | Effective in capturing short temporal patterns within sessions. | Requires a lot of training time, and struggles with long sequences. |
| **SASRec** | Balances model complexity and efficiency, outperforming GRU4Rec. | Does not consider bidirectional context. |
| **BERT4Rec** | Leverages bidirectional context, outperforming SASRec on multiple datasets. | Can be slower to train, performance gains may vary. |

# Discussion: Losses

We discussed **BPR, BCE and CE** losses:

- These losses are not model-specific, and all discussed models can be adapted accordingly.
- There are many alternatives to these losses:
  - TOP1-max, as introduced in the GRU4Rec paper [Hidasi et al., 2015]
  - Listwise losses, such as LambdaRank loss [Li et al., 2021]
  - Contrastive losses, such as InfoNCE [Zhou et al., 2020]

# Discussion

- There's a lot of open challenges in sequential recommendations:
  - Scaling up to catalogs with millions of items
    - Efficient nearest-neighbor algorithms or sub-item IDs.
  - User- and item-cold start scenarios
    - Generative models, like LLMs.
  - Learning from very long user histories
    - Architectural improvements, or smart data pre-processing.
- RecSys is a unique field in its connection to the industry. Challenges often arise based on circumstances and domain-specific requirements.

# Conclusion

Any questions?

Or once you've processed everything: d.j.a.vos@uva.nl

# Part 2

# Large Language Model-based Recommender Systems

# About me

**Chuan Meng**

Final-year PhD student at University of Amsterdam

Supervised by prof. dr. Maarten de Rijke and dr. Mohammad Aliannejadi

Thesis defence scheduled for 19 June 2025

Will join EdinburghNLP at the University of Edinburgh as a postdoctoral researcher

Research Topics: information retrieval (IR) and natural language processing (NLP), with a focus on:

- Proactive conversational search
- Model- and data-efficient neural ranking
- Automatic evaluation methods

As of June 2025, Google Scholar citations 378, H-index 12.

# Acknowledgements

This lecture is based on the paper
*"How Can Recommender Systems Benefit from Large Language Models: A Survey"* [Lin et al., 2025].

# Outline

- **Background:** Recommender pipeline & Large Language Models (LLMs) *[5 min]*
- **Where to Adapt LLMs in Recommender Systems** *[20 min]*
- **How to Adapt LLMs in Recommender Systems** *[10 min]*
- **Challenges** *[5 min]*
- **Conclusions** *[2 min]*

# Background: Recommender system pipeline

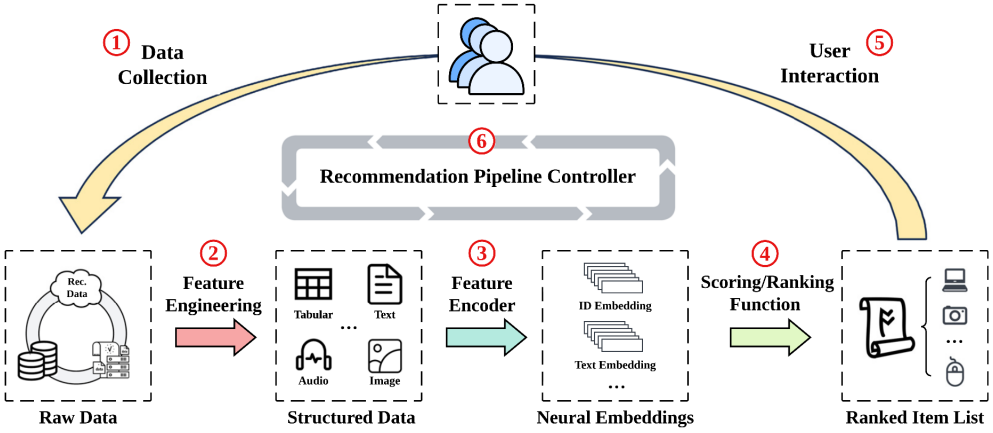



*Figure: An illustration of a typical modern recommender system pipeline.*

# Background: Large Language Models (LLMs)

Which abilities of LLMs can recommender systems potentially benefit from?

- Open-world knowledge
- Reasoning ability
- Generation ability

# Where to Adapt LLMs in Recommender Systems

LLMs can be used at five stages of recommender system pipelines:

- LLMs for feature engineering
- LLMs for feature encoding
- LLMs for ranking
- LLMs for user interaction
- LLMs for pipeline control

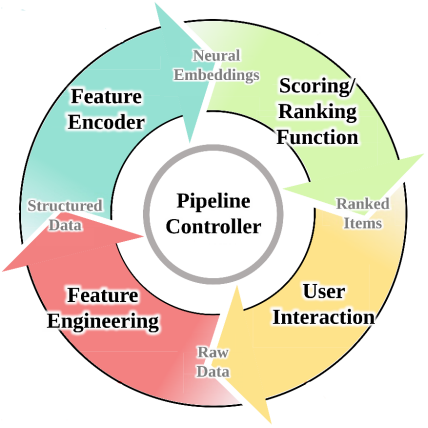



*Figure: Five stages in recommender systems where LLMs can be used.*

# Where to Adapt LLMs in Recommender Systems

LLMs for feature engineering

- LLMs take original features (e.g., item descriptions, user profiles) as input and generate auxiliary textual features for data augmentation, enriching the training data, alleviating long-tail and cold-start problems

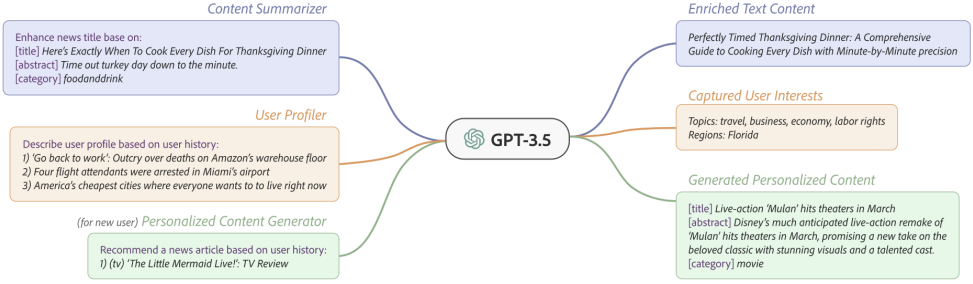



*Figure: LLMs as data augmenters [Liu et al., 2024]*

# Where to Adapt LLMs in Recommender Systems

LLMs for feature engineering

- LLMs take original features (e.g., item descriptions, user profiles) as input and generate auxiliary textual features for data augmentation, enriching the training data, alleviating long-tail and cold-start problems

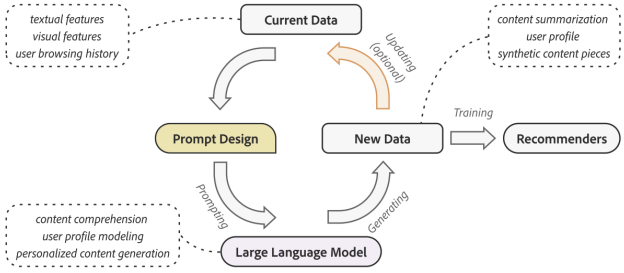



*Figure: data augmentation framework [Liu et al., 2024]*

# Where to Adapt LLMs in Recommender Systems

LLMs for feature encoding

- Enhancing representations with auxiliary textual features: LLMs are used as feature encoders in scenarios where abundant textual features are available

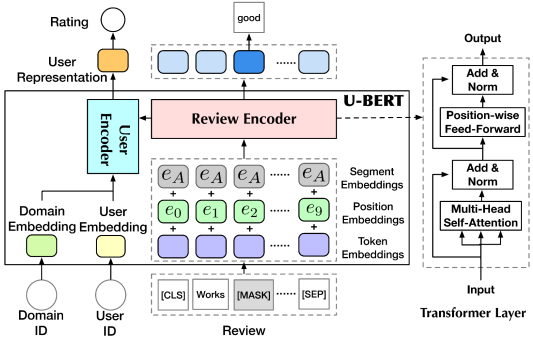



*Figure: Improving user representations by encoding review texts into dense vectors via BERT [Qiu et al., 2021]*

# Where to Adapt LLMs in Recommender Systems

LLMs for feature encoding

- Unified cross-domain recommendation: LLMs enable transfer learning and cross-domain recommendation by using natural language as a bridge across different domains

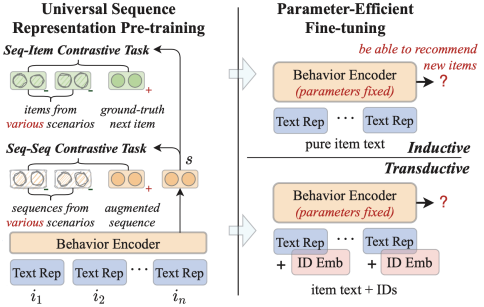



*Figure: Learning universal representations based on [Hou et al., 2022]*

# Where to Adapt LLMs in Recommender Systems

## LLMs for ranking

- Item scoring mode

**Zero-Shot User Rating Predictor**

Given a user's past movie ratings in the format: Title, Genres, Rating.
Ratings range from 1.0 to 5.0.

Babe (1995), Children's|Comedy|Drama, 4
There's Something About Mary (1998), Comedy, 4
Awakenings (1990), Drama, 4
Simple Plan, A (1998), Crime|Thriller, 4
Bug's Life, A (1998), Animation|Children's|Comedy, 5
Twelve Monkeys (1995), Drama|Sci-Fi, 5
Pleasantville (1998), Comedy, 4
Apollo 13 (1995), Drama, 4
Misery (1990), Horror, 4
South Park: Bigger, Longer and Uncut (1999), Animation|Comedy, 4

The candidate movie is Player, The (1992), Comedy|Drama. What's the rating that the user will give?

*Figure: An example of a zero-shot LLM prompt for item scoring [Kang et al., 2023]*

# Where to Adapt LLMs in Recommender Systems

## LLMs for ranking

- Item generation mode

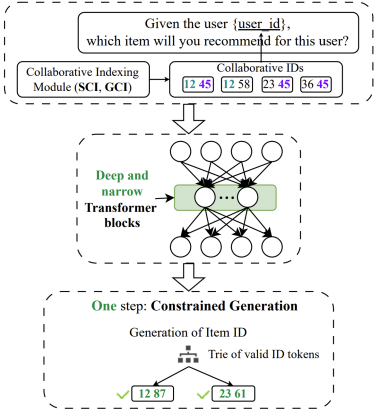



*Figure: open-set (with post-matching) [Mei and Zhang, 2023]*

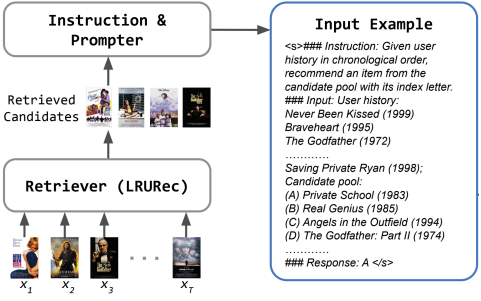



*Figure: closed-set (with pre-filtering) [Yue et al., 2023]*

# Where to Adapt LLMs in Recommender Systems

## LLMs for ranking

- Hybrid mode

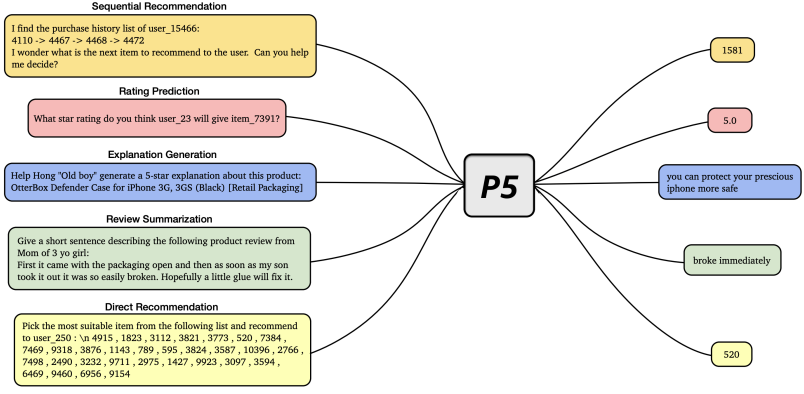



*Figure: A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5) [Geng et al., 2022]*

# Where to Adapt LLMs in Recommender Systems

## LLMs for user interaction: Task-oriented conversational recommender systems

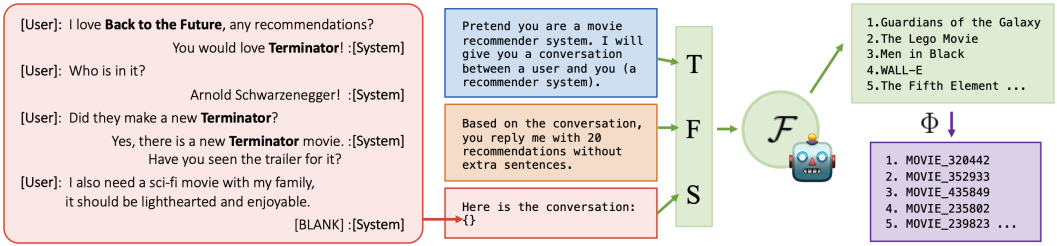



*Figure: an LLM-based task-oriented conversational using a prompting strategy that defines the task description (T), format requirement (F), and conversation context (S) [He et al., 2023]*

# Where to Adapt LLMs in Recommender Systems

LLMs for user interaction: Open-ended conversational recommender systems

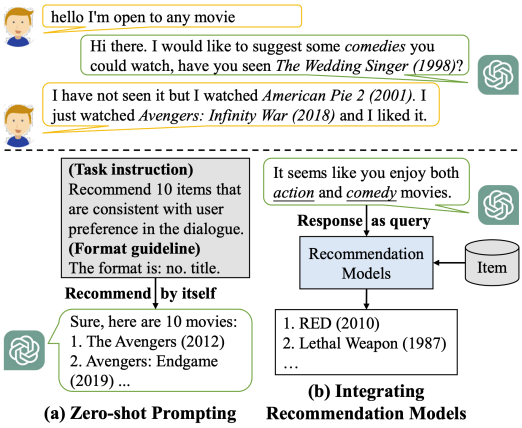



*Figure: An example of an LLM-based open-ended conversational recommender system [Wang et al., 2023]*

# Where to Adapt LLMs in Recommender Systems

## LLMs for pipeline control

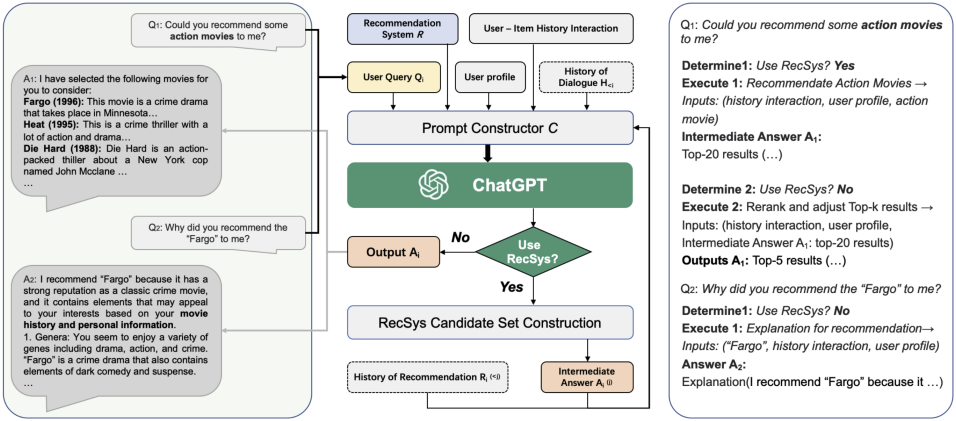



*Figure: An example of an LLM deciding whether or not to call the recommendation API [Gao et al., 2023]*

# How to Adapt LLMs in Recommender Systems

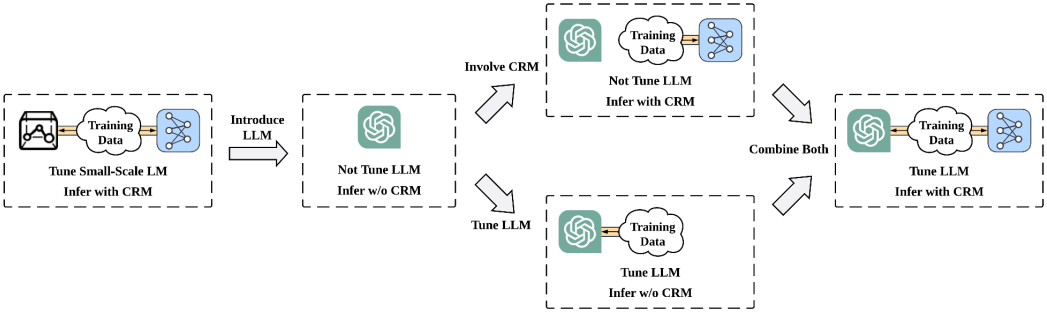



*Figure: Five paradigms of adapting LLMs in recommender systems. CRM stands for conventional recommendation models.*

Tune small-scale LM, infer with CRM (Starting from 2021)

- Earlier attempts generally perform joint optimization of small-scale language models and CRM based on the in-domain training data

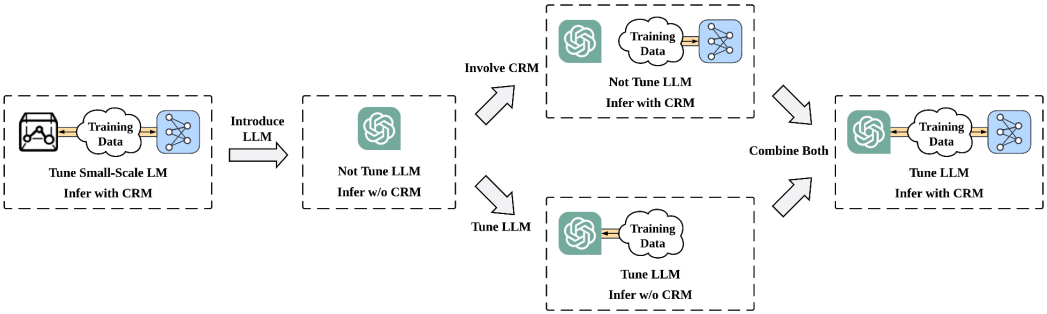



*Figure: Five paradigms of adapting LLMs in recommender systems. CRM stands for conventional recommendation models.*

Not tune LLM, infer w/o CRM (Starting from 2023)

- introduce a frozen LLM for recommendation without CRM
- inferior performance because **LLMs lack in-domain collaborative knowledge**

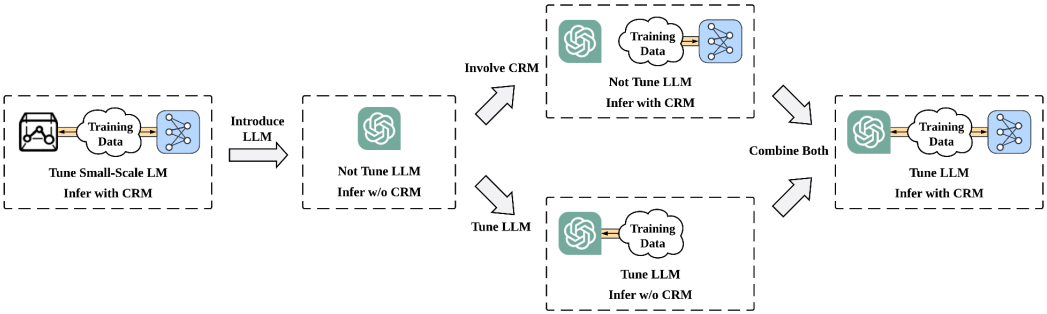



*Figure: Five paradigms of adapting LLMs in recommender systems. CRM stands for conventional recommendation models.*

## Not tune LLM, infer with CRM

- incorporating in-domain collaborative knowledge by using CRM

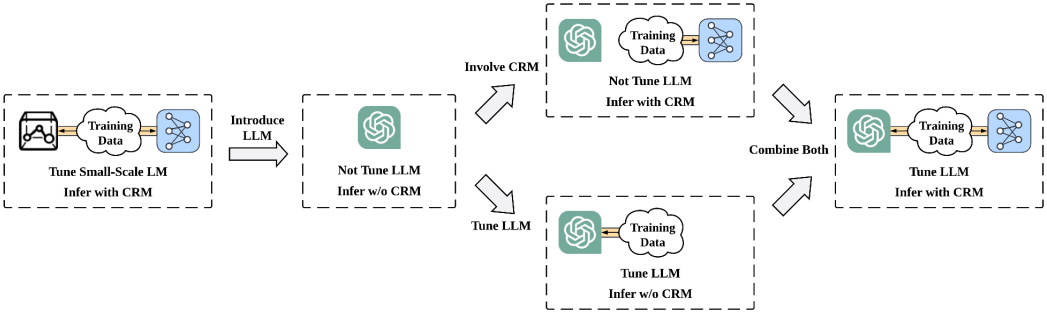



*Figure: Five paradigms of adapting LLMs in recommender systems. CRM stands for conventional recommendation models.*

Tune LLM, infer w/o CRM

- Incorporating in-domain collaborative knowledge by fine-tuning LLMs on in-domain data
- Parameter-efficient fine-tuning (PEFT) methods, like LoRA [Hu et al., 2022].

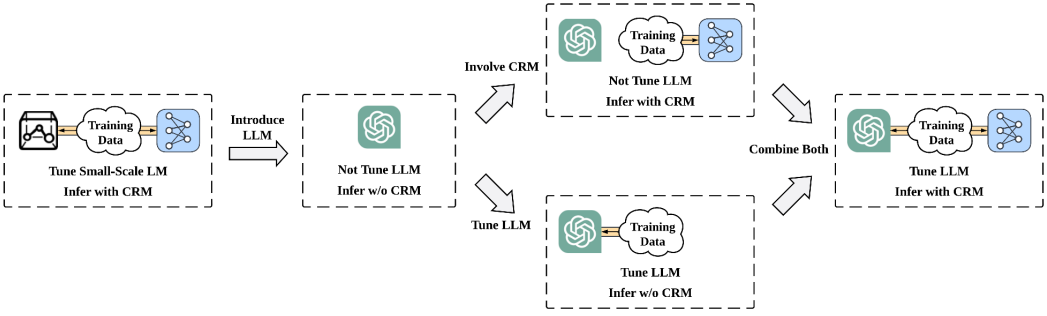



*Figure: Five paradigms of adapting LLMs in recommender systems.*

## Tune LLM, infer with CRM

- Incorporating in-domain collaborative knowledge by using the both strategies

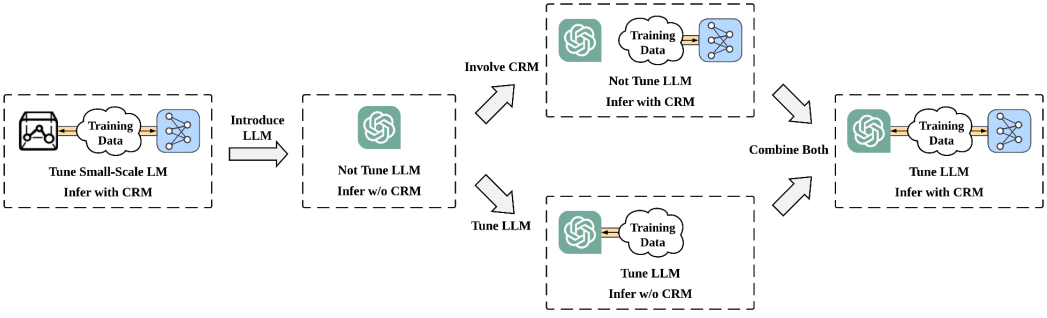



*Figure: Five paradigms of adapting LLMs in recommender systems. CRM stands for conventional recommendation models.*

# How to Adapt LLMs in Recommender Systems

Key for strong performance: bring in in-domain collaborative knowledge

- Infer with CRM during the inference phase (model-side solution)
- Fine-tuning LLMs during the training phase (data-side solution)

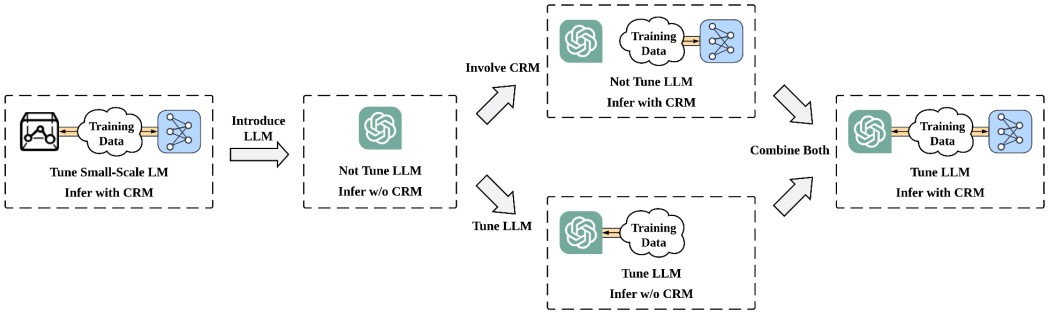



*Figure: Five paradigms of adapting LLMs in recommender systems. CRM stands for conventional recommendation models.*

# Challenges

Training efficiency

- Frequent LLM tuning is costly in time and compute
- Potential solutions:
  - Use PEFT methods
  - Reduce LLM training data and update frequency, while keeping full data and frequent updates for CRM

# Challenges

Inference efficiency

- LLMs have high inference latency due to their billions of parameters
- Potential solutions:
  - Pre-compute and store intermediate results to reduce runtime at the cost of extra memory (e.g., [Cui et al., 2022])
  - Distillation (e.g., [Cui et al., 2024])
  - Pruning (e.g., [Behdin et al., 2025])
  - Quantization (e.g., [Behdin et al., 2025])

# Challenges

Long text modeling

- LLMs struggle with long inputs (e.g., long user history or large candidate sets), even if the total tokens are within the context window
- Potential solutions:
  - Pre-filter input text before feeding into LLMs
  - Use CRM's latent representations to compress LLMs' input (e.g., [Zhang et al., 2025, Li et al., 2023])

# Challenges

Fairness

- LLMs have shown a lack of user-side and item-side fairness
- Potential solutions:
  - For user-side fairness, counterfactually fair prompting (e.g., [Hua et al., 2024])
  - For item-side fairness, special treatment for representing long-tail items (e.g., [Liu et al., 2023])

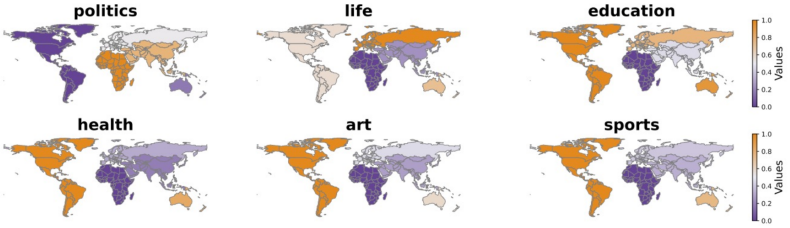



*Figure: An example of user-side unfairness, where LLMs recommend news categories based on users with regionally popular names [Xu et al., 2024]*

# Challenges

Privacy

- LLMs pre-trained, fine-tuned, or inferred on user-sensitive data may risk exposing private information and violating user privacy
- Potential solution:
  - Unlearning, i.e., removing some personal data (e.g., historical behaviors) from established LLM-based recommender systems (e.g., [Hu et al., 2024, Zhang et al., 2024, Pawelczyk et al., 2024])

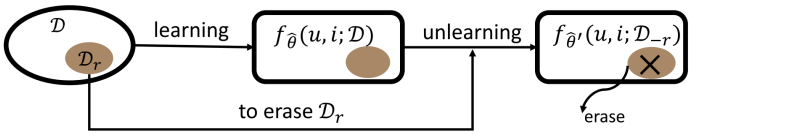



*Figure: Illustration of recommendation unlearning process [Zhang et al., 2024]*

# Challenges

## Explainability

- Explaining why an item is recommended helps build user trust
- Potential solution:
  - Fine-tuning LLMs for recommendation explanation (e.g., [Lei et al., 2024])

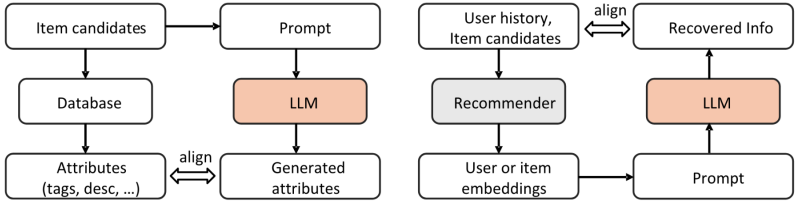



*Figure: Fine-tuning LLMs on different tasks related to recommendation explanation, such as Item discrimination and history reconstruction [Lei et al., 2024]*

# Conclusions

## Summary

- Where and how to adapt LLMs in recommender systems
- Challenges

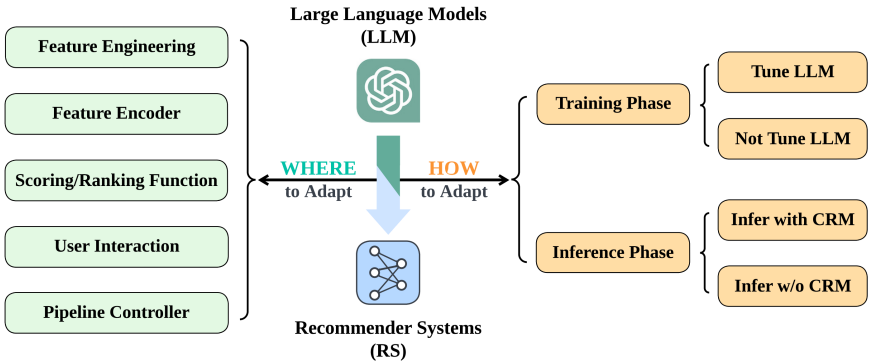



*Figure: Summary of where and how*

# References i


K. Behdin, Y. Dai, G. Dexter, A. Gupta, R. Mazumder, A. Saha, Q. Song, S. Tang, S. Zhu, and P.-L. Hsu. Efficient algorithms for leveraging llms for generative and predictive recommender systems. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 1–4, 2025.

Y. Cui, F. Liu, P. Wang, B. Wang, H. Tang, Y. Wan, J. Wang, and J. Chen. Distillation matters: Empowering sequential recommenders to match the performance of large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 507–517, 2024.

Z. Cui, J. Ma, C. Zhou, J. Zhou, and H. Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.

Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*, 2023.

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

Z. He, Z. Xie, R. Jha, H. Steck, D. Liang, Y. Feng, B. P. Majumder, N. Kallus, and J. 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.

B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv [cs.LG]*, Nov. 2015.

Y. Hou, S. Mu, W. X. Zhao, Y. Li, B. Ding, and J.-R. Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 585–593, 2022.

E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Z. Hu, Y. Zhang, M. Xiao, W. Wang, F. Feng, and X. He. Exact and efficient unlearning for large language model-based recommendation. *arXiv preprint arXiv:2404.10327*, 2024.

# References iii


W. Hua, Y. Ge, S. Xu, J. Ji, Z. Li, and Y. Zhang. Up5: Unbiased foundation model for fairness-aware recommendation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1899–1912, 2024.

W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, Nov. 2018.

W.-C. Kang, J. Ni, N. Mehta, M. Sathiamoorthy, L. Hong, E. Chi, and D. Z. Cheng. Do llms understand user preferences? evaluating llms on user rating prediction. *arXiv preprint arXiv:2305.06474*, 2023.

A. Klenitskiy and A. Vasilev. Turning dross into gold loss: is BERT4Rec really better than SASRec? In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1120–1125, New York, NY, USA, Sept. 2023. ACM.

Y. Lei, J. Lian, J. Yao, X. Huang, D. Lian, and X. Xie. Recexplainer: Aligning large language models for explaining recommendation models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1530–1541, 2024.

R. Z. Li, J. Urbano, and A. Hanjalic. New insights into metric optimization for ranking-based recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, July 2021. ACM.

X. Li, C. Chen, X. Zhao, Y. Zhang, and C. Xing. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443*, 2023.

J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, H. Zhang, Y. Liu, C. Wu, X. Li, C. Zhu, et al. How can recommender systems benefit from large language models: A survey. *ACM Transactions on Information Systems*, 43(2):1–47, 2025.

Q. Liu, N. Chen, T. Sakai, and X.-M. Wu. Once: Boosting content-based recommendation with both open- and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 452–461, 2024.

Z. Liu, S. Mei, C. Xiong, X. Li, S. Yu, Z. Liu, Y. Gu, and G. Yu. Text matching improves sequential recommendation by reducing popularity biases. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1534–1544, 2023.

## References v


K. Mei and Y. Zhang. LightIm: A lightweight deep and narrow language model for generative recommendation. *arXiv preprint arXiv:2310.17488*, 2023.

M. Pawelczyk, S. Neel, and H. Lakkaraju. In-context unlearning: Language models as few shot unlearners. In *Proceedings of the 41st International Conference on Machine Learning*, pages 40034–40050, 2024.

A. V. Petrov and C. Macdonald. Transformers for sequential recommendation. In *Proceedings of the 46th European Conference on Information Retrieval*, pages 369–374, 2024.

Z. Qiu, X. Wu, J. Gao, and W. Fan. U-bert: Pre-training user representations for improved recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4320–4327, 2021.

S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, New York, NY, USA, Apr. 2010. ACM.

F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

G. Shani, D. Heckerman, and R. I. Brafman. An mdp-based recommender system. *Journal of machine Learning research*, 6(Sep):1265–1295, 2005.

F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, Nov. 2019. ACM.

X. Wang, X. Tang, W. X. Zhao, J. Wang, and J.-R. Wen. Rethinking the evaluation for conversational recommendation in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10052–10065, 2023.

C. Xu, W. Wang, Y. Li, L. Pang, J. Xu, and T.-S. Chua. A study of implicit ranking unfairness in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7957–7970, 2024.

Z. Yue, S. Rabhi, G. d. S. P. Moreira, D. Wang, and E. Oldridge. Llamarec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089*, 2023.

Y. Zhang, Z. Hu, Y. Bai, J. Wu, Q. Wang, and F. Feng. Recommendation unlearning via influence function. *ACM Transactions on Recommender Systems*, 3(2):1–23, 2024.

Y. Zhang, F. Feng, J. Zhang, K. Bao, Q. Wang, and X. He. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.