The Symbiosis of LLMs and RecSys: from LLM for RecSys to RecSys for LLM

Yanjie Fu

Yanjie.fu@asu.edu

School of Computing and AI

Arizona State University

About Me

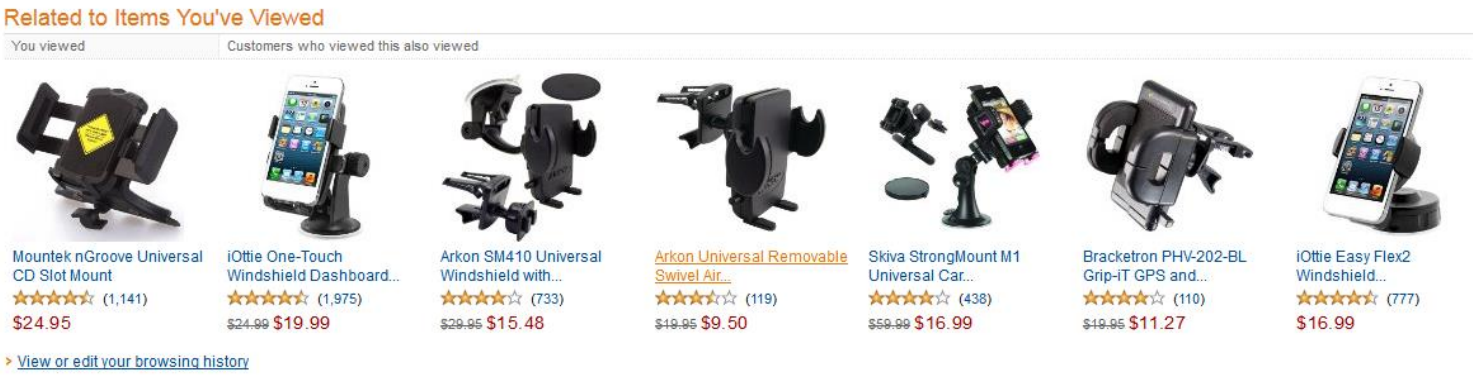
- **Associate Professor, School of Computing and AI, Fulton Schools of Engineering, Arizona State University**
- **Ph.D. from Rutgers University, NJ, USA**
- **B.E. from University of Science and Technology of China (USTC), Hefei, China**
- **Research and Teaching**
  - Selected awards: 2023 US NAE FOE Early Career Engineer, 2021 US NSF CAREER, 2018 NSF CISE CRII
  - 5 Best Paper (Runner-up, Finalist) Awards: KAIS Best of IEEE ICDM 2022, KAIS Best of IEEE ICDM 2021, ACM TSAS Best of SIGSpatial2020, ACM TKDD Best of SIGKDD2018, KAIS Best of IEEE ICDM2014

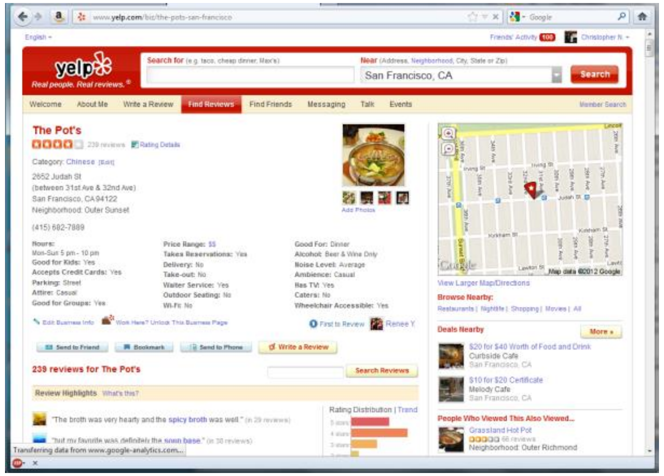
- Dissertation Ph.D. Students
  - Pengyang Wang (Tenure-track Assistant Professor at Univ of Macau, graduated in 2021)
  - Kunpeng Liu (Tenure-track Assistant Professor at Portland State University, graduated in 2022)
  - Wei Fan (Postdoc at University of Oxford UK, graduated in 2023)
  - Dongjie Wang (Tenure-track Assistant Professor at Univ of Kansas, graduated in 2024)
- Joint Exchange Program Ph.D. Students
  - Meng Xiao, an assistant professor at Chinese Academy of Sciences
  - Ziyue Qiao, an assistant professor at the Great Bay University
  - Lu Jiang, an assistant professor at Dalian Maritime University
  - Pengfei Wang, an associate professor at Chinese Academy of Sciences
  - Peijie Sun, an associate professor at Nanjing University of Posts and Telecommunications

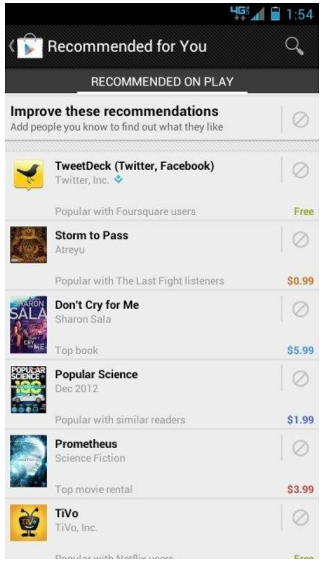
Recommender Systems (RecSys)

- Recommender systems: addressing information overload
  - Learning to recommend interesting items to users
- Formulations of RecSys
  - RecSys as rating prediction: predict the ratings of users for items
  - RecSys as matrix completion: complete missing entries of a user-item rating matrix
  - RecSys as link prediction: infer the link between a user and an item
  - RecSys as ranking inference: rank items for users
  - RecSys as similarity computing: compute the similarities between users and items to recommend based on similarities
  - RecSys as matching/pairing problems: match users with items
  - RecSys as allocation problems: allocate/assign items to users
  - RecSys as optimization problems: minimize/maximize the loss/likelihood between ground-truth ratings and predicted ratings

Sample RecSys





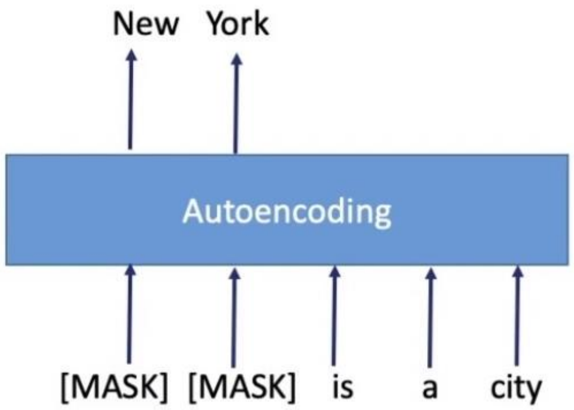


Large Language Models (LLM)

- LLM are advanced AI designed to **understand**, **generate**, and **interact with** human languages
- The interpretation of “LARGE”
  - Large training data, large parameter sizes, large computing power, large downstream tasks (e.g., text understanding, generation, summarization, translation, sentiment, classification, entity recognition, text-to-X/X-to-text generation, ChatBots, QA)
- X as LLM
  - The generation capability of LLM allows us to reformulate any AI task as LLM, including RecSys as LLM

Basic Language Models

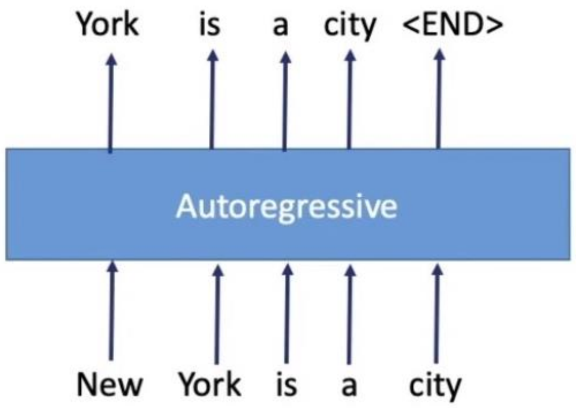
Auto-Encoding Language Models



- Predict masked tokens using context information.

$$p(x) = \sum_{t=1}^{N} mask_t \log p(t_k|\boldsymbol{Content})$$

Auto-Regressive Language Models



- Predict the current token based on the tokens that appear before (or after) it.

$$p(x) = \sum_{k=1}^{N} \log p\left(t_k \middle| t_1, \dots, t_{k-1}\right)$$

Sample LLMs

- Technical Path
  - Transformer
  - BERT (encoder only)
  - GPT (decoder only)
  - BART (encoder-decoder)
  - Domain-specific LLM
  - Multi-modal LLM

- Selected LLM Tools
  - Llama: open source, tunable (https://www.llama.com/)
  - ChatGPT/o1: strong power, costly (https://openai.com/)
  - Claude: good at coding and reasoning (https://www.anthropic.com/claude)

LLM’s One-Sided Contributions to RecSys

- LLM for RecSys
  - Reformulating RecSys tasks into LLM token generation (direct item recommendation, rating prediction, sequential recommendation, explainable recommendation)
  - Generalize pre-training and fine-tuning to fine-tune RecSys on specific recommendation datasets/tasks
  - Leverage LLM’s comprehension ability to profile user, item, and contexts
  - Leverage LLM’s interaction ability to provide continuous refinement with user feedback
  - Leverage LLM’s textual generation to provide explainable recommendation
  - LLM as few-shot or zero-shot recommenders

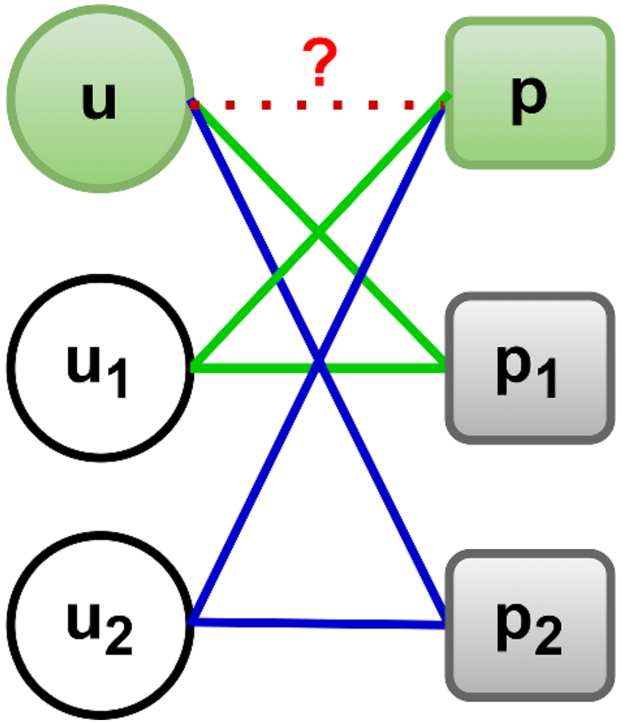
Reciprocal Benefits Between LLM and RecSys

- RecSys for LLM
  - User modeling: can inspire us to develop personalized LLM
  - Decision and choice modeling: can inspire us to recommend the most appropriate LLM to a query

LLM for RecSys: Graph Knowledge Structured LLM as RecSys

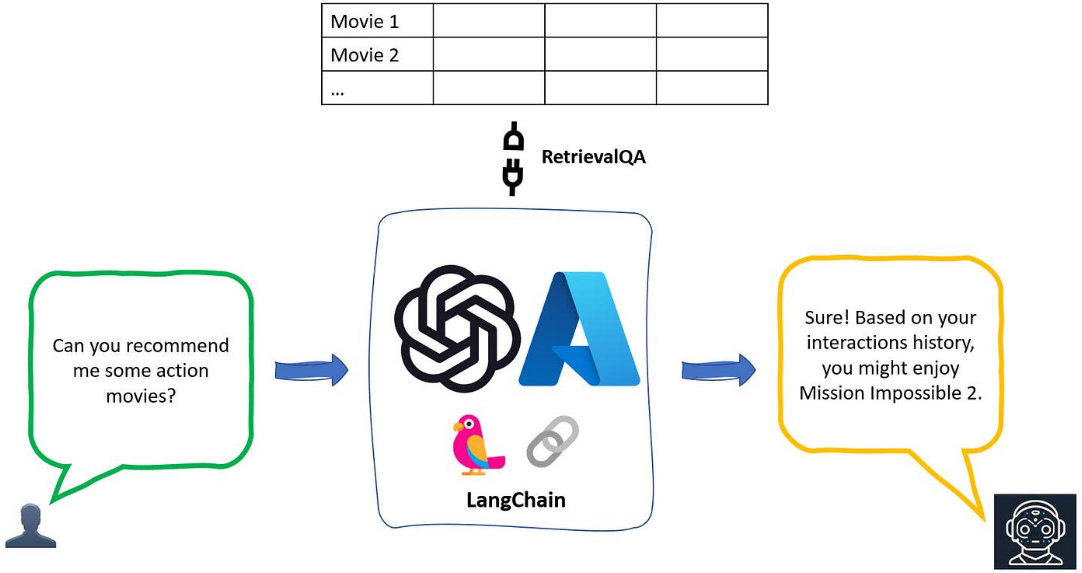
A Graph-based View of RecSys

- See user-item interactions or user-item rating matrix as graphs
- Leverage graph structure and topology to enhance RecSys

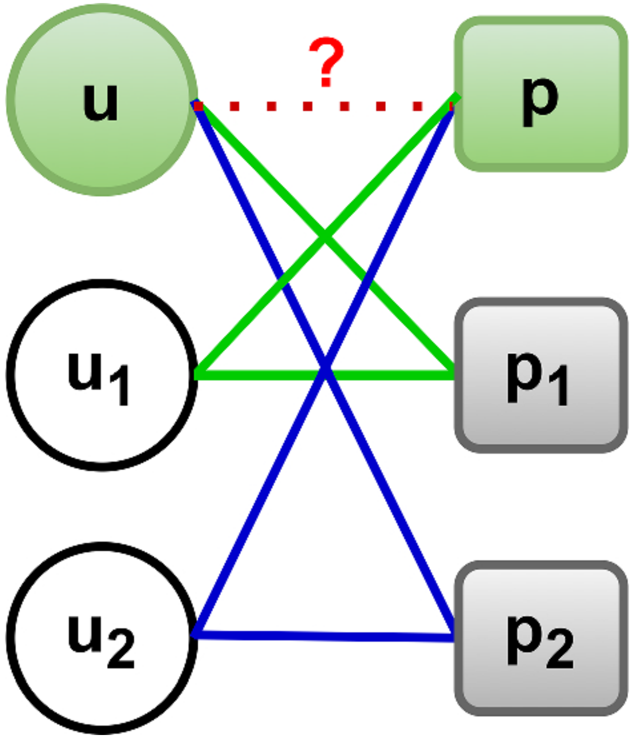


A LLM-based View of RecSys

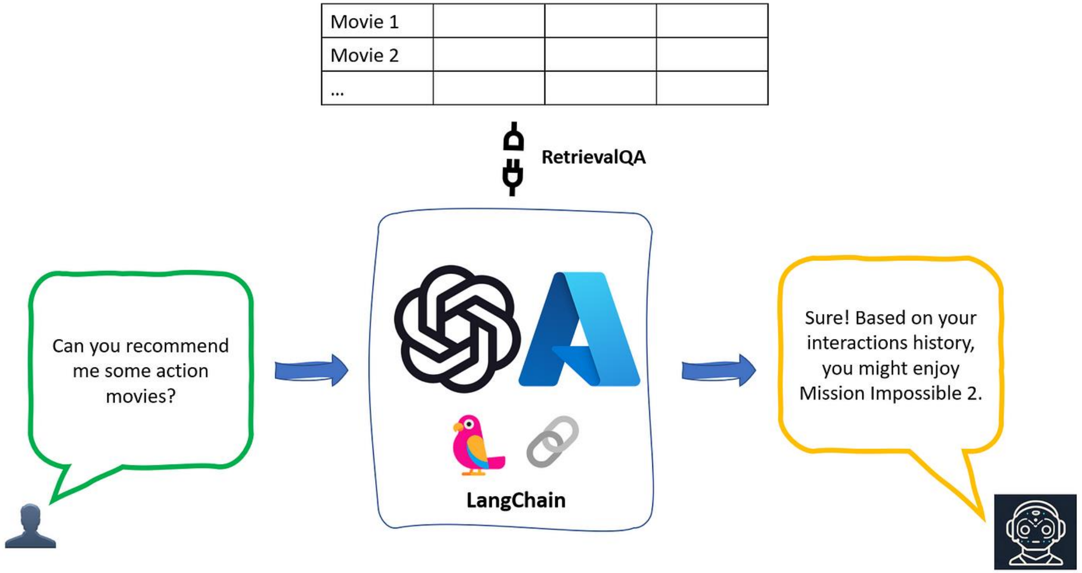
- Recommending an item as generating a special token
- LLMs can equip RecSys with the ability to connect contextual language cues
    - A customer searches “summer clothing"
        - Classic recsys: receive broad suggestions based on past purchases or generic category trends
        - LLM recsys: analyze not just “summer clothing”, but also analyze search queries, product descriptions, and user reviews, to suggest items with specific attributes like "lightweight," "breathable," or "eco-friendly."



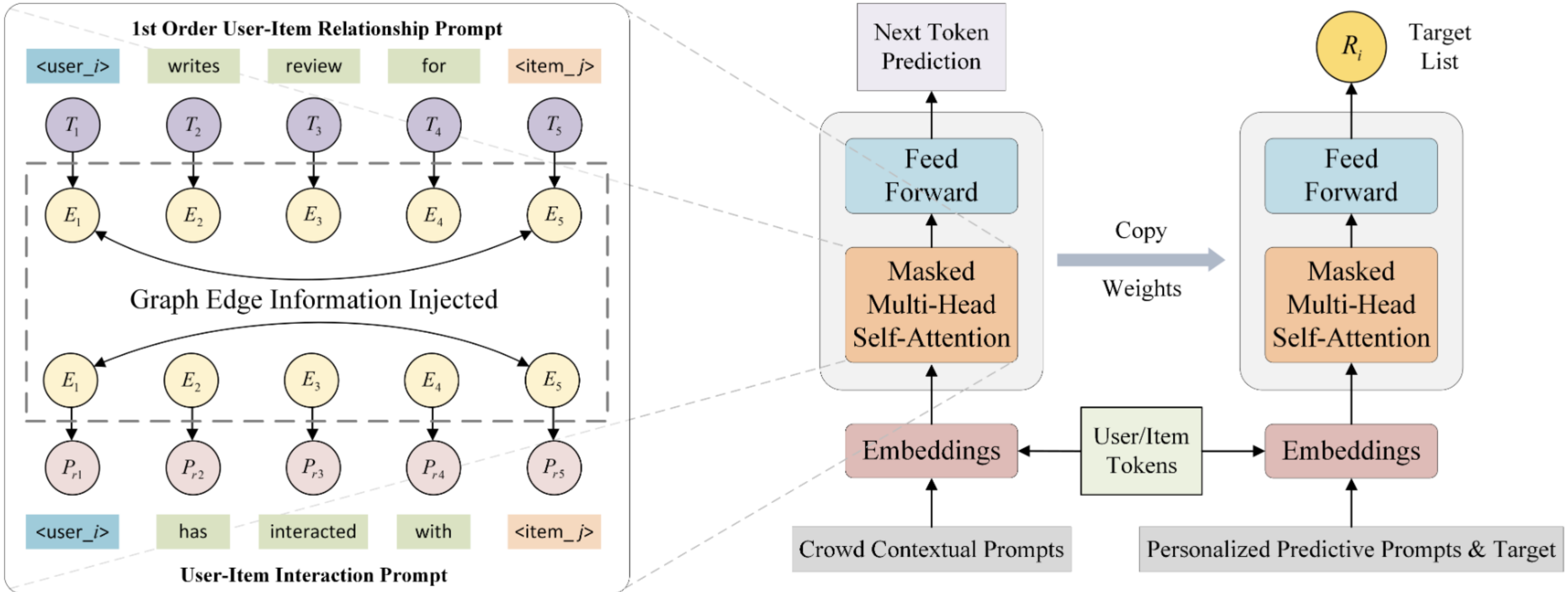
Our AI Task: Integrating the Graph and LLM Views for Enhancing RecSys



+



Overview of Proposed Solution



(1) Graph Knowledge Guided Attention for GPT-2 Backbone

(2) Pre-training Stage

(3) Fine-tunine Stage
(4) Prediction

Step 1: Graph Connectivity Guided Attentive LLM Backbone Model

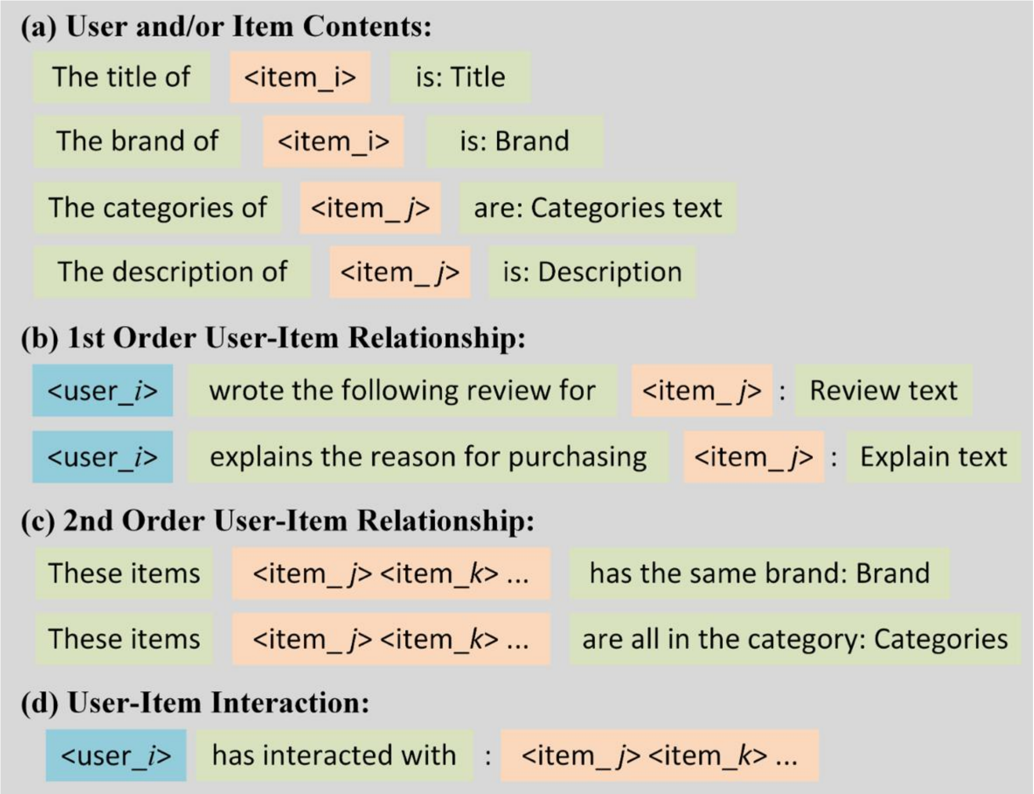
- Reformulate RecSys into a probabilistic generative problem in response to prompts
- GPT2 as LLM base model
  - The Transformer architecture
  - Pre-train on vast text datasets to predict subsequent words
  - The attention mechanism: determine how much attention to pay to each word when generating the next word in the sequence
- Integrating two types of graph connectivity into attention

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}} + R\right) V,$$

  - The direct connection: 1/0 binary connection indicator between nodes
  - The indirect connection: a normalized shortest path score between nodes computed based on the entire graph

Step 2: Pretraining The Backbone Model

- A: Data collection
  - user descriptions
  - item descriptions
  - user reviews for items
  - historical events that users interact (e.g., rate or purchase) with items
- B: Constructing training data with contextual and relational cues
- C: Optimization Objective
  - Given the training data, GPT2 is to predict the next token in the textual sequence
  - The objective is to maximize the token generation likelihood of the training data.



Step 3: Using Personalized Predictive Prompts for Fine-tuning

- A: Personalized Predictive Prompts

(prompt) <user_i> has interacted with <item_j'> <item_k'>

The user will interact with : (target) R_i

- B: Optimization Objective: minimizing the rating prediction loss between gold rating and estimated rating

Experimental Data

Dataset	User	Item	Interaction	Content
AM-Beauty	10,553	6,086	94,148	165,228
AM-Toys	11,268	7,309	95,420	170,551
AM-Sports	22,686	12,301	185,718	321,887
AM-Luxury	2,382	1,047	21,911	15,834
AM-Scientific	6,875	3,484	50,985	43,164
AM-Instruments	20,307	7,917	183,964	143,113
AM-Food	95,421	32,180	834,514	691,543

Baseline Methods

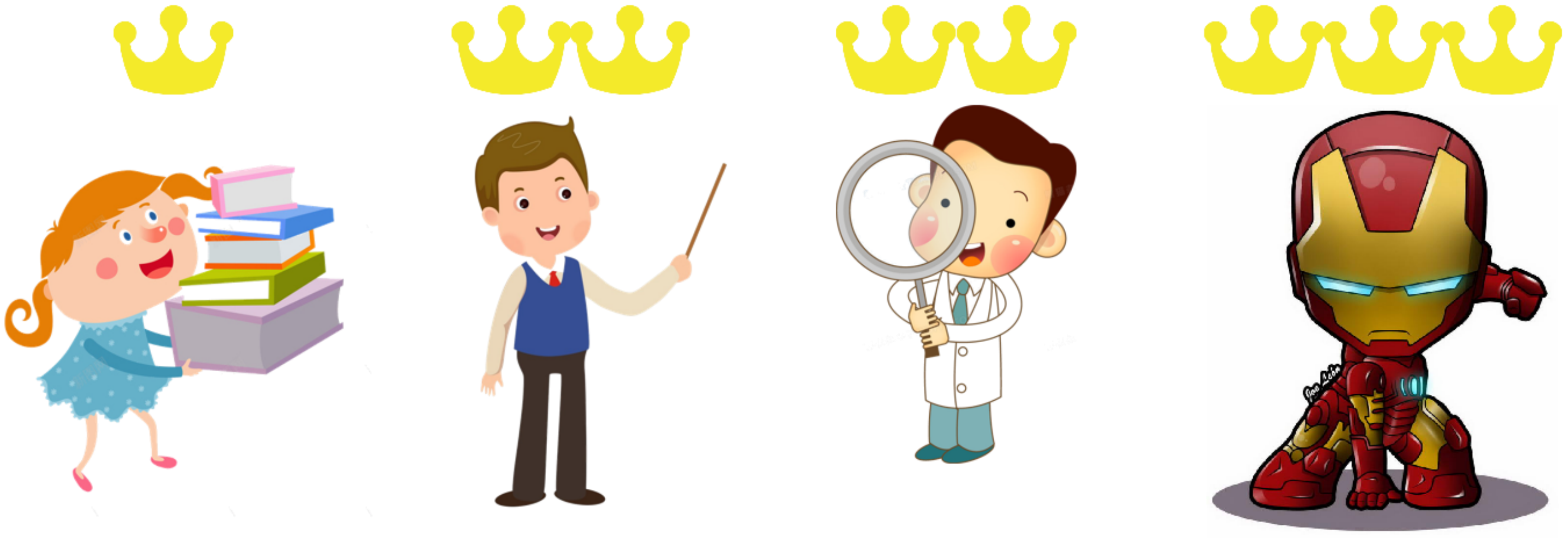
- Multi-VAE [21] is an ID-based collaborative filtering method that completes recommendation tasks by using a polynomial likelihood variational autoencoder to reconstruct ratings.
- MD-CVAE [51] extends Multi-VAE by introducing dual feature VAE on text features to regularize rating reconstruction.
- BERT4REC [36] uses BERT-like mask language modeling to learn user/item embeddings, integrated with a bidirectional self-attention mechanism, for recommendations.
- S^3Rec [50] extends BERT4Rec by adding auxiliary tasks such as item attribute prediction to enhance MLM, which can integrate content features for self-supervised learning.
- UniSRec [14] leverages item description texts to learn transferable sequence representations across different domains, employing a lightweight architecture with contrastive pre-training tasks for robust performance.
- FDSA [47] enhances prediction accuracy by not only considering item-level transition patterns but also integrating and weighing heterogeneous item features to capture both explicit and implicit feature-level sequences.
- SASRec [17] captures long-term user behaviors by selectively focusing on relevant past actions.
- GRU4Rec [13] focuses on short session data where traditional matrix factorization fails and demonstrates significant improvements over conventional item-to-item methods.
- LightGCN [12] ignores feature transformation and nonlinear activation to enhance training efficiency and recommendation performance.

Comparison Results of Rating Prediction

Dataset	Metric	Multi-VAE	MD-CVAE	LightGCN	BERT4Rec	S^3Rec	UniSRec	FDSA	SASRec	GRU4Rec	LLM-NoPretrain	LLM-NoFineTune	LLM-NoGKIA	LLM-NoGHIP	Ours
AM-Beauty	Recall@20	0.1295	0.1472	0.1429	0.1126	0.1354	0.1462	0.1447	0.1503	0.0997	0.0464	0.0441	0.1225	0.1267	**0.1590**
	Recall@40	0.1720	0.2058	0.1967	0.1677	0.1789	0.1898	0.1875	0.2018	0.1528	0.0709	0.0691	0.1665	0.1799	**0.2177**
	NDCG@100	0.0835	0.0871	0.0890	0.0781	0.0867	0.0907	0.0834	0.0929	0.0749	0.0339	0.0323	0.0790	0.0827	**0.1029**
AM-Toys	Recall@20	0.1076	0.1107	0.1096	0.0853	0.1064	0.1110	0.0972	0.0869	0.0657	0.0477	0.0580	0.0896	0.0858	**0.1349**
	Recall@40	0.1558	0.1678	0.1558	0.1375	0.1524	0.1457	0.1268	0.1146	0.0917	0.0689	0.1003	0.1272	0.1179	**0.1873**
	NDCG@100	0.0781	0.0812	0.0775	0.0532	0.0665	0.0638	0.0662	0.0525	0.0439	0.0330	0.0481	0.0612	0.0594	**0.0876**
AM-Sports	Recall@20	0.0659	0.0714	0.0677	0.0521	0.0616	0.0714	0.0681	0.0541	0.0720	0.0449	0.0394	0.0555	0.0558	**0.0764**
	Recall@40	0.0975	0.1180	0.0973	0.0701	0.0813	0.1143	0.0866	0.0739	0.1086	0.0719	0.0613	0.0846	0.0830	**0.1240**
	NDCG@100	0.0446	0.0514	0.0475	0.0305	0.0438	0.0504	0.0475	0.0361	0.0498	0.0322	0.0278	0.0391	0.0379	**0.0535**
AM-Luxury	Recall@20	0.2306	0.2771	0.0000	0.2076	0.2241	**0.3091**	0.2759	0.2550	0.2126	0.1872	0.1885	0.2474	0.2679	0.3066
	Recall@40	0.2724	0.3206	0.0000	0.2404	0.2672	**0.3675**	0.3176	0.3008	0.2522	0.2233	0.2254	0.2880	0.3028	0.3441
	NDCG@100	0.1697	0.2064	0.0000	0.1617	0.1542	0.2010	0.2107	0.1965	0.1623	0.1223	0.1235	0.1834	0.2065	**0.2331**
AM-Scientific	Recall@20	0.1069	0.1389	0.0000	0.0871	0.1089	**0.1492**	0.1188	0.1298	0.0849	0.0708	0.0668	0.1383	0.1206	0.1480
	Recall@40	0.1483	0.1842	0.0000	0.1160	0.1541	**0.1954**	0.1547	0.1776	0.1204	0.1037	0.0960	0.1822	0.1575	0.1908
	NDCG@100	0.0766	0.0872	0.0000	0.0606	0.0715	0.1056	0.0846	0.0864	0.0594	0.0568	0.0465	0.0940	0.0810	**0.1072**
AM-Instruments	Recall@20	0.1096	0.1398	0.0000	0.1183	0.1352	0.1684	0.1382	0.1483	0.1271	0.0766	0.0727	0.1387	0.1426	**0.1698**
	Recall@40	0.1628	0.1743	0.0000	0.1531	0.1767	0.2239	0.1787	0.1935	0.1660	0.1004	0.0948	0.1741	0.1779	**0.2265**
	NDCG@100	0.0735	0.1040	0.0000	0.0922	0.0894	0.1075	0.1080	0.0934	0.0998	0.0500	0.0478	0.1042	0.1044	**0.1312**
AM-Food	Recall@20	0.1062	0.1170	0.0000	0.1036	0.1157	0.1423	0.1099	0.1171	0.1140	0.0224	0.0204	0.1275	0.1264	**0.1438**
	Recall@40	0.1317	0.1431	0.0000	0.1284	0.1456	0.1661	0.1317	0.1404	0.1389	0.0299	0.0274	0.1559	0.1487	**0.1673**
	NDCG@100	0.0727	0.0863	0.0000	0.0835	0.0926	0.1024	0.0904	0.0942	0.0910	0.0153	0.0141	0.0898	0.0963	**0.1119**

RecSys for LLM: Dynamic Query-LLM Routing as Adaptive Choice Modeling in RecSys

Motivation



Mistral 7B | Llama 2 Chat (70B) | GPT-3.5 Turbo | GPT-4

Response Quality

We want to select the model which can answer the query correctly

Cost

At the set level, the comparable response quality with lower cost is possible

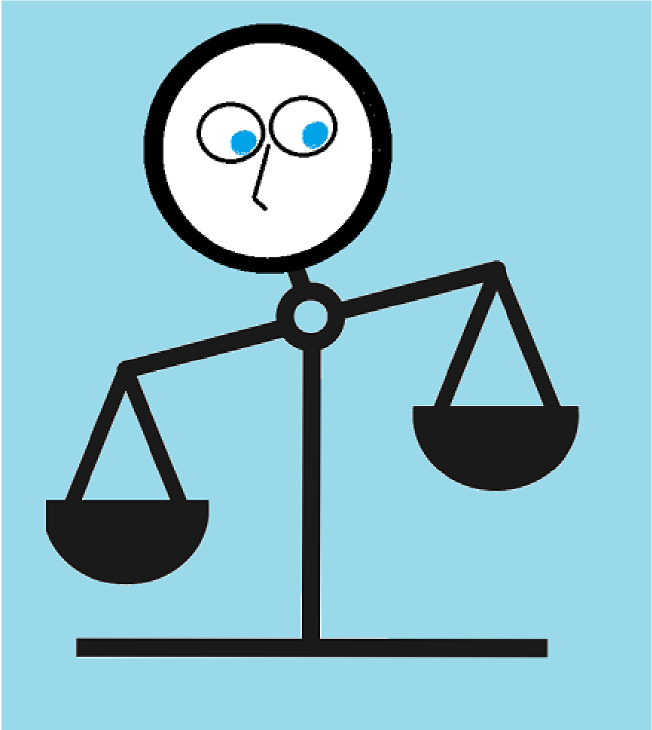
Latency

We don't want the query to queue for a long time

We aim to balance performance, cost, and latency to achieve the trade-off.

The AI Task

- The LLM routing task aims to identify the most suitable model for each query in the query stream to
  - maximize response quality
  - minimize cost and latency



Challenges

- A dynamic query stream
- Trade-offs among quality, cost, and latency
- Navigating a varying (e.g., new LLM addition or old LLM removal) set of LLM candidates over time
- Enabling continual learning in even after deployment

Why Existing Literature Isn't Sufficient

- Non-predictive (Cascading): try small LM first, then decide to switch to LLM or not
  - each query is answered by **more than one LM** (higher cost, higher delay)
  - the decision maker is another LM, requiring **extra** time and computing resources
  - when multiple LLMs are involved, it is **hard to sequence** them (from small to large)
- Predictive

Why Existing Literature Isn't Sufficient

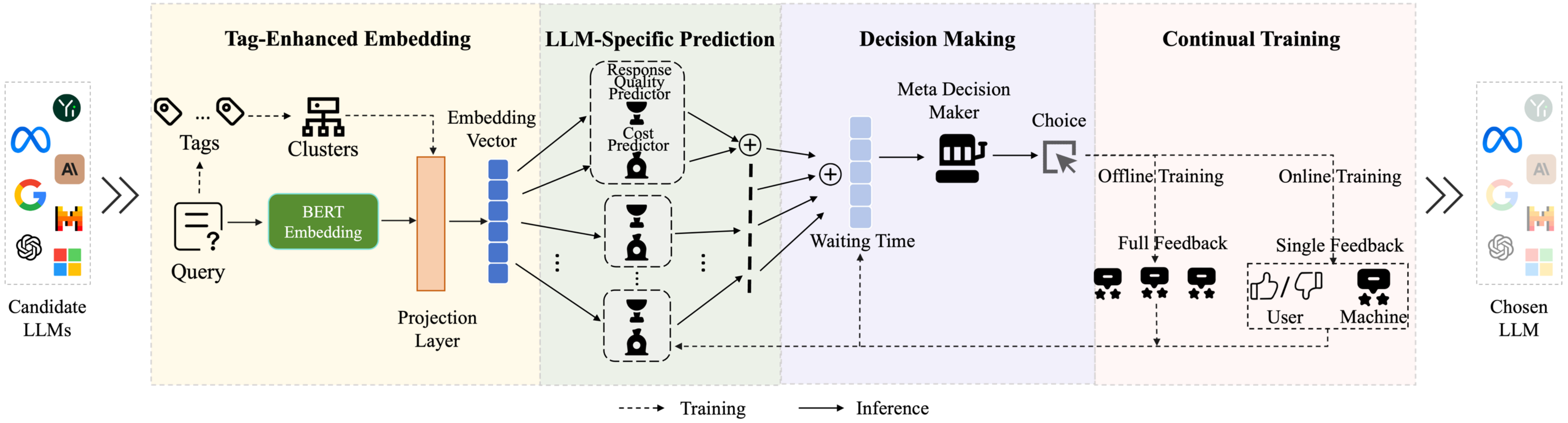
- Predictive: predict the features and characteristics of the query

  - classifier: **no strong connection** between the query and the final label (new LLM -> new label)
  - response quality predictor: **no cost consideration**
  - set-level optimization: some queries may be **ignored** (users may be disappointed)
  - **common** embedding vector for **different** candidate LMs
  - no **time (system information)** limitation consideration

The Unique Perspective

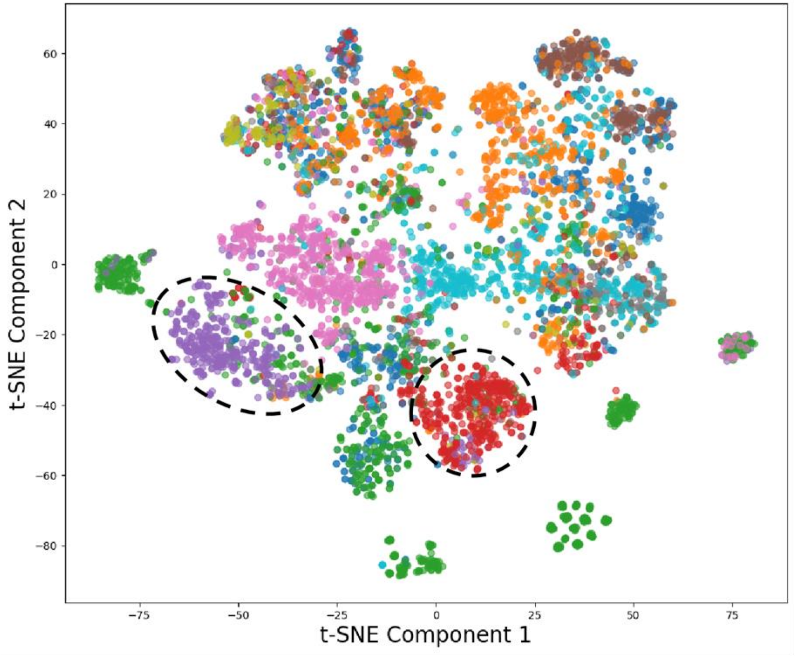
- Dynamic routing system:
  - Queries arrive sequentially → query level operation
- Predictive pipeline:
  - No LLM inference is needed when routing
- Informative embeddings:
  - Use query tags to enhance the encoder
- Trade-off:
  - Budget: adjust the weight between cost and performance
  - Delay: employ latency penalty when choosing the final LLM

Overview of the Proposed Solution

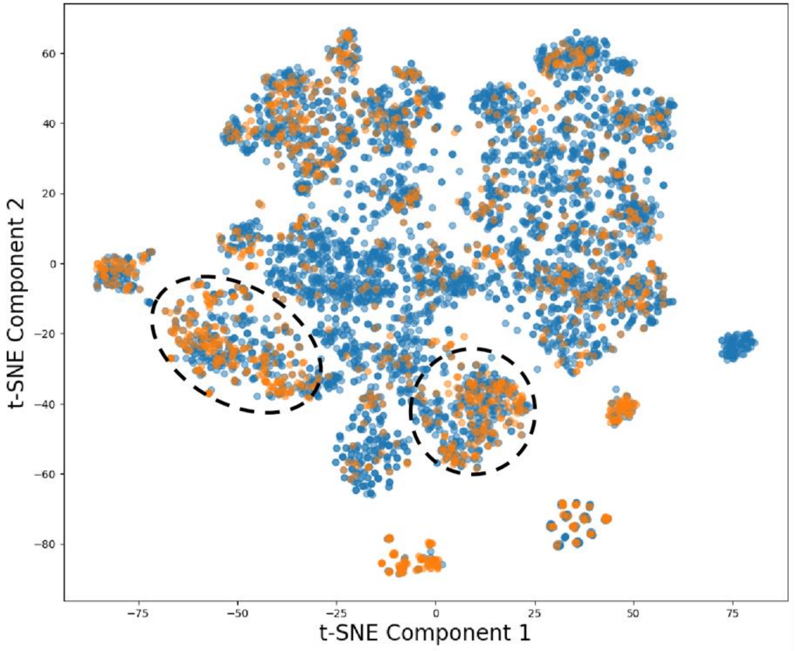


Step 1: Tag-Enhanced Query Embedding (1)

- Why using tags as representation of queries? The semantics of query tags closely connect to LLM response quality.



Each color representing a cluster of queries



GPT-4 has a higher frequency of errors (marked as orange) in the legal (marked as red) and math (marked as purple) domains

Tag-Enhanced Embedding

- BERT-based encoder for sentence embedding

$$e_n = \text{Encoder}(q_n),$$

- Employ the InsTag [1] to generate fine-grained tags, then cluster them

- Train encoder based on cluster labels

$$\mathcal{L}_{\text{intra}} = -\frac{1}{|Q|}\sum_{i=1}^{|Q|}\log\frac{\exp(\mathbf{e}_i \cdot \boldsymbol{\mu}_i)}{\sum_{j=1}^{|D|}\exp(\mathbf{e}_i \cdot \boldsymbol{\mu}_j)}.$$

$$\mathcal{L}_{\text{inter}} = \frac{1}{|D|}\sum_{j=1}^{|D|}\log\sum_{k\neq j}\exp(\boldsymbol{\mu}_j \cdot \boldsymbol{\mu}_k).$$

[1] Lu, Keming, et al. "# instag: Instruction tagging for analyzing supervised fine-tuning of large language models." *The Twelfth International Conference on Learning Representations*. 2023.

Estimating the Accuracy, Latency, Costs of a LLM-Query Pair

- For each LLM, we learn a regression model to predict the response quality of the LLM on a query:

$$\hat{p}_{n,l} = f_l^{\mathsf{rq}}(\mathbf{e}_n; \boldsymbol{\theta}_l^{\mathsf{rq}}),$$

- Predict response length for estimating total cost:

$$\hat{\mathtt{len}}_{n,l}^{\mathsf{res}} = f_l^{\mathsf{rl}}(\mathbf{e}_n; \boldsymbol{\theta}_l^{\mathsf{rl}}),$$

$$\hat{c}_{n,l} = \underbrace{\mathtt{len}_{n,l}^{\mathsf{prm}} \cdot \mathtt{price}_l^{\mathsf{prm}}}_{\text{input cost}} + \underbrace{\hat{\mathtt{len}}_{n,l}^{\mathsf{res}} \cdot \mathtt{price}_l^{\mathsf{res}}}_{\text{output cost}},$$

Meta Decision Maker

- Select the most suitable according to the score = trade-offs the predicted quality and cost + potential prediction uncertainty - waiting time
- Balancing of response quality and cost
- Uncertainty is employed to correct errors in predicting
- Time penalty prevents the excessive waiting time

Continual Learning (1): Offline Training

- Offline Training:
  - Before the deployment
  - Full feedback from all candidate LLMs (arms)

- Predictors are updated:

$$\boldsymbol{\theta}_l^{\mathtt{rq}} := \boldsymbol{\theta}_l^{\mathtt{rq}} - \eta_1 \cdot \nabla_{\boldsymbol{\theta}_l^{\mathtt{rq}}} \mathcal{L}(p_{n,l}, \hat{p}_{n,l}),$$

$$\boldsymbol{\theta}_l^{\mathtt{rl}} := \boldsymbol{\theta}_l^{\mathtt{rl}} - \eta_2 \cdot \nabla_{\boldsymbol{\theta}_l^{\mathtt{rl}}} \mathcal{L}(\mathtt{len}_{n,l}^{\mathtt{res}}, \hat{\mathtt{len}}_{n,l}^{\mathtt{res}}),$$

$$\mathbf{A}_l := \mathbf{A}_l + \mathbf{e}_n^T \cdot \mathbf{e}_n.$$

Continual Learning (2): Online Training

- Online Training:
  - Post-deployment
  - Partial feedback only from the selected and highly-scored LLMs over iterations
- Refined Feedback: the same as offline training
- Binary Feedback:

$$s'_{n,l} = s_{n,l} + \kappa_{n,l} \cdot s^{\mathsf{df}}_{n,l}, \qquad \left[s^{\mathsf{df}}_{n,1}, s^{\mathsf{df}}_{n,2}, \ldots, s^{\mathsf{df}}_{n,|M|}\right] = f^{\mathsf{df}}(\mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}).$$

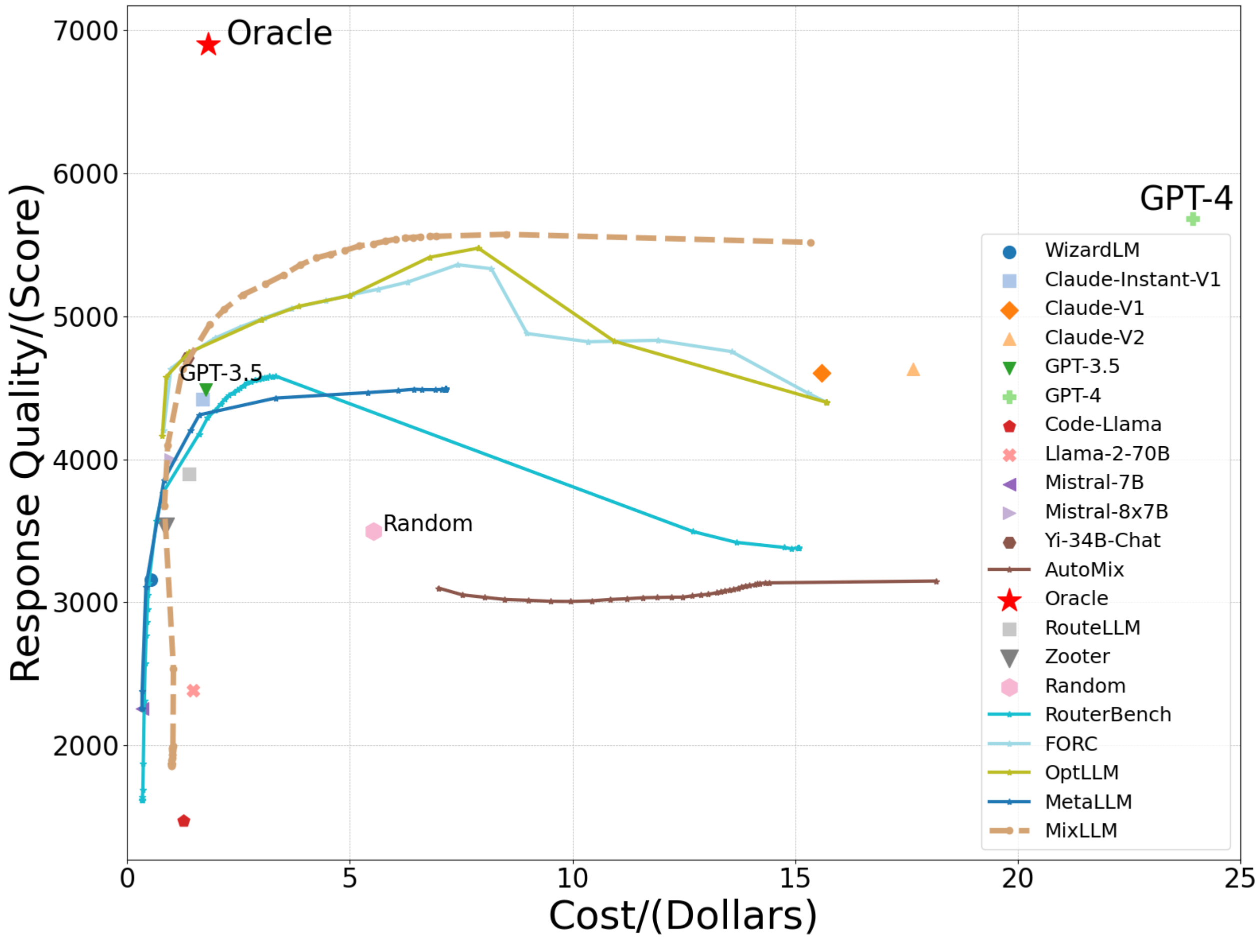
$$\kappa_{n,l} = \frac{1}{\mathrm{Var}_n[s^{\mathsf{df}}_{n,l}] + \epsilon},$$

$$\boldsymbol{\theta}^{\mathsf{df}} := \boldsymbol{\theta}^{\mathsf{df}} - \eta_3 \cdot \nabla_{\boldsymbol{\theta}^{\mathsf{df}}} \log \pi(m^*_n \mid \mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}) \cdot r_n.$$

$$\nabla_{\boldsymbol{\theta}^{\mathsf{df}}} \log \pi(m^*_n \mid \mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}) = \nabla_{\boldsymbol{\theta}^{\mathsf{df}}} \left(s^{\mathsf{df}}_{n,m^*_n} - \log \sum_{k=1}^{L} \exp\left(s^{\mathsf{df}}_{n,k}\right) \right)$$

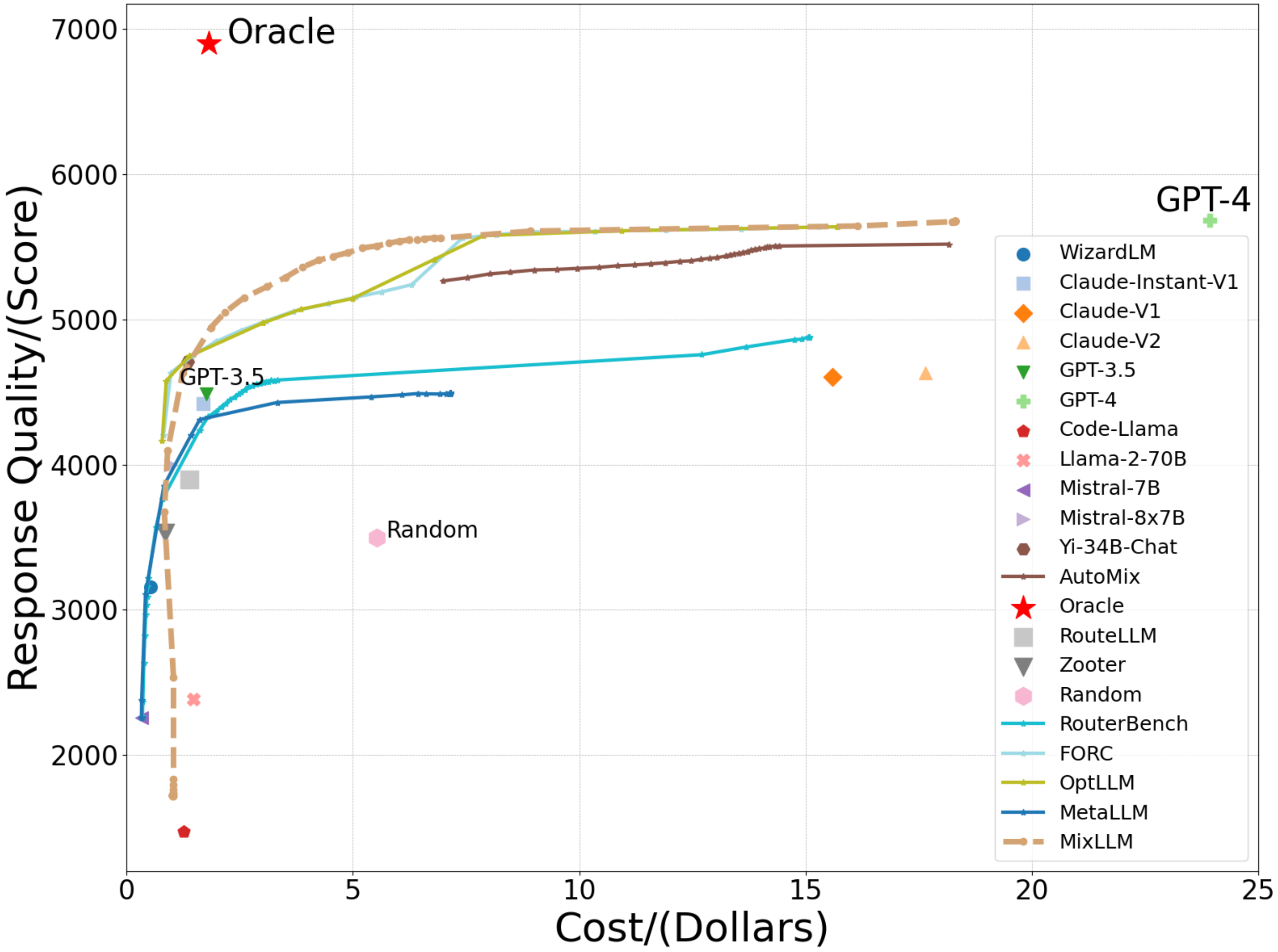
Results: Performance

- Our method outperforms baselines and maintains performance when latency is high.
- Under time constraints, performance may decline even with a high budget, as some queries might be ignored due to high latency



Results: Performance

- MixLLM performs well even without the time penalty.



Results: Continuous training

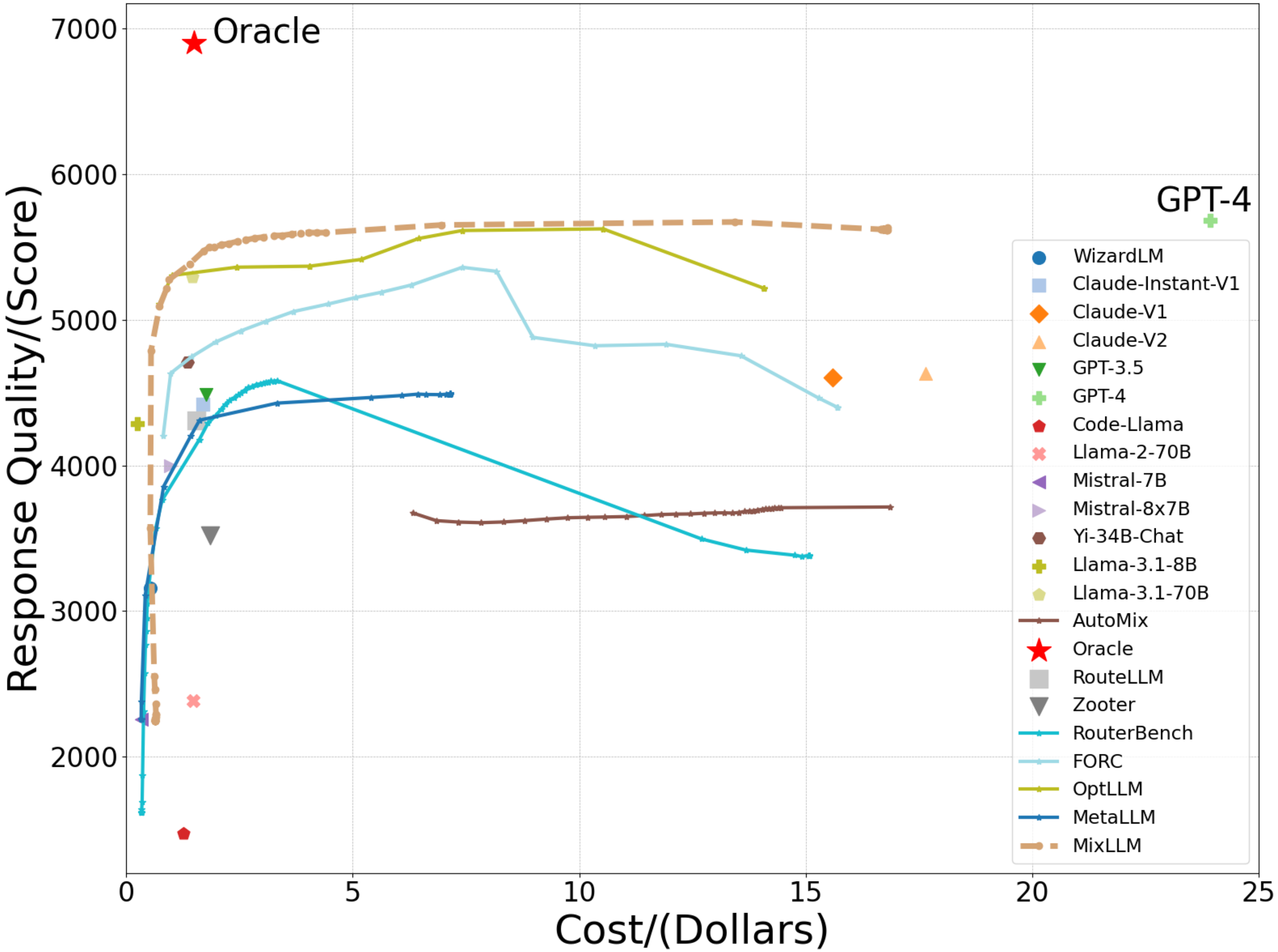
- In real-world applications, collecting full feedback is difficult and expensive,
- But the responses to queries can serve as partial feedback. And the amount of data during inference will far exceed that during training.

- Continuous training offers improved performance.

Setting	Offline : Online		
	80:20	50:50	30:70
Without Online Training	75.54%	71.98%	69.74%
With Refined Feedback	76.45%	72.99%	71.29%
Improvement	**1.21%**	**1.39%**	**2.22%**
With Binary Feedback	75.93%	72.37%	70.65%
Improvement	**0.52%**	**0.53%**	**1.31%**

Results: Adaptability

- With the introduction of the powerful Llama 3.1 models, MixLLM achieves 98.55% of GPT-4's response quality while reducing the cost to just 18.36%.
- MixLLM is highly efficient, as the parameters in the original arms remain unchanged.



Conclusion Remark

Q & A

Yanjie Fu

Yanjie.fu@asu.edu

School of Computing and AI

Arizona State University

Thank You for Listening

Yanjie Fu

Yanjie.fu@asu.edu

School of Computing and AI

Arizona State University