CSCE 670 - Information Storage and Retrieval

Week 13: Large Language Models for Ranking and Recommendation

Yu Zhang

yuzhang@tamu.edu

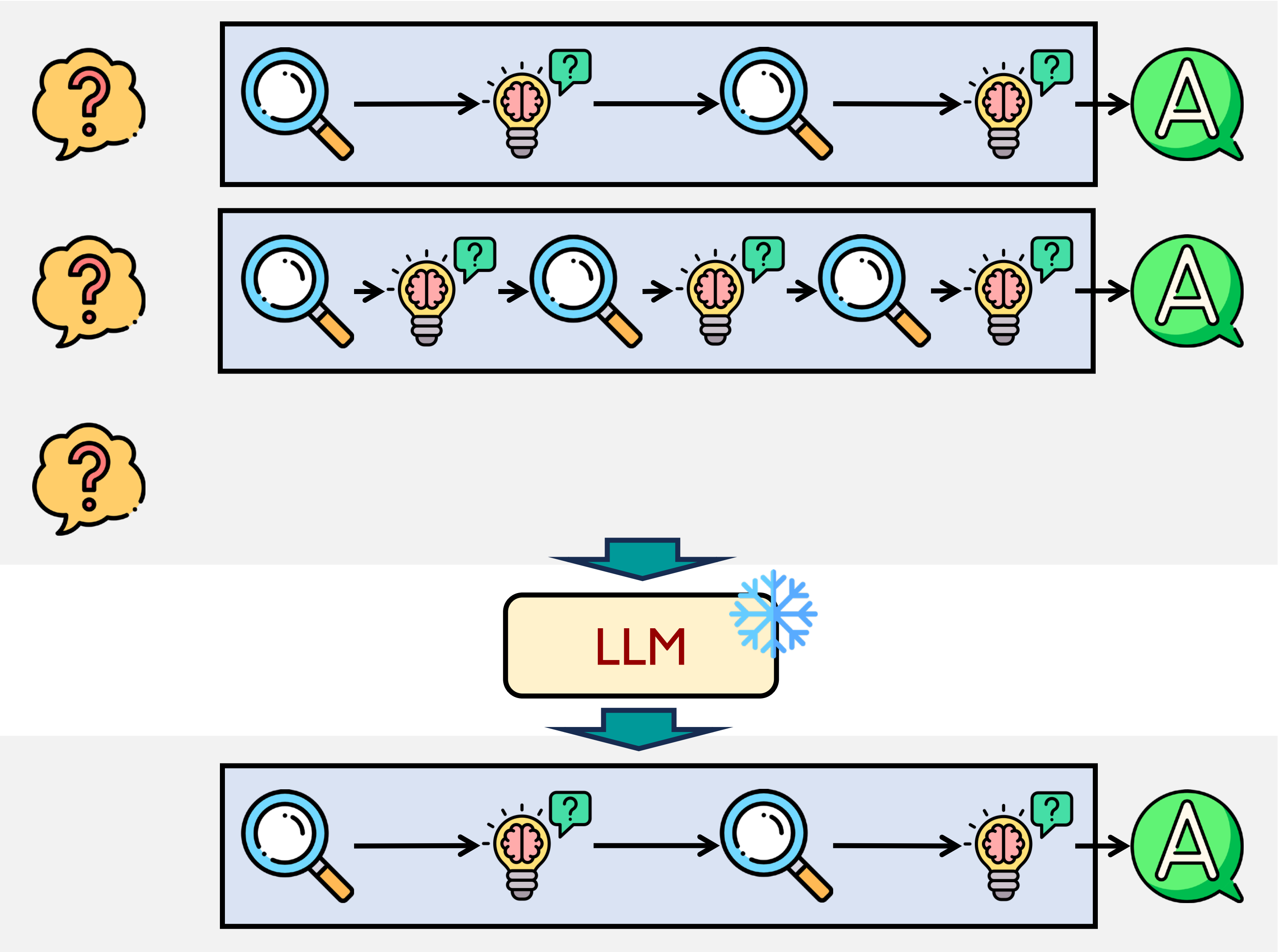
Course Website: https://yuzhang-teaching.github.io/CSCE670-S26.html

Large Language Models with Search Engines (Cont'd)

Recap: How to teach LLMs to use search engines?

- Solution 1: Few-Shot Prompting
- Inference-time technique
- Put reasoning and action examples in the prompt, then ask the question

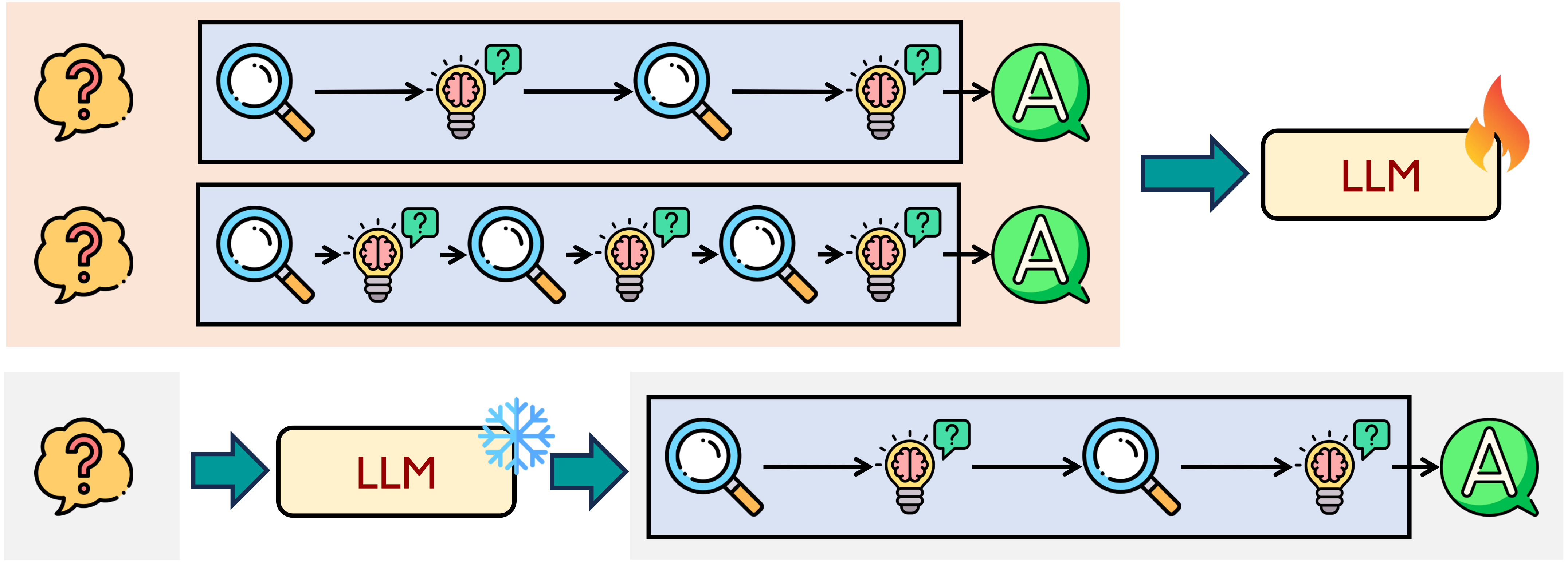
- Limitation: The ability to call a search engine is not internalized through model parameter updates



Recap: How to teach LLMs to use search engines?

- Solution 2: Supervised Fine-Tuning
- Training-based solution
- Collect reasoning and search trajectories to fine-tune the LLM

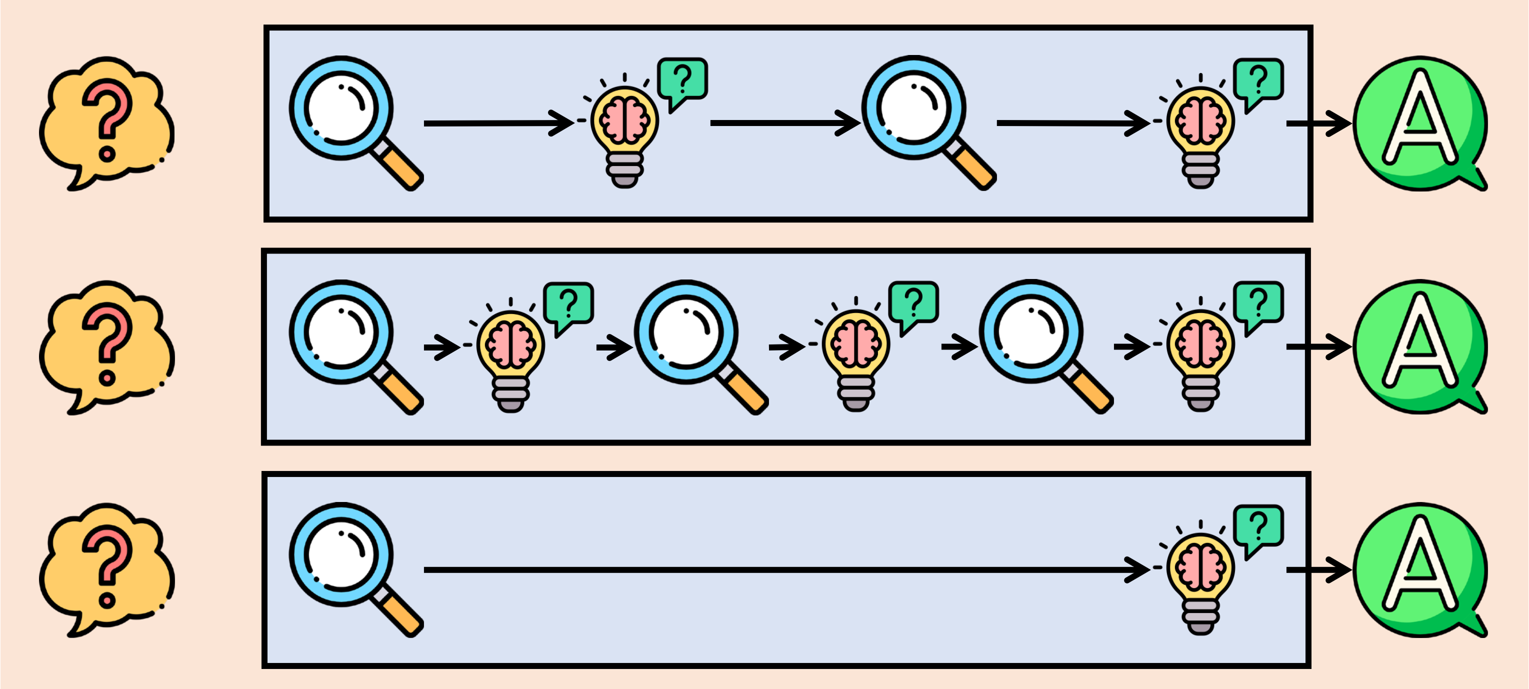
- Limitation 1: Hard to scale due to reliance on high-quality labeled trajectories
- Limitation 2: Poor generalization to unseen tools (e.g., trained on BM25 but deployed on Google API)



Recap: How to teach LLMs to use search engines?

- Solution 3: Reinforcement Learning
  - Train the model to generate outputs that maximize the reward
  - We no longer need annotated trajectories!

I do not have this, …



…, but I have this
(which is enough for RL)!

Search-R1 [Jin et al., COLM 2025]

Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning

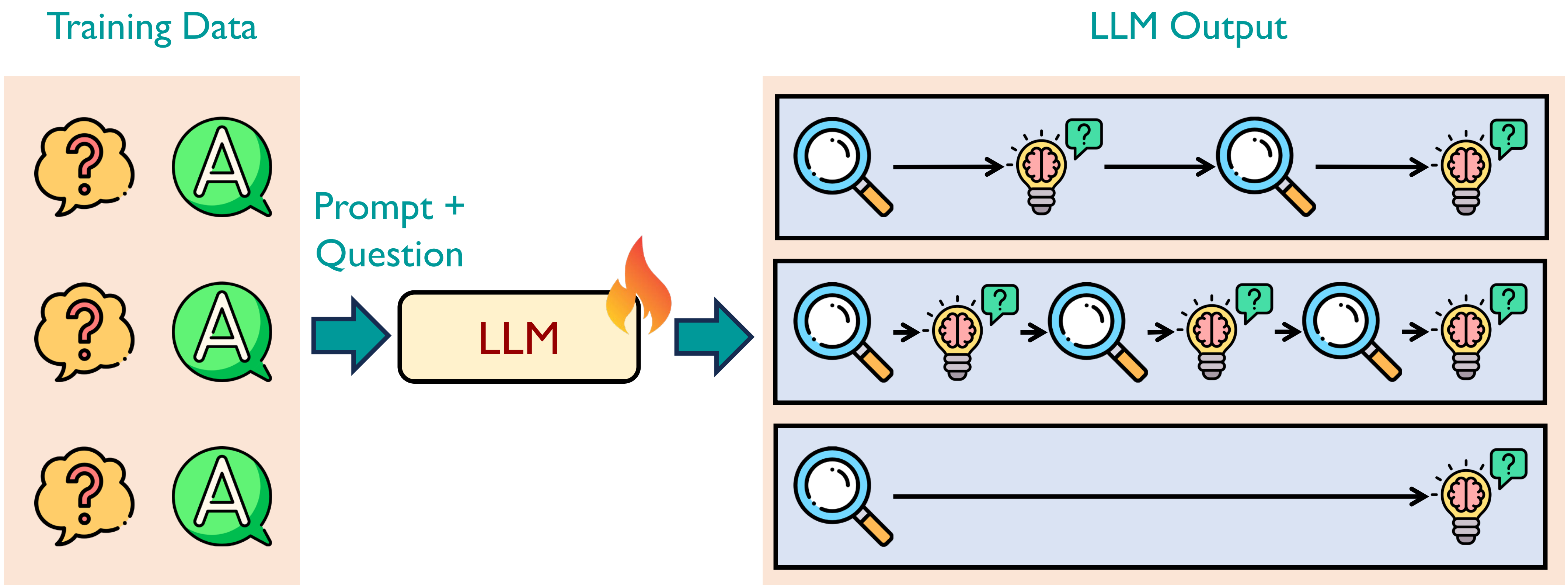
Bowen Jin[1], **Hansi Zeng**[2], **Zhenrui Yue**[1], **Jinsung Yoon**[3], **Sercan Ö. Arık**[3], **Dong Wang**[1], **Hamed Zamani**[2], **Jiawei Han**[1]
[1] Department of Computer Science, University of Illinois at Urbana-Champaign
[2] Center for Intelligent Information Retrieval, University of Massachusetts Amherst
[3] Google Cloud AI Research
{bowenj4,zhenrui3,dwang24,hanj}@illinois.edu, {hzeng, zamani}@cs.umass.edu
{jinsungyoon,soarik}@google.com

Abstract

Efficiently acquiring external knowledge and up-to-date information is essential for effective reasoning and text generation in large language models (LLMs). Prompting advanced LLMs with reasoning capabili-

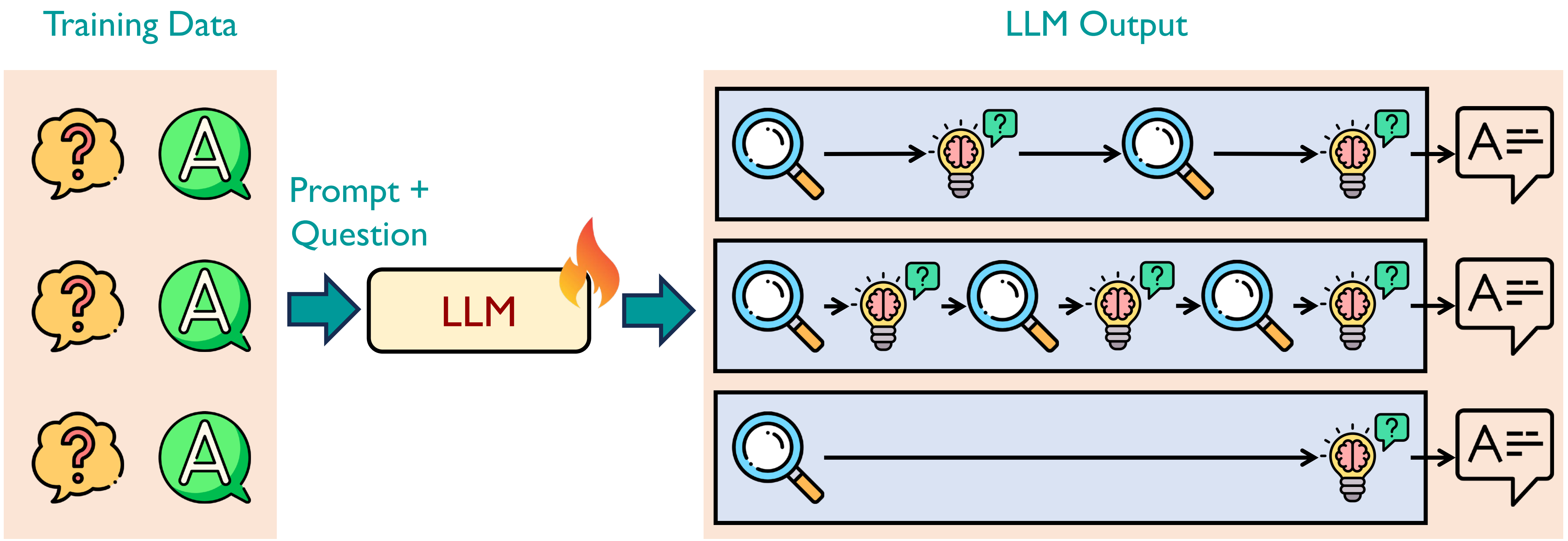
Reinforcement Learning with Outcome-Based Rewards

- Step 1: Let the LLM autonomously explore possible trajectories



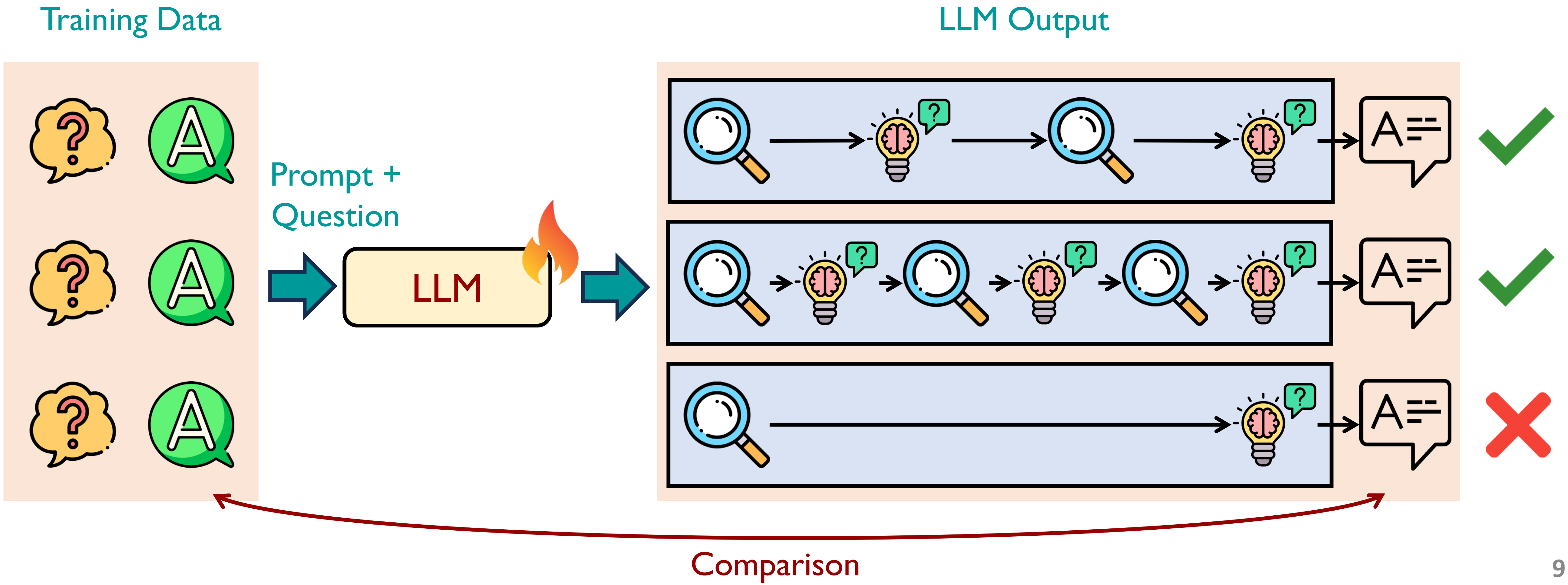
Reinforcement Learning with Outcome-Based Rewards

- Step 2: The LLM produces a final answer based on its search and reasoning trajectories.



Reinforcement Learning with Outcome-Based Rewards

- Step 3: We evaluate whether the final answer is correct. Trajectories leading to correct answers are rewarded, while those producing incorrect answers are penalized.



Reinforcement Learning with Outcome-Based Rewards

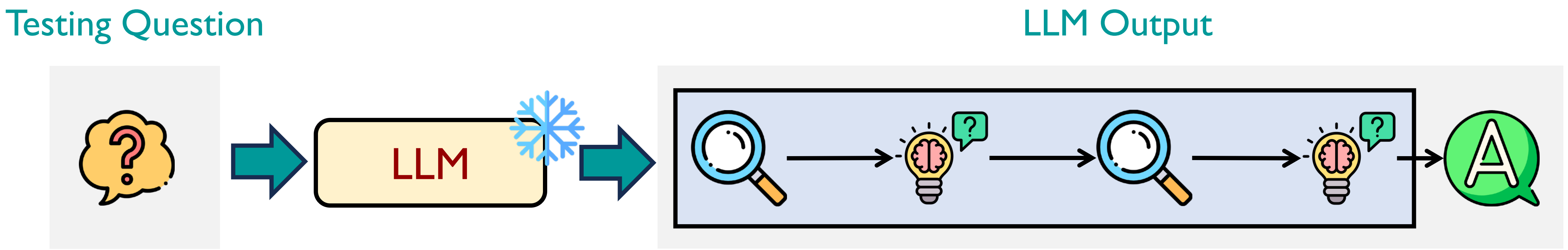
- Trajectories leading to correct answers are rewarded, while those producing incorrect answers are penalized. Informally, this means we …
  - Maximize the probability that the LLM generates good trajectories, and
  - Minimize the probability that the LLM generates bad ones.
- BUT these trajectories contain not only tokens generated by the LLM, but also tokens retrieved from external sources!

> The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

  - Tokens retrieved from external sources are NOT included in the probability computation (i.e., NOT used for training).

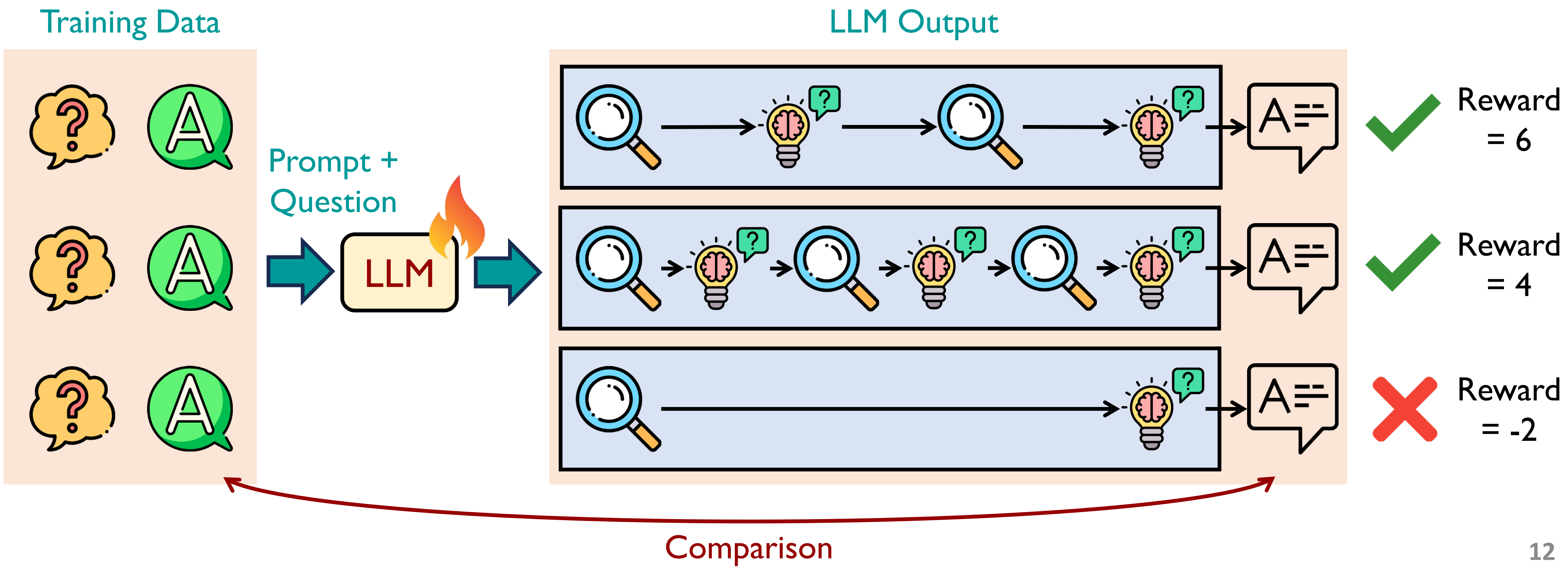
Reinforcement Learning with Outcome-Based Rewards

- Step 4: After training, the LLM learns search and reasoning capabilities that can be applied at inference time.



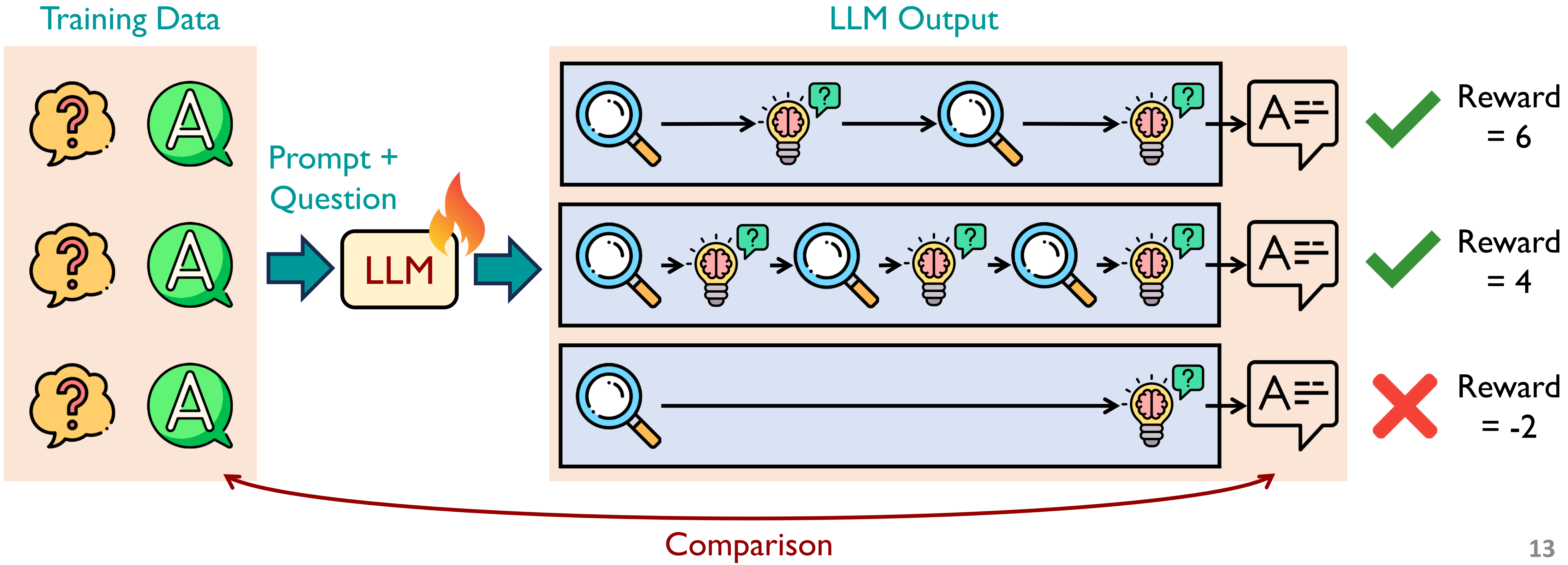
Reinforcement Learning with More Complicated Rewards

- E.g., Wang et al., *Acting Less is Reasoning More! Teaching Model to Act Efficiently*. arXiv 2025.
- Reward considers not only the correctness of the answer but also the efficiency of the process (e.g., number of search and reasoning steps).



Reinforcement Learning with More Complicated Rewards

- How do we apply the following two principles now? (i.e., how do we define "good" and "bad"?)
  - Maximize the probability that the LLM generates good trajectories, and
  - Minimize the probability that the LLM generates bad ones.

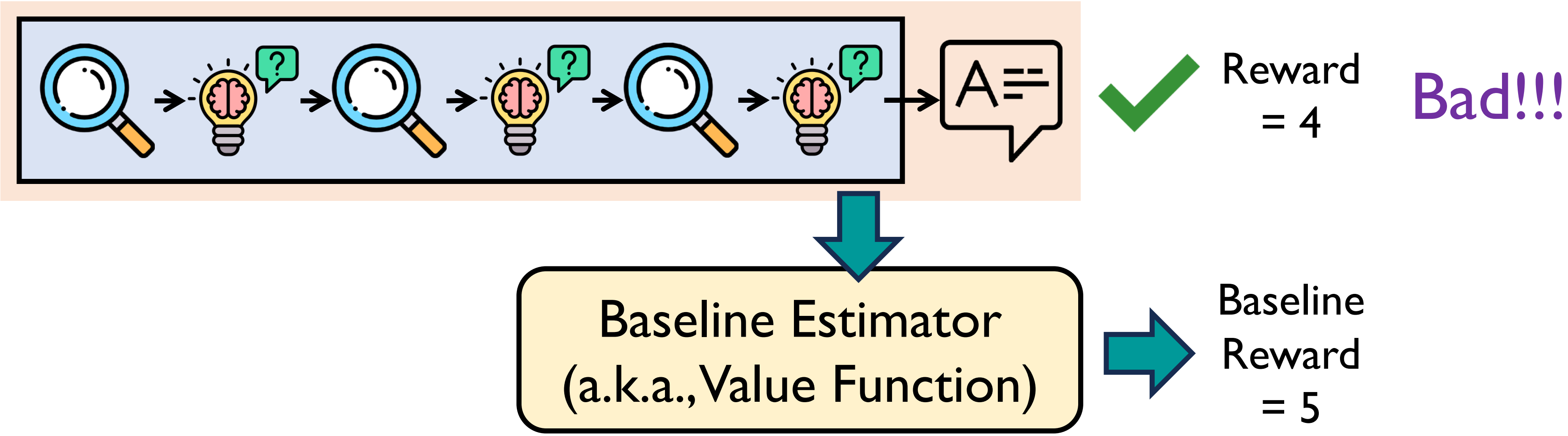


Reinforcement Learning with More Complicated Rewards

- Idea 1: Ignoring efficiency, a trajectory is good if the answer is correct, and bad if the answer is incorrect.
  - Bad idea. Why?
  - For the same question, if the LLM in the previous training epoch answered it correctly using 2 search steps, and the LLM in this epoch answers it correctly using 4 search steps, should we consider the current trajectory good?
  - (Equivalently, should we consider the current LLM to have performed better than in the previous epoch on this sample?)

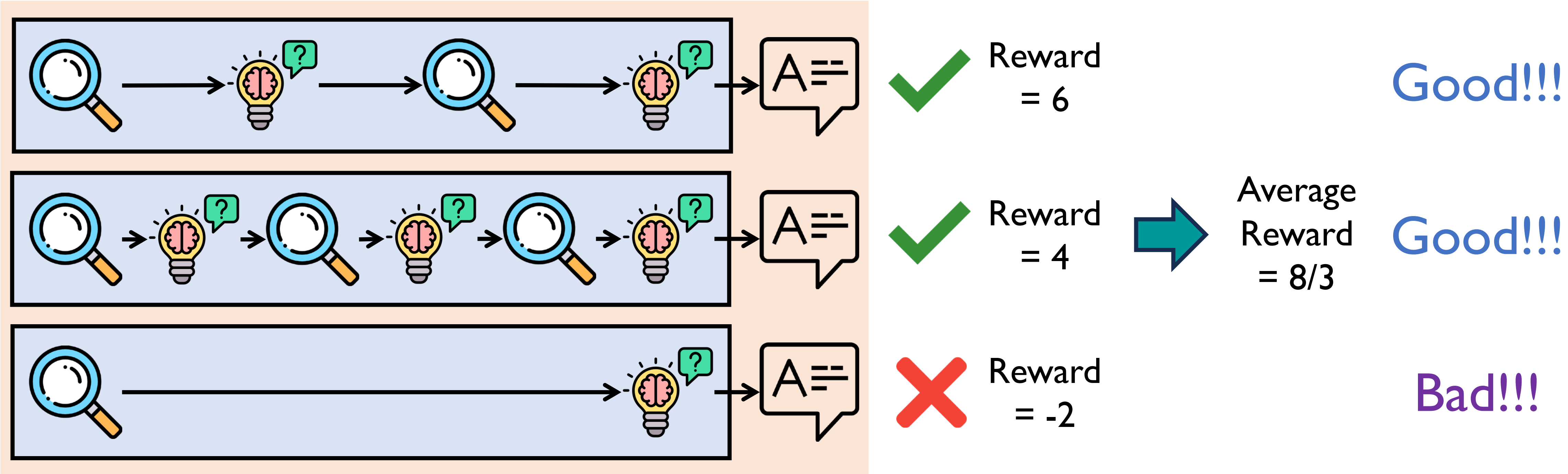
Reinforcement Learning with More Complicated Rewards

- Idea 2 (Proximal Policy Optimization, PPO [Schulman et al., 2017]): Trajectories with rewards higher than a learned baseline are considered good; those lower than the baseline are considered bad.
  - A trajectory is good only if it can improve the model over its (estimated) current performance.
  - However, this requires frequent use of the baseline estimator, which incurs high computation and memory costs.



Reinforcement Learning with More Complicated Rewards

- Idea 3 (Group Relative Policy Optimization, GRPO [Shao et al., 2024]): Trajectories with rewards higher than the group average are considered good; those lower than the group average are considered bad.
  - A trajectory is good only if it can improve the model over its (average) current performance.
  - Get rids of the baseline estimator

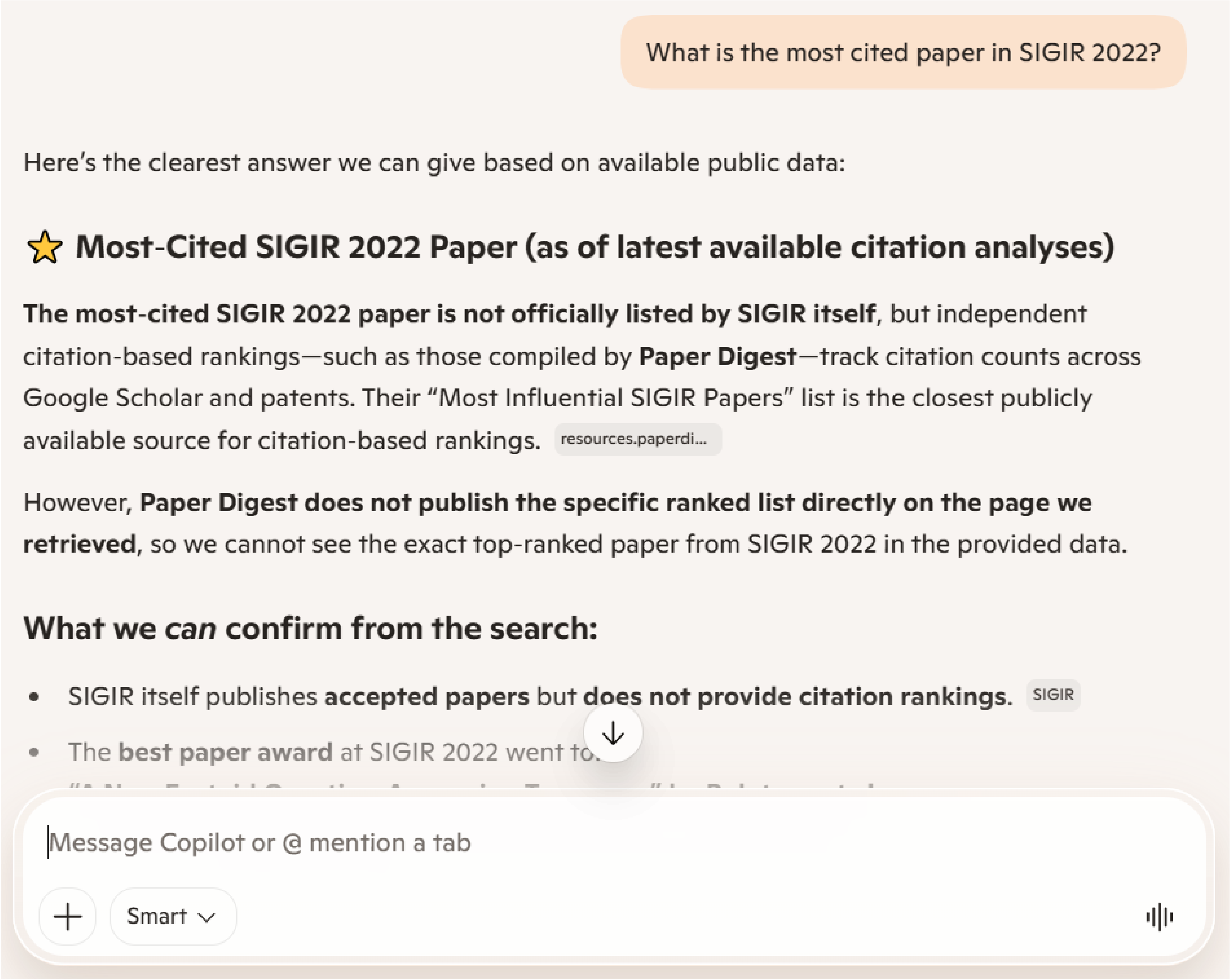


Summary: LLMs with Search Engines > LLMs

Methods	General QA			Multi-Hop QA				
	NQ†	TriviaQA*	PopQA*	HotpotQA†	2wiki*	Musique*	Bamboogle*	Avg.
Qwen2.5-7b-Base/Instruct								
Direct Inference	0.134	0.408	0.140	0.183	0.250	0.031	0.120	0.181
CoT	0.048	0.185	0.054	0.092	0.111	0.022	0.232	0.106
IRCoT	0.224	0.478	0.301	0.133	0.149	0.072	0.224	0.239
Search-o1	0.151	0.443	0.131	0.187	0.176	0.058	0.296	0.206
RAG	0.349	0.585	0.392	0.299	0.235	0.058	0.208	0.304
SFT	0.318	0.354	0.121	0.217	0.259	0.066	0.112	0.207
R1-base	0.297	0.539	0.202	0.242	0.273	0.083	0.296	0.276
R1-instruct	0.270	0.537	0.199	0.237	0.292	0.072	0.293	0.271
Rejection Sampling	0.360	0.592	0.380	0.331	0.296	0.123	0.355	0.348
Search-R1-base	**0.480**	**0.638**	**0.457**	**0.433**	0.382	**0.196**	**0.432**	**0.431**
Search-R1-instruct	0.393	0.610	0.397	0.370	**0.414**	0.146	0.368	0.385
Qwen2.5-3b-Base/Instruct								
Direct Inference	0.106	0.288	0.108	0.149	0.244	0.020	0.024	0.134
CoT	0.023	0.032	0.005	0.021	0.021	0.002	0.000	0.015
IRCoT	0.111	0.312	0.200	0.164	0.171	0.067	0.240	0.181
Search-o1	0.238	0.472	0.262	0.221	0.218	0.054	**0.320**	0.255
RAG	0.348	0.544	0.387	0.255	0.226	0.047	0.080	0.270
SFT	0.249	0.292	0.104	0.186	0.248	0.044	0.112	0.176
R1-base	0.226	0.455	0.173	0.201	0.268	0.055	0.224	0.229
R1-instruct	0.210	0.449	0.171	0.208	0.275	0.060	0.192	0.224
Rejection Sampling	0.294	0.488	0.332	0.240	0.233	0.059	0.210	0.265
Search-R1-base	**0.406**	**0.587**	**0.435**	0.284	0.273	0.049	0.088	0.303
Search-R1-instruct	0.341	0.545	0.378	**0.324**	**0.319**	**0.103**	0.264	**0.325**

Completed: How do search engines help LLMs?
Next: How do LLMs help search engines?

LLMs with Search Engines > Search Engines?



- In February 2023, Bing released an LLM-enhanced search engine, Bing Chat, powered by GPT-4.
- What effects did this have, and how could they be studied?
- At that time, Google had not yet introduced a comparable AI mode.

Impact on Scientific Literature Search [Kusumegi et al., Science 2025]

Science

Current Issue First release papers Archive About Submit man

HOME > SCIENCE > VOL. 390, NO. 6779 > SCIENTIFIC PRODUCTION IN THE ERA OF LARGE LANGUAGE MODELS

POLICY ARTICLE | ARTIFICIAL INTELLIGENCE

Scientific production in the era of large language models

With the production process rapidly evolving, science policy must consider how institutions could evolve

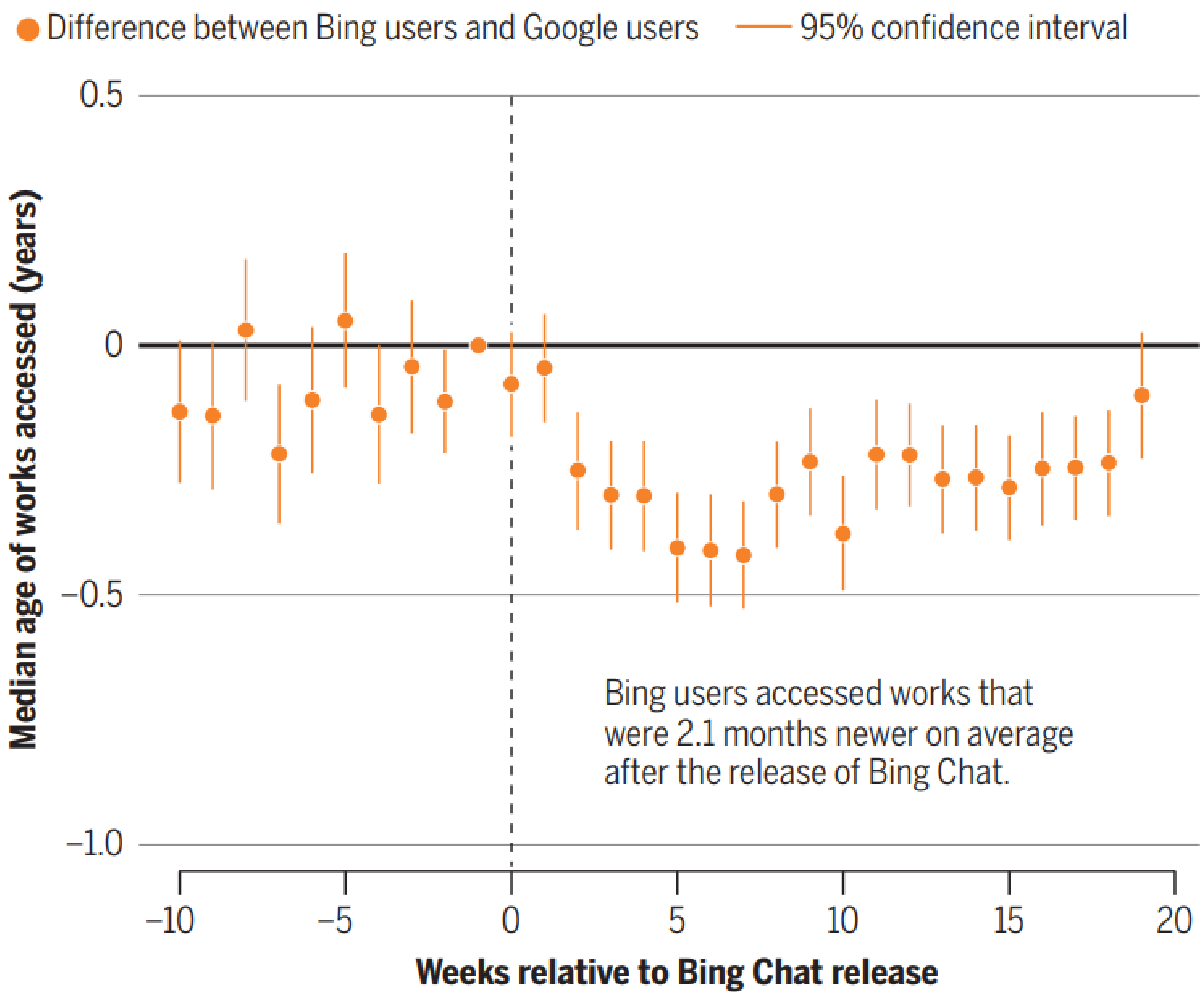
KEIGO KUSUMEGI, XINYU YANG, PAUL GINSPARG, MATHIJS DE VAAN, TOBY STUART, AND YIAN YIN Authors Info & Affiliations

SCIENCE • 18 Dec 2025 • Vol 390, Issue 6779 • pp. 1240-1243 • DOI: 10.1126/science.adw3000

Impact on Scientific Literature Search

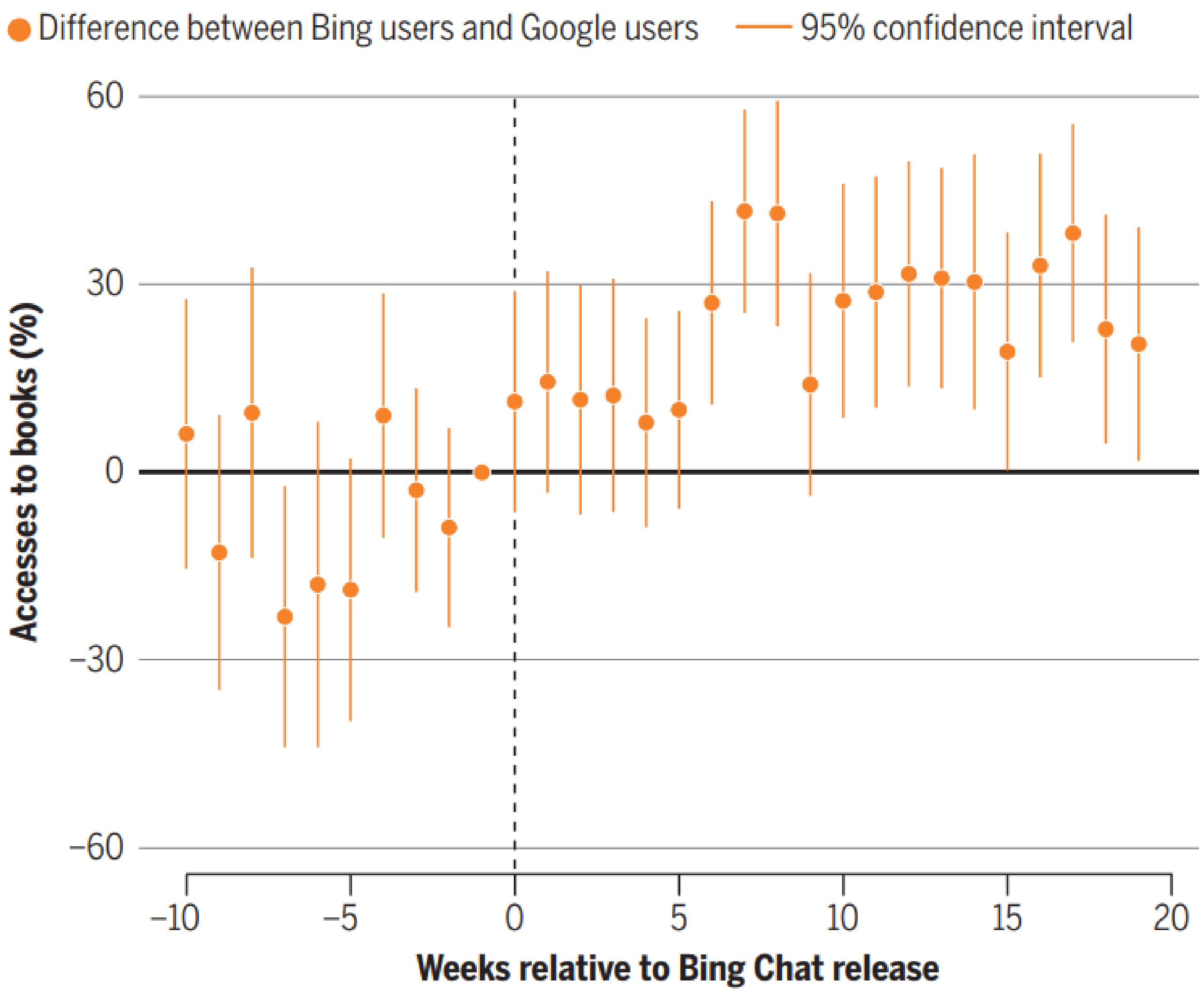
- The authors from Cornell had full access to arXiv data.
- They studied arXiv documents accessed by Bing/Google users (i.e., redirected from a Bing/Google webpage) before and after February 2023.

- Observation 1:The LLM-enhanced search engine guided users toward more recent work.



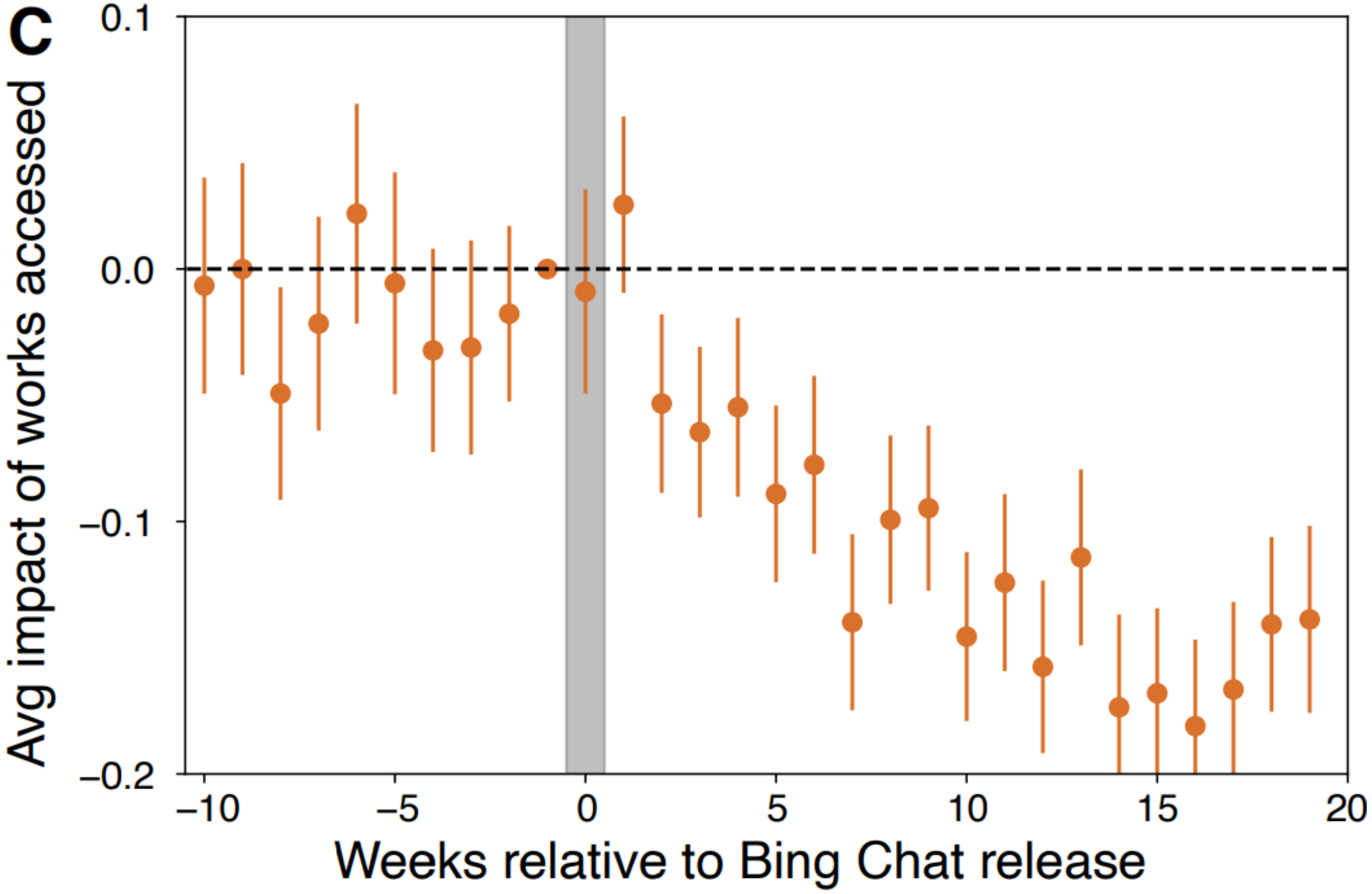
Impact on Scientific Literature Search

- Observation 2: The LLM-enhanced search engine guided users toward more books (i.e., more diverse knowledge sources).



Impact on Scientific Literature Search

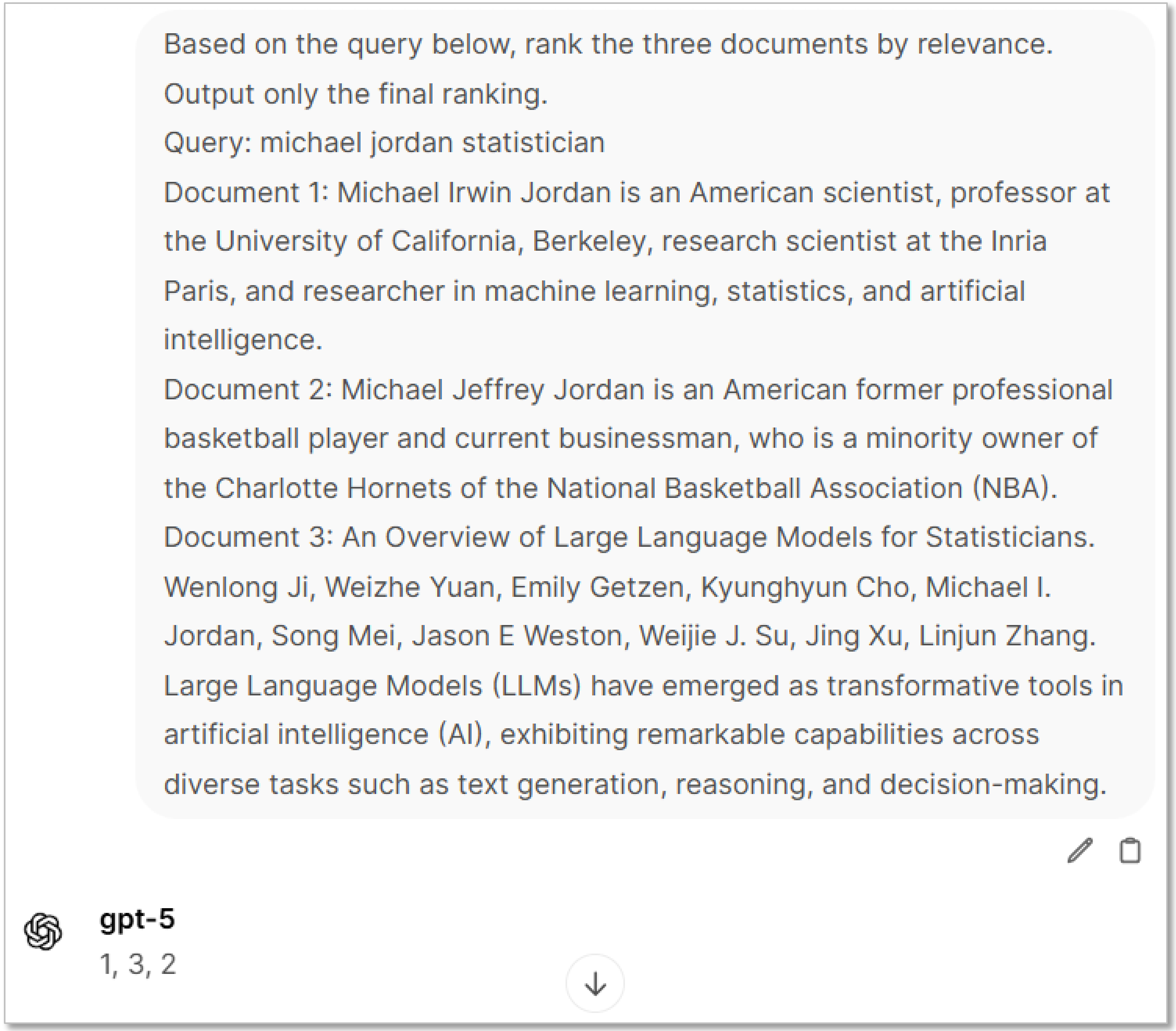
- Observation 3: The LLM-enhanced search engine guided users toward papers that have accumulated fewer citations.



How do LLMs help search engines rank documents?

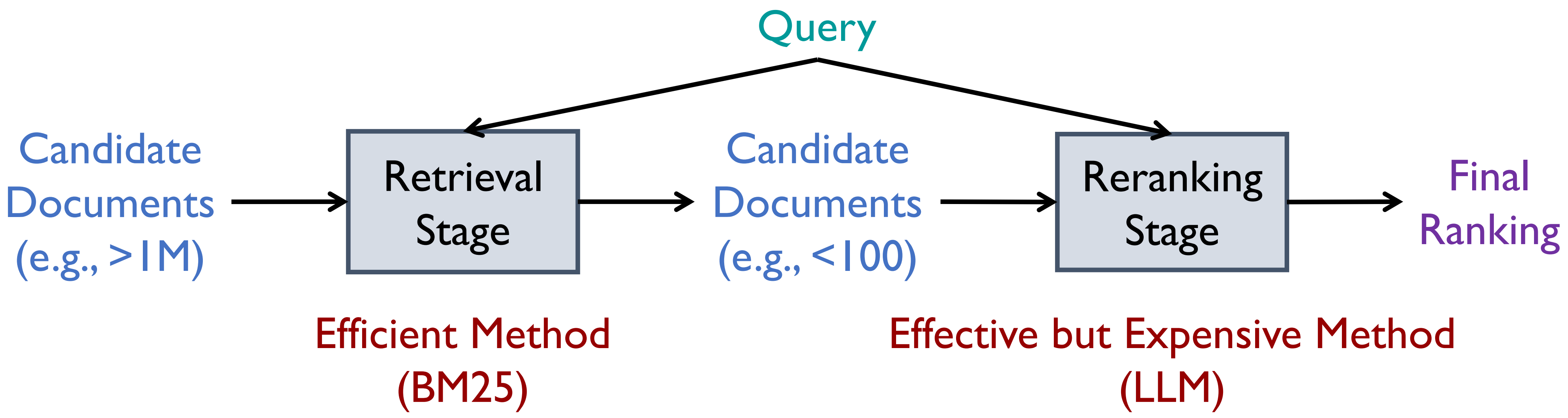
- Initial Idea: Feed the query and all candidate documents to the LLM, and let it output the ranking

- Is this strategy generalizable?



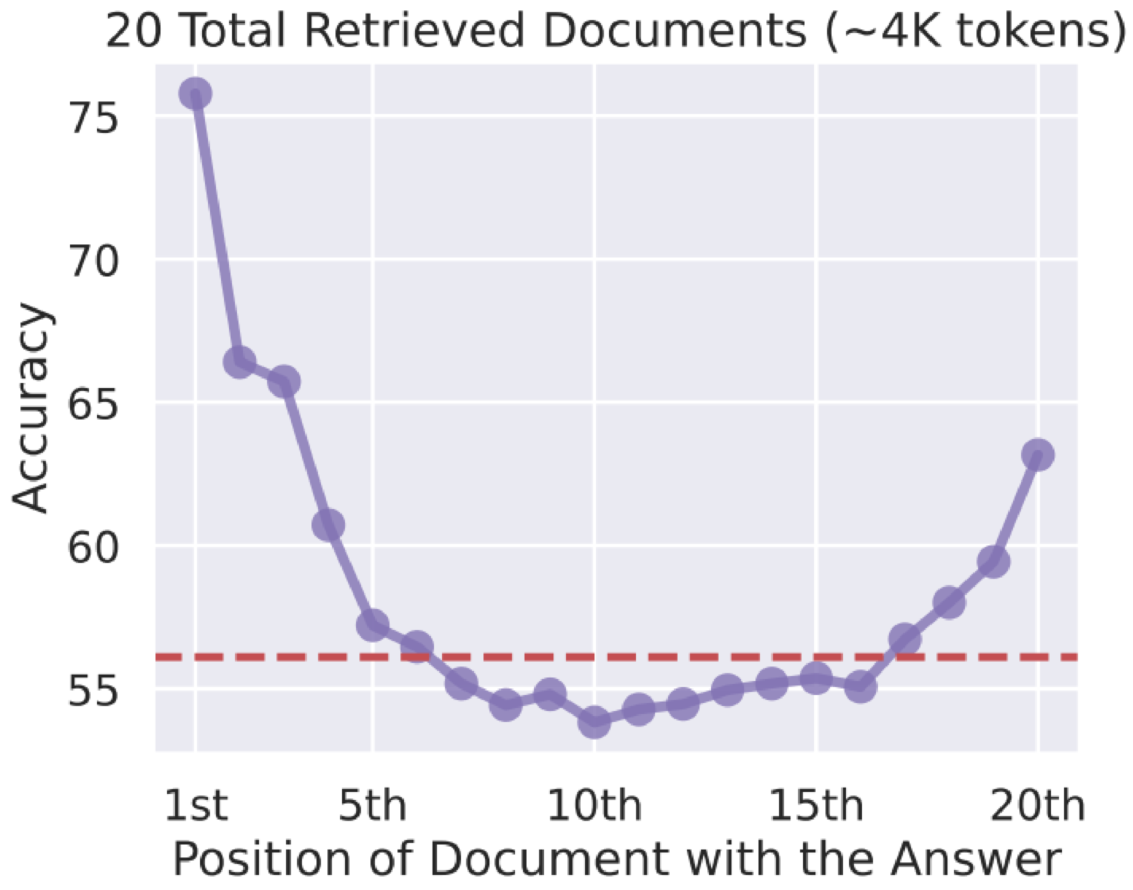
How do LLMs help ranking?

- Initial Idea: Feed the query and all candidate documents to the LLM, and let it output the ranking
- Can we generalize this idea to 1,000 or 1,000,000 candidate documents?
  - No! Because of the maximum input length limit, prohibitive costs, …
  - How to address this?
    - The retrieval-reranking paradigm

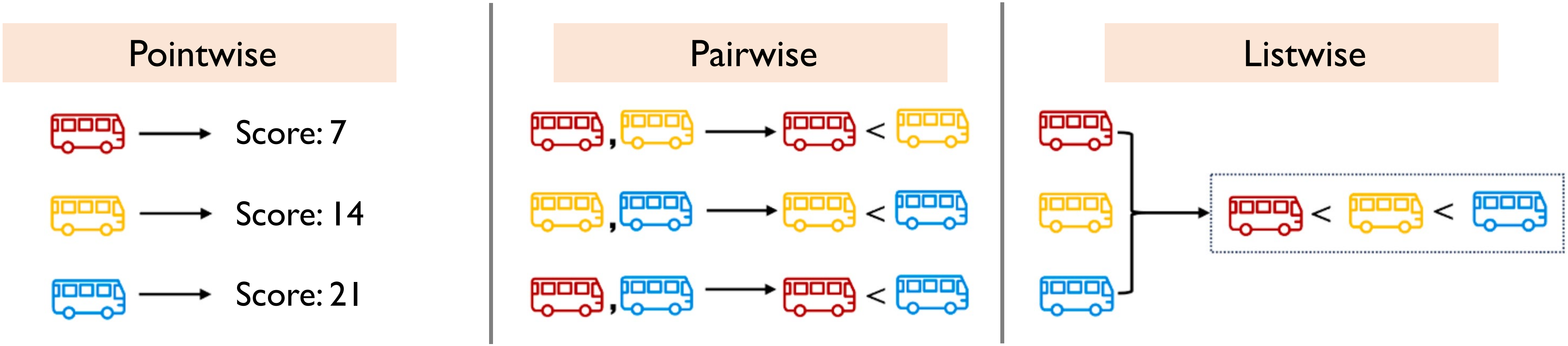


How do LLMs help ranking?

- Initial Idea: Feed the query and all candidate documents to the LLM, and let it output the ranking
- Can we generalize this idea to 20 candidate documents?
  - Not entirely! Because LLMs "lost in the middle" when the context is long
  - If LLMs are more powerful with no more than 5 candidate documents, how can we make it rank 20 documents?



Background: Three Types of Ranking Strategies



- Final Output:

Pointwise

- Given a query q and a candidate document d, directly calculate $\text{score}(q, d)$
- Examples?
  - TF-IDF
  - BM25
  - SVM [Nallapati, SIGIR 2004] in the Learning to Rank lecture
  - Dual Embedding Space Model [Nalisnick et al., WWW 2016] in the Neural Ranking lecture
  - Cross-Encoder [Dai and Callan, SIGIR 2019] in the BERT-Based Ranking lecture
  - …

Pairwise

- Given a query q and two candidate documents d_1, d_2, predict if d_1 should be ranked higher than d_2
- Examples?
  - Ranking SVM [Joachims, KDD 2002] in the Learning to Rank lecture
  - RankNet [Burges et al., ICML 2005] in the Learning to Rank lecture
  - Dense Passage Retrieval [Karpukhin et al., EMNLP 2020] in the Neural Ranking lecture
  - ColBERT [Khattab and Zaharia, SIGIR 2020] in the BERT-Based Ranking lecture
  - …

Listwise

- Given a query q and ALL candidate documents $d_1, d_2, \ldots, d_N$, output the ranking list of $d_1, d_2, \ldots, d_N$
- Examples
  - Our initial idea using ChatGPT
  - LambdaRank [Burges et al., NIPS 2006] can be either pairwise or listwise
  - LambdaMART [Burges, 2010] can be either pairwise or listwise
  - …

How to apply these three strategies when an LLM is available?

Pointwise Ranking w/ LLMs: Method 1 [Liang et al., arXiv 2022]

- Given a query q and a candidate document d, directly calculate $\mathrm{score}(q, d)$

Passage: {{passage}}
Query: {{query}}
Does the passage answer the query?

Yes (or No)

The LLM's output will be binary, but you need a continuous score for ranking. What can you do?

Given a passage and a query, predict whether the passage includes an answer to the query by producing either 'Yes' or 'No'.

Passage: {{passage}}
Query: {{query}}
Does the passage answer the query?
Answer:

Pointwise Ranking w/ LLMs: Method 1 [Liang et al., arXiv 2022]

- Check the model's probabilities for outputting the "*Yes*" or "*No*" token!
- If the model outputs "*Yes*", then $\mathrm{score}(q, d) = 1 + p(\text{“Yes”})$
- If the model outputs "*No*", then $\mathrm{score}(q, d) = 1 - p(\text{“No”})$

- Limitation?
  - Rely on the output probabilities of an LLM, which was unavailable for many closed-source LLM APIs
  - Currently, ChatGPT's web interface still does not show token probabilities, but you can get them via the OpenAI Responses API

json   Copy code

```json
"logprobs": true,
"top_logprobs": 5
```

Pointwise Ranking w/ LLMs: Method 2 [Sachan et al., EMNLP 2022]

- Given a query q and a candidate document d, directly calculate $\mathrm{score}(q, d)$

Please write a question based on this passage.
Passage: {{passage}}
Query:

{{query}}

The relevance between a query and a document is measured by the probability of the model to generate the query based on the document.

Please write a question based on this passage.
Passage: {{passage}}
Question: {{query}}

- How to calculate this probability?

Pointwise Ranking w/ LLMs: Method 2 [Sachan et al., EMNLP 2022]

- How to calculate this probability?

$$\prod_t p(q_t | i, d, q_{<t})$$

  - i: instruction; d: document; $q_{<t}$: the first $(t-1)$ tokens in the query
  - Or equivalently,

$$\text{score}(q, d) = \frac{1}{|q|} \sum_t \log p(q_t | i, d, q_{<t})$$

```
Please write a question based on this passage.
Passage: {{passage}}
Question: {{query}}
```

- Still rely on the output **probabilities** of an LLM
- Can we rely on the output tokens only?

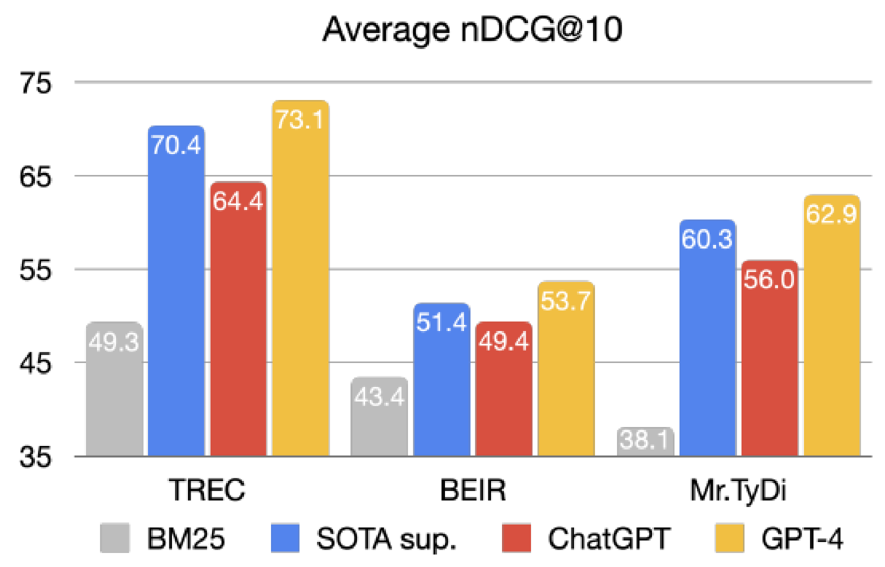
Listwise [Sun et al., EMNLP 2023]

Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents

Weiwei Sun[1*] **Lingyong Yan**[2] **Xinyu Ma**[2] **Shuaiqiang Wang**[2]
Pengjie Ren[1] **Zhumin Chen**[1] **Dawei Yin**[2†] **Zhaochun Ren**[3†]
[1]Shandong University, Qingdao, China [2]Baidu Inc., Beijing, China
[3]Leiden University, Leiden, The Netherlands
{sunnweiwei,lingyongy,xinyuma2016,shqiang.wang}@gmail.com
{renpengjie,chenzhumin}@sdu.edu.cn yindawei@acm.org
z.ren@liacs.leidenuniv.nl

Abstract

Large Language Models (LLMs) have demonstrated remarkable zero-shot generalization across various language-related tasks, including search engines. However, existing work utilizes the generative ability of LLMs for Information Retrieval (IR) rather than direct passage ranking. The discrepancy between the pre-training objectives of LLMs and the ranking objective poses another challenge. In this paper, we first investigate generative LLMs such



Listwise Ranking w/ LLMs

- Given a query q and ALL candidate documents $d_1, d_2, \ldots, d_N$, output the ranking list of $d_1, d_2, \ldots, d_N$

Instruction used for text-davinci-003

This is RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query.

The following are `{{num}}` passages, each indicated by number identifier []. I can rank them based on their relevance to query: `{{query}}`

[1] `{{passage_1}}`

[2] `{{passage_2}}`

(more passages) ...

The search query is: `{{query}}`

I will rank the `{{num}}` passages above based on their relevance to the search query. The passages will be listed in descending order using identifiers, and the most relevant passages should be listed first, and the output format should be [] > [] > etc, e.g., [1] > [2] > etc.

The ranking results of the `{{num}}` passages (only identifiers) is:

Instruction used for gpt-3.5-turbo and gpt-4

system:
You are RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query.

user:
I will provide you with `{{num}}` passages, each indicated by number identifier []. Rank them based on their relevance to query: `{{query}}`.

assistant:
Okay, please provide the passages.

user:
[1] `{{passage_1}}`

assistant:
Received passage [1]

user:
[2] `{{passage_2}}`

assistant:
Received passage [2]

(more passages) ...

user
Search Query: `{{query}}`.
Rank the `{{num}}` passages above based on their relevance to the search query. The passages should be listed in descending order using identifiers, and the most relevant passages should be listed first, and the output format should be [] > [], e.g., [1] > [2]. Only response the ranking results, do not say any word or explain.

Listwise Ranking w/ LLMs

- Given a query q and ALL candidate documents $d_1, d_2, \ldots, d_N$, output the ranking list of $d_1, d_2, \ldots, d_N$

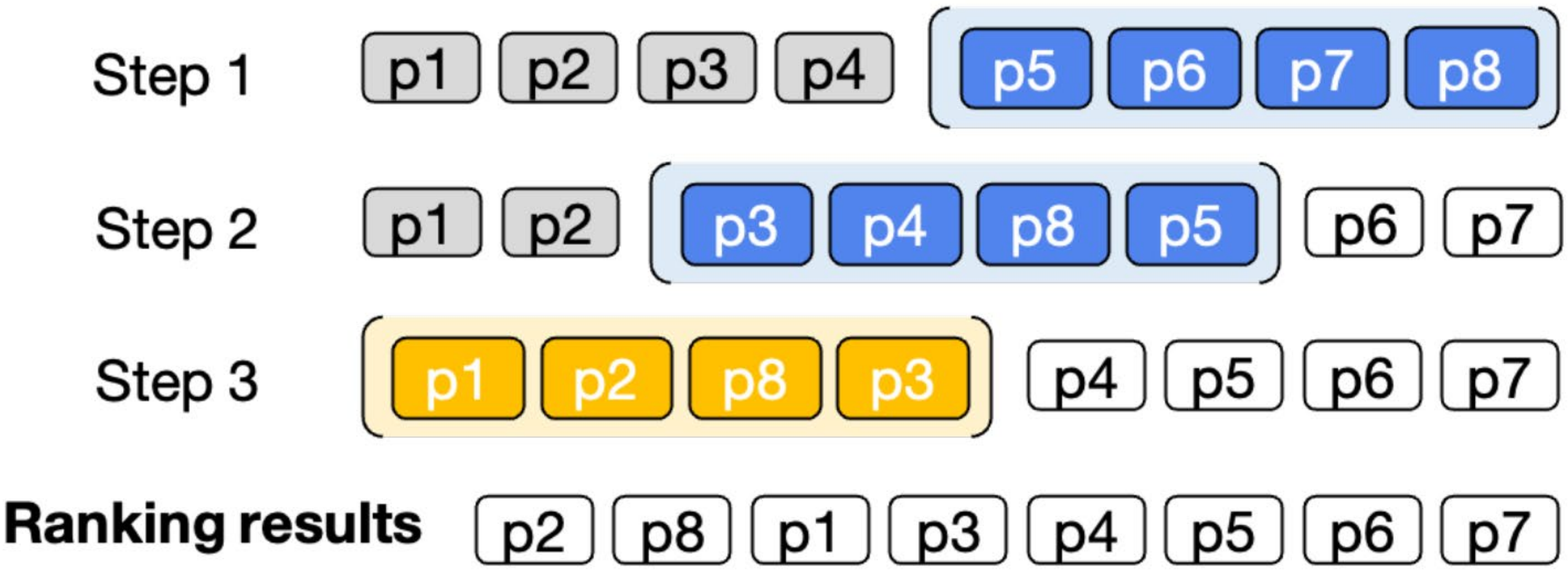
```
The following are passages related to query {{query}}
[1] {{passage_1}}
[2] {{passage_2}}
(more passages)
Rank these passages based on their relevance to the query.
```

```
[2] > [3] > [1] > [...]
```

- Previous unresolved issue: If LLMs are more powerful with no more than 5 candidate documents, how can we make it rank 20 documents?

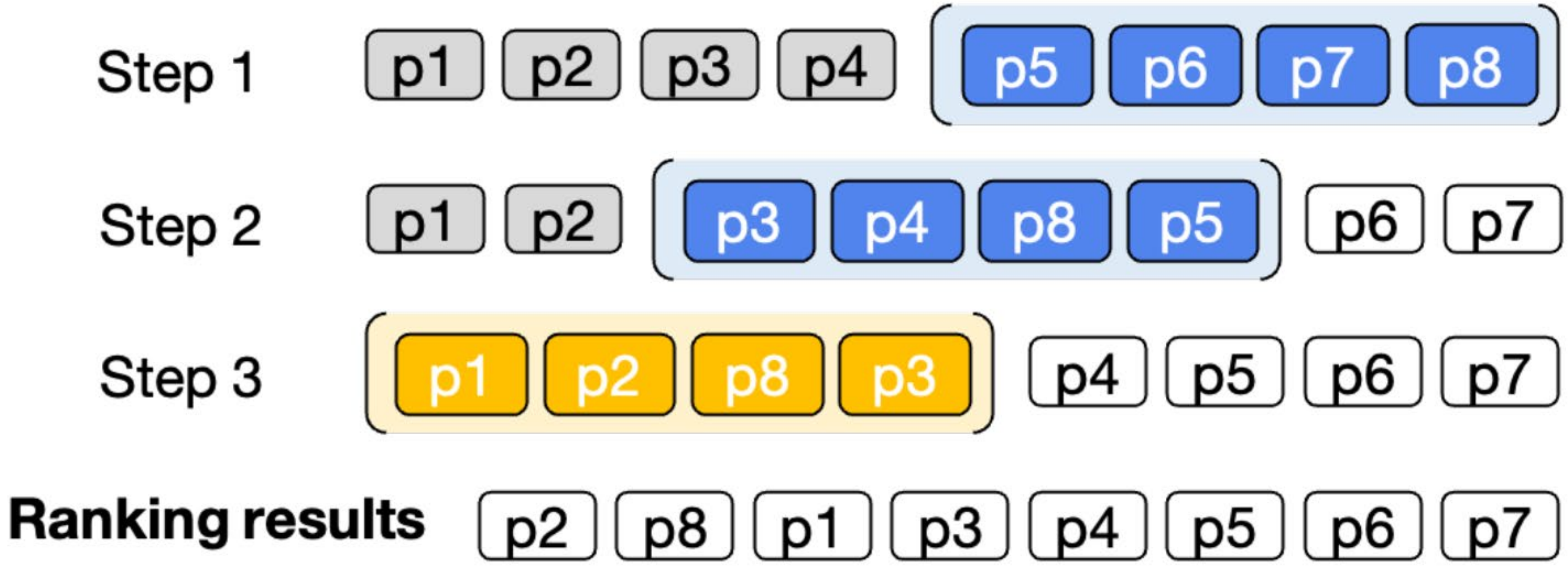
Sliding Window

- The LLM ranks only w documents at a time (e.g., $w = 4$)
- The LLM ranks all candidate in a back-to-first order using a sliding window
- The top-s documents in the previous window will participate in the next window (e.g., $s = 2$)



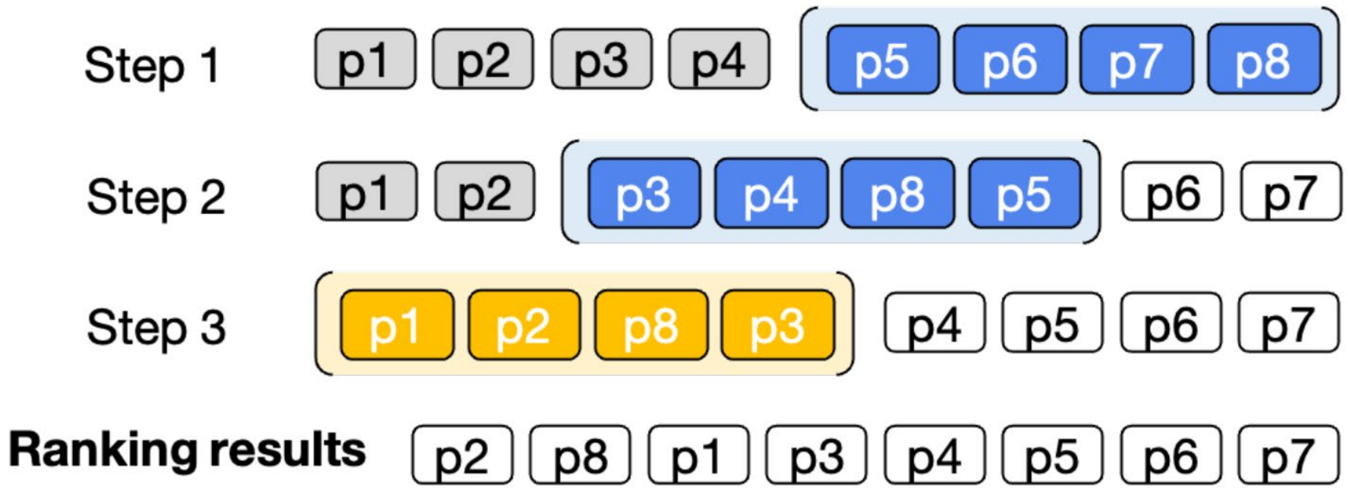
Sliding Window

- Assume there are N documents to be reranked. How many times do you need to call the LLM?
  - $\left\lceil \frac{N-w}{w-s} \right\rceil + 1$ (e.g., $\left\lceil \frac{8-4}{4-2} \right\rceil + 1 = 3$)



Sliding Window

- Why back-to-first?
  - Ensure that the top portion of the ranking list is more accurate, while the ordering toward the end is relatively less important
- How many of the top-ranked documents can this method guarantee to be absolutely accurate? (Assume the LLM is perfect at ranking)
  - Top-s



Prediction Failures of Listwise Ranking w/ LLMs

- Missing: When LLMs only outputs a partial list of the input documents
  - E.g., [2] > [4] > [1]
- Rejection: LLMs refuse to perform the ranking task and produce irrelevant outputs
- Repetition: LLMs output the same document more than once
  - E.g., [2] > [4] > [3] > [1] > [4]
- Inconsistency: The same list of documents have different output rankings when they are fed in with different order or context

Pairwise [Qin et al., NAACL 2024]

Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, Michael Bendersky
Google Research
{zhenqin,jagerman,kaihuibj,hlz,junru,lyyanle,jmshen,tianqiliu,jialu,metzler,xuanhui,bemike}@google.com

Abstract

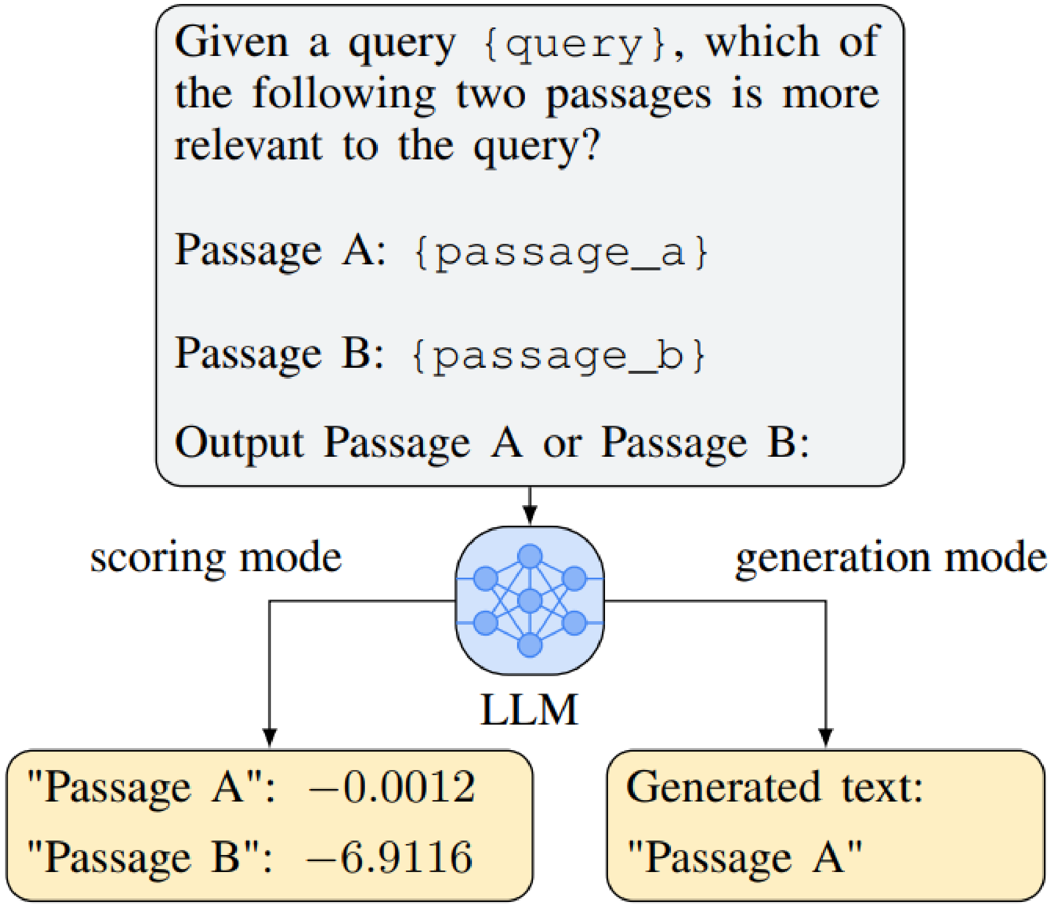
Ranking documents using Large Language Models (LLMs) by directly feeding the query and candidate documents into the prompt is an interesting and practical problem. However, researchers have found it difficult to outperform fine-tuned baseline rankers on benchmark datasets. We analyze pointwise and listwise ranking prompts used by existing methods and argue that off-the-shelf LLMs do not fully understand these challenging ranking formulations. In this paper, we propose to sig-

potentially trained with millions of labeled examples, even in the zero-shot setting (Kojima et al., 2022; Agrawal et al., 2022; Huang et al., 2022; Hou et al., 2023).

However, there is limited success for the important text ranking problem using off-the-shelf LLMs (Ma et al., 2023). Existing results usually significantly underperform well-trained baseline rankers (e.g., Nogueira et al. (2020); Zhuang et al. (2023)). The only exception is a recent approach proposed by Sun et al. (2023b), which depends on

Pairwise Ranking w/ LLMs

- Given a query q and two candidate documents d_1, d_2, predict if d_1 should be ranked higher than d_2
- Scoring mode: Compare the output probabilities p(“*Passage A*”) and p(“*Passage B*”)
- Generation mode: Directly check the output tokens

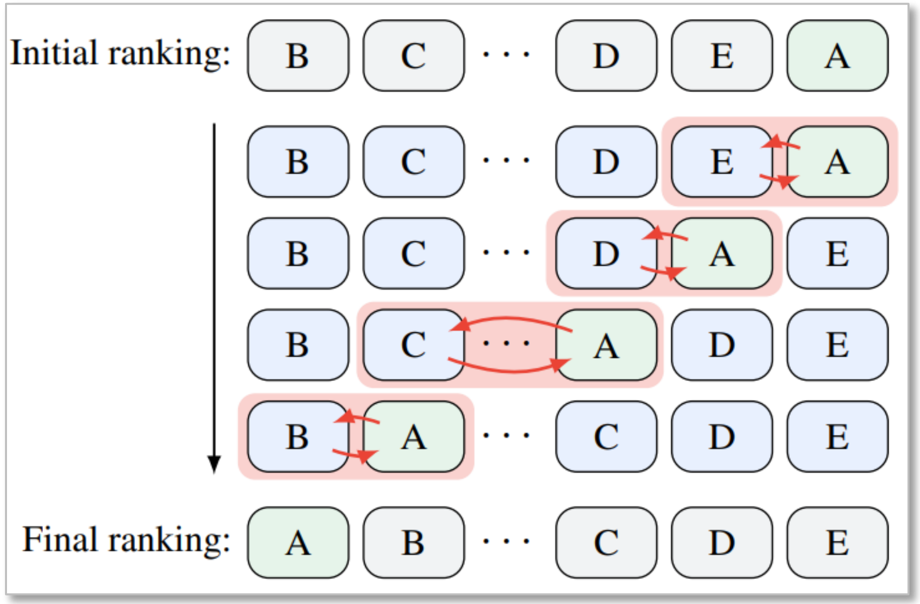


Pairwise Ranker → Ranking for the Entire List

- All pair comparisons (PRP-Allpair): $O(N^2)$ API calls

$$\text{score}(q, d) = 1 \times \sum_{d' \neq d} \mathbb{I}(d \succ d') + 0.5 \times \sum_{d' \neq d} \mathbb{I}(d = d')$$

- Sorting-based (PRP-Sorting): $O(N \log N)$ API calls
  - Quicksort or Heapsort
- Sliding window (PRP-Sliding): $O(N)$ API calls
  - Back-to-first
  - One-pass Bubblesort
  - k-pass Bubblesort (k is small)



Pairwise Ranker → Ranking for the Entire List

- Sliding window (PRP-Sliding): $O(N)$ API calls
  - Back-to-first
  - k-pass Bubblesort (k is small)
  - Guarantee the top-k documents to be absolutely accurate (Assume the LLM is perfect at ranking)

- Comparison between Pointwise, Pairwise, and Listwise ranking w/ LLMs:

Method	# of LLM API Calls	Generation API	Scoring API
Pointwise	$O(N)$	No	Yes
Listwise	$O(N)$	Yes	No
Pairwise	$O(N^2), O(N \log N), O(N)$	Yes	Yes

Results on TREC

Method	LLM	Size	TREC-DL2019 NDCG@1	TREC-DL2019 NDCG@5	TREC-DL2019 NDCG@10	TREC-DL2020 NDCG@1	TREC-DL2020 NDCG@5	TREC-DL2020 NDCG@10
BM25	NA	NA	54.26	52.78	50.58	57.72	50.67	47.96
Supervised Methods								
monoBERT	BERT	340M	79.07	73.25	70.50	78.70	70.74	67.28
monoT5	T5	220M	79.84	73.77	71.48	77.47	69.40	66.99
monoT5	T5	3B	79.07	73.74	71.83	80.25	72.32	68.89
RankT5	T5	3B	79.07	75.66	72.95	80.86	73.05	69.63
Unsupervised LLM Methods								
LRL	text-davinci-003	175B	-	-	65.80	-	-	62.24
RankGPT	gpt-3	175B	50.78	50.77	49.76	50.00	48.36	48.73
RankGPT	text-davinci-003	175B	69.77	64.73	61.50	69.75	58.76	57.05
RankGPT	gpt-3.5-turbo	154B*	*82.17*	71.15	65.80	79.32	66.76	62.91
RankGPT	gpt-4	1T*	**82.56**	**79.16**	**75.59**	78.40	74.11	*70.56*
UPR	FLAN-T5-XXL	11B	62.79	62.07	62.00	64.20	62.05	60.34
RG	FLAN-T5-XXL	11B	67.05	65.41	64.48	65.74	66.40	62.58
UPR	FLAN-UL2	20B	53.10	57.68	58.95	64.81	61.50	60.02
RG	FLAN-UL2	20B	70.93	66.81	64.61	75.62	66.85	65.39
PRP-Allpair	FLAN-T5-XL	3B	74.03	71.73	69.75	79.01	72.22	68.12
PRP-Sorting	FLAN-T5-XL	3B	77.52	71.88	69.28	74.38	69.44	65.87
PRP-Sliding-10	FLAN-T5-XL	3B	75.58	71.23	68.66	75.62	69.00	66.59
PRP-Allpair	FLAN-T5-XXL	11B	72.09	71.28	69.87	82.41	74.16	69.85
PRP-Sorting	FLAN-T5-XXL	11B	74.42	69.62	67.81	72.53	71.28	67.77
PRP-Sliding-10	FLAN-T5-XXL	11B	64.73	69.49	67.00	75.00	70.76	67.35
PRP-Allpair	FLAN-UL2	20B	73.64	74.77	72.42	*85.19*	*74.73*	**70.68**
PRP-Sorting	FLAN-UL2	20B	74.42	73.60	71.88	84.57	72.52	69.43
PRP-Sliding-10	FLAN-UL2	20B	78.29	*75.49*	*72.65*	**85.80**	**75.35**	70.46

Results on BEIR

Method	LLM	Size	Covid	Touche	DBPedia	SciFact	Signal	News	Robust04	Avg
BM25	NA	NA	59.47	**44.22**	31.80	67.89	33.05	39.52	40.70	45.23
Supervised Methods										
monoBERT	BERT	340M	70.01	31.75	41.87	71.36	31.44	44.62	49.35	48.63
monoT5	T5	220M	78.34	30.82	42.42	73.40	31.67	46.83	51.72	50.74
monoT5	T5	3B	80.71	32.41	44.45	**76.57**	32.55	48.49	56.71	53.13
RankT5	T5	3B	82.00	37.62	44.19	76.86	31.80	48.15	52.76	53.34
TART-Rerank	T5	3B	75.10	27.46	42.53	74.84	25.84	40.01	50.75	48.08
Unsupervised LLM Methods										
UPR	FLAN-T5-XXL	11B	72.64	21.56	35.14	73.54	30.81	42.99	47.85	46.36
RG	FLAN-T5-XXL	11B	70.31	22.10	31.32	63.43	26.89	37.34	51.56	43.28
UPR	FLAN-UL2	20B	70.69	23.68	34.64	71.09	30.33	41.78	47.52	45.68
RG	FLAN-UL2	20B	70.22	24.67	30.56	64.74	29.68	43.78	53.00	45.24
RankGPT	gpt-3.5-turbo	154B	76.67	36.18	44.47	70.43	32.12	*48.85*	50.62	51.33
PRP-Allpair	FLAN-T5-XL	3B	81.86	26.93	44.63	73.25	32.08	46.52	54.02	51.33
PRP-Sorting	FLAN-T5-XL	3B	80.41	28.23	42.84	67.94	30.95	42.95	50.07	49.06
PRP-Sliding-10	FLAN-T5-XL	3B	77.58	*40.48*	44.77	73.43	**35.62**	46.45	50.74	52.72
PRP-Allpair	FLAN-T5-XXL	11B	79.62	29.81	41.41	*74.23*	32.22	47.68	**56.76**	51.67
PRP-Sorting	FLAN-T5-XXL	11B	78.75	29.61	39.23	70.10	31.28	44.68	53.01	49.52
PRP-Sliding-10	FLAN-T5-XXL	11B	74.39	41.60	42.19	72.46	35.12	47.26	52.38	52.20
PRP-Allpair	FLAN-UL2	20B	**82.30**	29.71	*45.94*	75.70	32.26	48.04	*55.49*	*52.78*
PRP-Sorting	FLAN-UL2	20B	*82.29*	25.80	44.53	67.07	32.04	45.37	51.45	49.79
PRP-Sliding-10	FLAN-UL2	20B	79.45	37.89	**46.47**	73.33	*35.20*	**49.11**	53.43	**53.55**

Summary: LLMs for Ranking

Please write a question based on this passage.
Passage: {{passage}}
Query:

{{query}}

Pointwise

Passage: {{passage}}
Query: {{query}}
Does the passage answer the query?

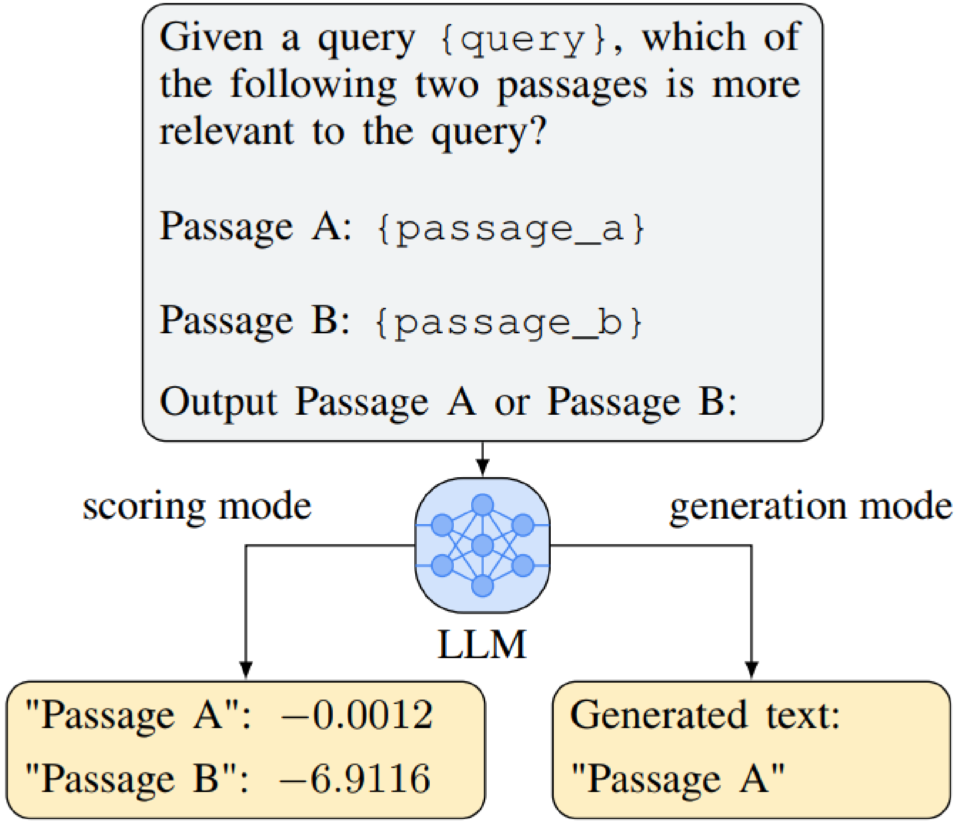
Yes (or No)

Pointwise

The following are passages related to query {{query}}
[1] {{passage_1}}
[2] {{passage_2}}
(more passages)
Rank these passages based on their relevance to the query.

[2] > [3] > [1] > [...]

Listwise



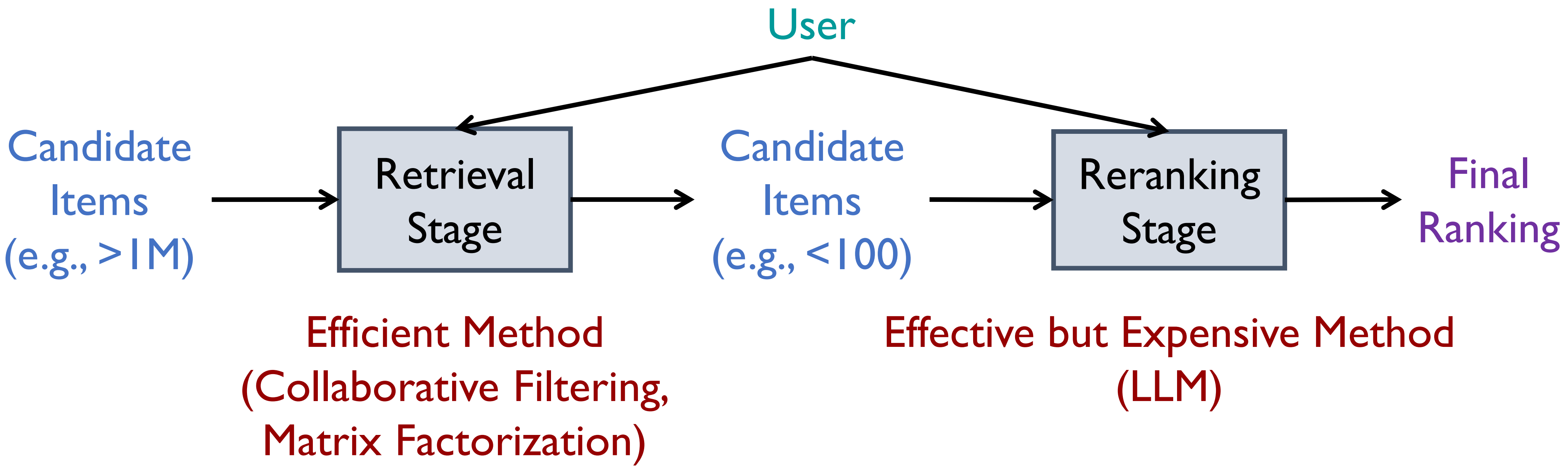
Pairwise

Takeaway Messages: LLMs for Ranking

- BM25 retrieval + LLM reranking is currently the best-performing paradigm for zero-shot retrieval in unseen domains
  - Ultimately, we outperform BM25 alone in the zero-shot setting!
- If you want to leverage the LLM's output probabilities, use either a pointwise or pairwise approach
- If you want to leverage the LLM's output tokens only, use either a pairwise or listwise approach
- Sliding window is an efficient and effective strategy. It is often not only faster than all-pair comparisons and full sorting but also surprisingly performs better
  - Possibly because LLMs are NOT perfect or consistent

Next: How do LLMs help recommender systems?

- The retrieval-reranking paradigm (again!)



- What is the difference between using LLMs for reranking in search vs. in recommendation?

Homework 4 (Part 2): ChatGPT/Gemini for Recommendation

Part 2. ChatGPT (50 points total)

For this part, we will explore the recommendation performance of either ChatGPT or Gemini:

- find two cases where ChatGPT (or Gemini) performs well,
- find two cases where ChatGPT (or Gemini) fails and analyze the reasons behind the failures.

You have great leeway in how you approach this problem. Is it a ratings prediction task? Or a ranking task? Or a pairwise comparison task? What kinds of items and what kinds of users? How should you prompt? These are all open questions.

Potential sources of inspiration here! [Liu et al., arXiv 2023]

Is ChatGPT a Good Recommender? A Preliminary Study

Junling Liu*
william.liuj@gmail.com
Alibaba Group
China

Chao Liu*
chize.lc@antgroup.com
Ant Group
China

Peilin Zhou*
zhoupalin@gmail.com
Hong Kong University of Science and Technology(Guangzhou)
China

Renjie Lv
lvrenjie.lrj@antgroup.com
Ant Group
China

Kang Zhou
kangbeyond89@163.com
Alibaba Group
China

Yan Zhang
yanbest0117@163.com
Alibaba Group
China

ABSTRACT

Recommendation systems have witnessed significant advancements and have been widely used over the past decades. However, most traditional recommendation methods are task-specific and therefore lack efficient generalization ability. Recently, the emergence of ChatGPT has significantly advanced NLP tasks by enhancing the capabilities of conversational models. Nonetheless, the application of ChatGPT in the recommendation domain has not been thoroughly investigated. In this paper, we employ ChatGPT as a general-purpose recommendation model to explore its potential for transferring extensive linguistic and world knowledge acquired from large-scale corpora to recommendation scenarios. Specifically, we design a set of prompts and evaluate ChatGPT's performance

KEYWORDS

Large-Language Model, ChatGPT, Recommendation System

ACM Reference Format:
Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is ChatGPT a Good Recommender? A Preliminary Study. In *The 1st workshop on recommendation with generative models, October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As a crucial technique for addressing information overload and enhancing user experience, recommendation systems have witnessed significant advancements over the past decade and have

A Naïve Idea

- Pointwise: Feed the user and each candidate item to the LLM, and let it output the predicted rating

How will User 4403 rate Item 6212? (1 being lowest and 5 being highest)

gpt-5

I don't have enough information to predict that rating. Please provide:

- User 4403's past ratings (or preferences)
- Other users' ratings for Item 6212 (or item features)
- Optionally the dataset's global mean rating

If you need a naive, data-free baseline, the neutral midpoint is 3.

We need to describe the item and the user!

Incorporating Item Content and User Rating History

Here is the user's rating history:

Bundle Monster 100 PC 3D Designs Nail Art Nailart Manicure Fimo Canes Sticks Rods Stickers Gel Tips — 5.0

Winstonia's Double-Ended Nail Art Marbling Dotting Tool Pen Set with 10 Different Sizes (5 Colors) — 3.0

Nail Art Jumbo Stamp Stamping Manicure Image Plate 2: Tropical Holiday by Cheeky® — 2.0

Nail Art Jumbo Stamp Stamping Manicure Image Plate 6: Happy Holidays by Cheeky® — 2.0

Based on the above rating history, please predict the user's rating for the product: "SHANY Nail Art Set (24 Famous Colors Nail Art Polish, Nail Art Decoration)"

(1 being the lowest and 5 being the highest.)

gpt-5

4.0

Method	RMSE
MF	1.1973
MLP	1.3078
LLM prompting (gpt-3.5-turbo)	1.0751

Incorporating Item Content and User Rating History

Here is the user's rating history:
Bundle Monster 100 PC 3D Designs Nail Art Nailart Manicure Fimo Canes Sticks Rods Stickers Gel Tips — 5.0
Winstonia's Double-Ended Nail Art Marbling Dotting Tool Pen Set with 10 Different Sizes (5 Colors) — 3.0
Nail Art Jumbo Stamp Stamping Manicure Image Plate 2: Tropical Holiday by Cheeky® — 2.0
Nail Art Jumbo Stamp Stamping Manicure Image Plate 6: Happy Holidays by Cheeky® — 2.0
Based on the above rating history, please predict the user's rating for the product: "SHANY Nail Art Set (24 Famous Colors Nail Art Polish, Nail Art Decoration)"
(1 being the lowest and 5 being the highest.)

gpt-5
4.0

- How should we understand this solution?
- Content-based approach: We are having the LLM infer the user's profile from items the user has already rated, BUT this inference process is not based on a formula we explicitly wrote in advance

Incorporating Item Content and User Rating History

Here is the user's rating history:

Bundle Monster 100 PC 3D Designs Nail Art Nailart Manicure Fimo Canes Sticks Rods Stickers Gel Tips — 5.0

Winstonia's Double-Ended Nail Art Marbling Dotting Tool Pen Set with 10 Different Sizes (5 Colors) — 3.0

Nail Art Jumbo Stamp Stamping Manicure Image Plate 2: Tropical Holiday by Cheeky® — 2.0

Nail Art Jumbo Stamp Stamping Manicure Image Plate 6: Happy Holidays by Cheeky® — 2.0

Based on the above rating history, please predict the user's rating for the product: "SHANY Nail Art Set (24 Famous Colors Nail Art Polish, Nail Art Decoration)" (1 being the lowest and 5 being the highest.)

gpt-5

4.0

- If the user has already rated 1,000 items, and it is not possible to feed all of them into the LLM, what can be done?
- First select the 10 items that are most similar to the candidate item (e.g., using the BERT embedding of the product title)

Incorporating Item Content and User Rating History

Here is the user's rating history:

Bundle Monster 100 PC 3D Designs Nail Art Nailart Manicure Fimo Canes Sticks Rods Stickers Gel Tips — 5.0

Winstonia's Double-Ended Nail Art Marbling Dotting Tool Pen Set with 10 Different Sizes (5 Colors) — 3.0

Nail Art Jumbo Stamp Stamping Manicure Image Plate 2: Tropical Holiday by Cheeky® — 2.0

Nail Art Jumbo Stamp Stamping Manicure Image Plate 6: Happy Holidays by Cheeky® — 2.0

Based on the above rating history, please predict the user's rating for the product: "SHANY Nail Art Set (24 Famous Colors Nail Art Polish, Nail Art Decoration)" (1 being the lowest and 5 being the highest.)

gpt-5

4.0

- How should we understand this solution?
- Collaborative Filtering: We are having the LLM aggregate the (user, candidate item) rating from the (user, similar item) ratings, BUT this aggregation process is not based on a formula we explicitly wrote in advance.

What if we only have implicit feedback?

- Pointwise supervision is no longer available
- Listwise: Feed the user (described by the items the user has interacted with) and all candidate items (described by their content) to the LLM, and let it output the ranking

- Instruction
- User interaction history (a list of product titles, without any particular order)
- Candidate items

Requirements: Select 10 items to recommend and rank them in order of priority, from highest to lowest. Output format: a Python list. Do not include any explanations or additional text.
The user has interacted with the following items (in no particular order): ["Skin Obsession Jessner's Chemical Peel Kit Anti-aging and Anti-acne Skin Care Treatment", "Xtreme Brite Brightening Gel 1oz", ..., "Reviva - Light Skin Peel, 1.5 oz cream"]. From the candidate list below, choose the top 10 items to recommend to the user and rank them from highest to lowest priority.
Candidates: ["Rogaine for Women Hair Regrowth Treatment (3 × 2 oz bottles)", "Best Age Spot Remover", ..., "L'Oreal Kids Extra Gentle 2-in-1 Shampoo with a Burst of Cherry Almond, 9.0 Fluid Ounces"].

What if we only have implicit feedback?

- Can I create an in-context demonstration example?

- Instruction
- User interaction history (hold out one item as a demonstration example)
- In-context demonstration example (the held-out item + a dummy candidate pool containing items the user has not interacted with)
- Candidate items

Requirements: Select 10 items to recommend and rank them in order of priority, from highest to lowest. Output format: a Python list. Do not include any explanations or additional text.
The user has interacted with the following items (in no particular order): ['Maybelline New York Eye Studio Lasting Drama Gel Eyeliner, Eggplant 956, 0.106 Ounce', "L'Oreal Paris Healthy Look Hair Color, 8.5 Blonde/White Chocolate", ..., 'Duo Lash Adhesive, Clear, 0.25 Ounce']. Given that the user has interacted with 'WAWO 15 Color Professional Makeup Eyeshadow Camouflage Facial Concealer Neutral Palette' from a candidate pool: ['MASH Bamboo Reusable Cuticle Pushers Remover / Manicure Pedicure Stick', 'Urban Decay All Nighter Long-Lasting Makeup Setting Spray 4 oz', ..., 'Classic Cotton Balls Jumbo Size, 100 Count'], select the top 10 items to recommend from a new candidate pool: ['Neutrogena Ultra Sheer Sunscreen SPF 45 Twin Pack 6.0 Ounce', 'Blinc Eyeliner Pencil - Black', ..., 'Skin MD Natural + SPF15 combines the benefits of a shielding lotion and a sunscreen lotion']. Note that the items in the new candidate pool are not ordered.

What if we want to perform sequential recommendation?

- Simple version:
  - Instruction
  - User interaction history (a list of product titles, in chronological order)
  - Candidate items
- Create an in-context demonstration example:
  - Instruction
  - User interaction history (hold out the last item as a demonstration example)
  - In-context demonstration example (the held-out item + a dummy candidate pool containing items the user has not interacted with)
  - Candidate items

Performance

- LLMs still underperform non-LLM baselines (fully supervised) by a notable margin!

Recommendation with Implicit Feedback

Method	nDCG@5	nDCG@10
BPR-MF	0.0857	0.1224
BPR-MLP	0.0848	0.1225
LLM prompting (gpt-3.5-turbo)	0.0216	0.0398

Sequential Recommendation

Method	nDCG@5	nDCG@10
SASRec	0.0249	0.0318
S^3-Rec	0.0244	0.0327
LLM prompting (gpt-3.5-turbo)	0.0135	0.0135

Hou et al. [ECIR 2024]

Large Language Models are Zero-Shot Rankers for Recommender Systems

Yupeng Hou[1,2], Junjie Zhang[1], Zihan Lin[3], Hongyu Lu[4], Ruobing Xie[4], Julian McAuley[2], and Wayne Xin Zhao[1(✉)]

[1] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
yphou@ucsd.edu, junjie.zhang@ruc.edu.cn, batmanfly@gmail.com
[2] UC San Diego, San Diego, USA
[3] School of Information, Renmin University of China, Beijing, China
[4] WeChat, Tencent, Shenzhen, China

Abstract. Recently, large language models (LLMs) (*e.g.,* GPT-4) have demonstrated impressive general-purpose task-solving abilities, including the potential to approach recommendation tasks. Along this line of research, this work aims to investigate the capacity of LLMs that act as the ranking model for recommender systems. We first formalize the recommendation problem as a conditional ranking task, considering sequential interaction histories as *conditions* and the items retrieved by

Prompting Strategies

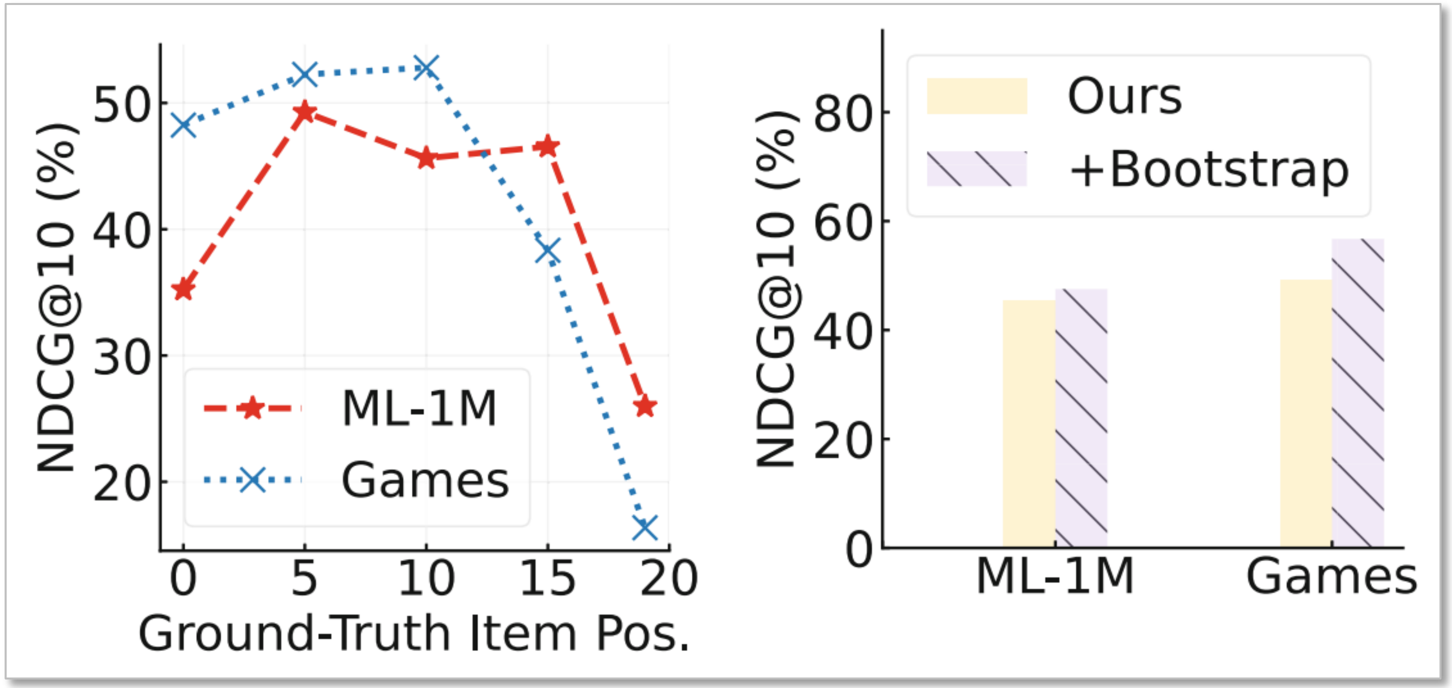
- Sequential prompting: Arrange the historical interactions in chronological order (already introduced in [Liu et al., arXiv 2023])
  - *"I've watched the following movies in the past in order: '0. Multiplicity', '1. Jurassic Park', ..."*
- In-context learning: Holding out the last item in the interaction history as a demonstration example (already introduced in [Liu et al., arXiv 2023])
  - *"If I've watched the following movies in the past in order: '0. Multiplicity', '1. Jurassic Park', ..., then you should recommend Dead Presidents to me and now that I've watched Dead Presidents, then ..."*
- Recency-focused prompting: In addition to the sequential interaction records, we can add an additional sentence to emphasize the most recent interaction
  - *"I've watched the following movies in the past in order: '0. Multiplicity', '1. Jurassic Park', ... Note that my most recently watched movie is Dead Presidents. ..."*

Performance

	Method	ML-1M				Games			
		N@1	N@5	N@10	N@20	N@1	N@5	N@10	N@20
full	Pop	22.91	45.16	52.33	55.36	28.35	47.42	52.96	57.45
	BPRMF [49]	34.60	59.87	64.29	65.39	44.92	62.33	66.27	68.94
	SASRec [33]	61.39	76.39	78.89	79.79	56.90	73.19	75.92	77.14
zero-shot	BM25 [50]	4.70	12.68	17.88	33.19	13.92	28.81	34.61	44.35
	UniSRec [30]	7.37	18.80	26.67	37.93	18.95	33.99	40.71	48.42
	VQ-Rec [29]	5.98	15.48	23.74	35.85	7.28	18.28	26.21	37.62
	Sequential	18.28	36.35	42.85	49.02	30.28	45.48	50.57	56.55
	Recency-Focused	19.57	37.73	44.23	50.01	**34.03**	**48.77**	**53.50**	**59.01**
	In-Context Learning	**21.77**	**39.59**	**45.83**	**51.62**	33.95	48.44	53.10	58.92

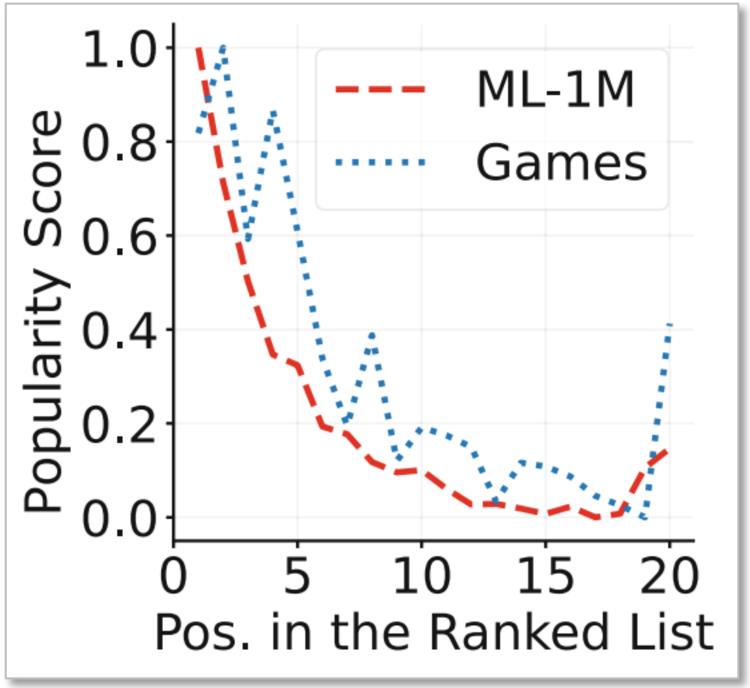
LLMs suffer from position bias

- The ranking performance drops significantly when the ground-truth items appear at the last few positions
- Solution in this paper: Rank the candidate set repeatedly for multiple times, with candidates randomly shuffled at each round. Then merge the results of each round to derive the final ranking



LLMs suffer from popularity bias

- Popular items tend to be ranked at higher positions by LLMs
- Why?
  - For popular items, the associated text may also appear frequently in the pre-training corpora of LLMs
  - E.g., a best-selling book would be widely discussed on the Web



Fine-Tuning a Language Model for Recommendation

Recformer [Li et al., KDD 2023]

Text Is All You Need: Learning Language Representations for Sequential Recommendation

Jiacheng Li
University of California, San Diego
j9li@eng.ucsd.edu

Ming Wang
Amazon, United States
mingww@amazon.com

Jin Li
Amazon, United States
jincli@amazon.com

Jinmiao Fu
Amazon, United States
jinnmiaof@amazon.com

Xin Shen
Amazon, United States
xinshen@amazon.com

Jingbo Shang
University of California, San Diego
jshang@eng.ucsd.edu

Julian McAuley
University of California, San Diego
jmcauley@eng.ucsd.edu

ABSTRACT

Sequential recommendation aims to model dynamic user behavior from historical interactions. Existing methods rely on either explicit item IDs or general textual features for sequence modeling to understand user preferences. While promising, these approaches still struggle to model cold-start items or transfer knowledge to new datasets. In this paper, we propose to model user preferences and item features as language representations that can be generalized to new items and datasets. To this end, we present a novel framework, named Recformer, which effectively learns language representations for sequential recommendation. Specifically, we propose to formulate an item as a "sentence" (word sequence) by flattening item key-value attributes described by text so that an item sequence

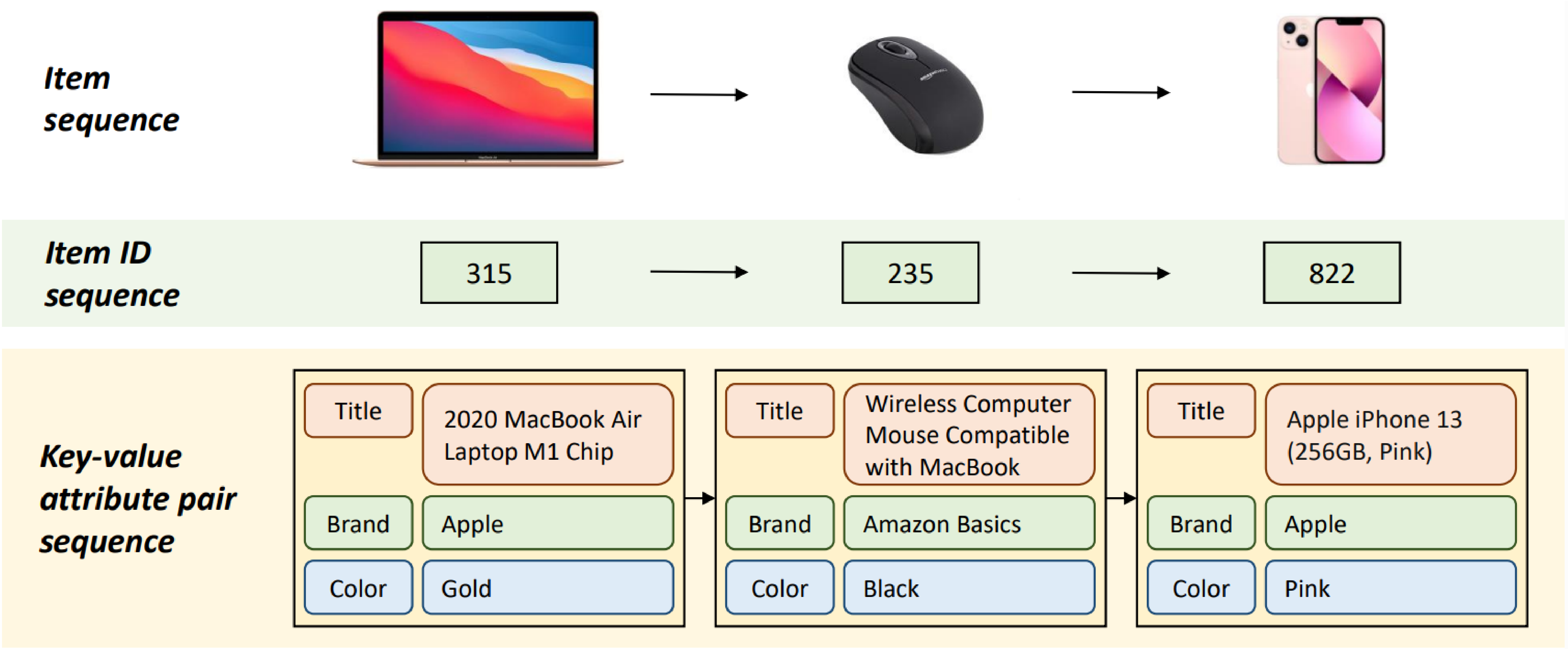
ACM Reference Format:

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599519

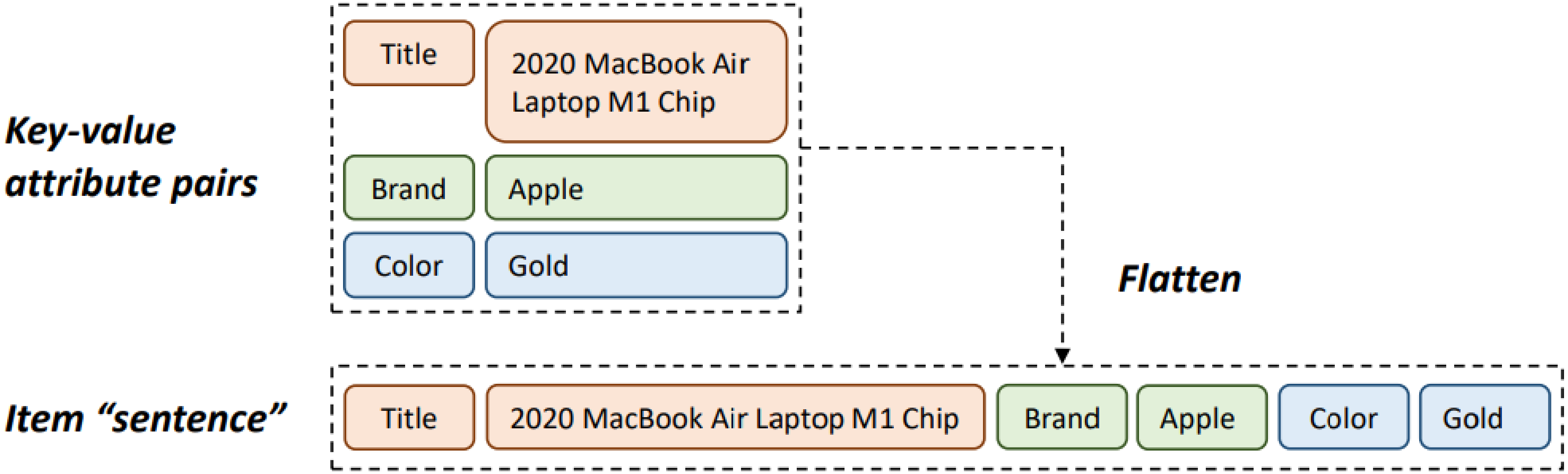
1 INTRODUCTION

Sequential recommender systems model historical user interactions as temporally-ordered sequences to recommend potential items that users are interested in. Sequential recommenders [11, 14, 25, 27] can capture both short-term and long-term preferences of users

Item Key-Value Attribute Pairs

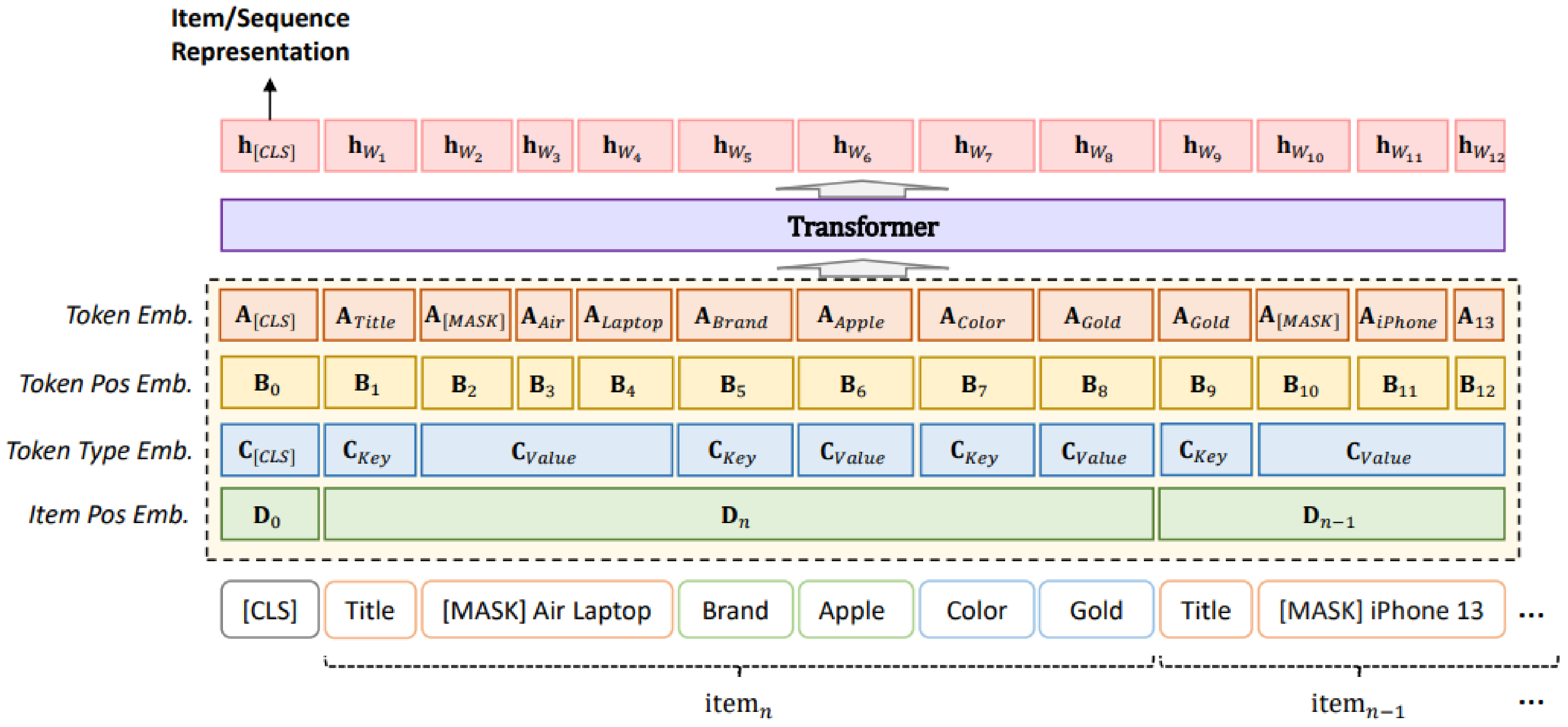


Flatten Key-Value Attribute Pairs into a "Sentence"



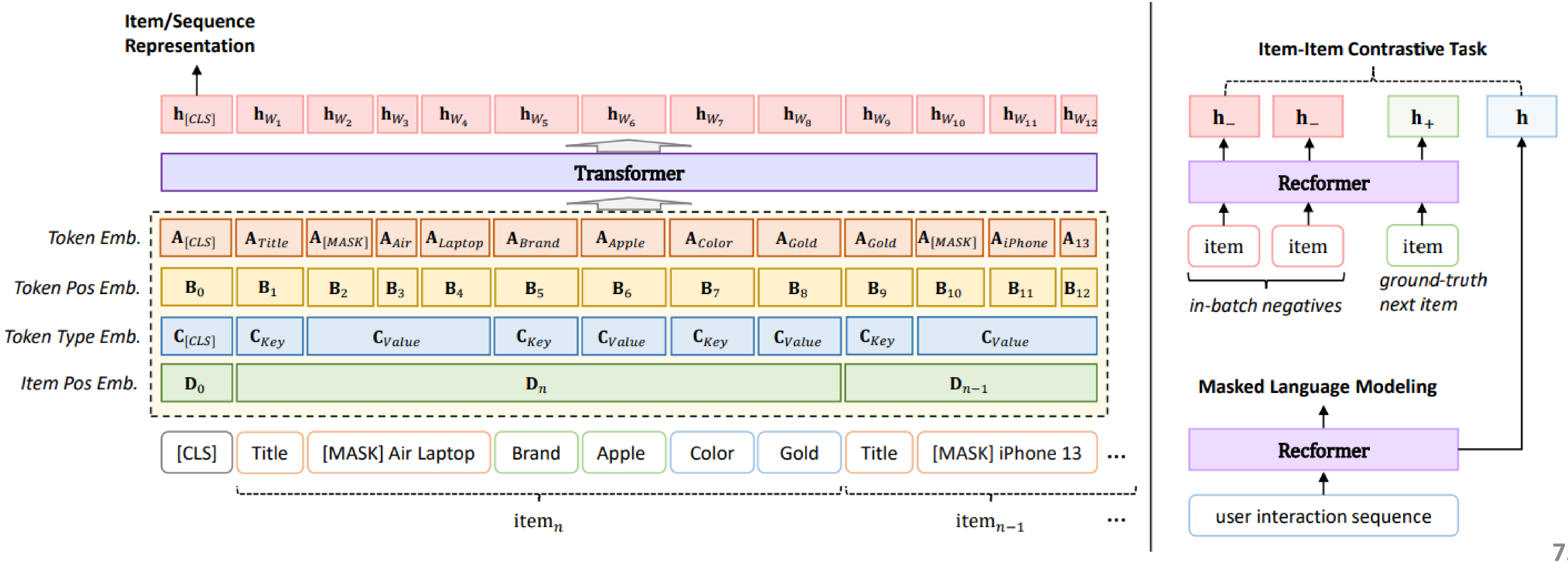
Feed "Sentences" into a Transformer Encoder

- Putting the most recent items on the left helps preserve them when the interaction history exceeds the model's maximum input length



Contrastive Learning

- Putting the most recent items on the left helps preserve them when the interaction history exceeds the model's maximum input length
- Contrastive learning helps the model better learn item representation vectors



Performance

Dataset	Metric	ID-Only Methods				ID-Text Methods		Text-Only Methods			Improv.
		GRU4Rec	SASRec	BERT4Rec	RecGURU	FDSA	S^3-Rec	ZESRec	UniSRec	RECFORMER	
Scientific	NDCG@10	0.0826	0.0797	0.0790	0.0575	0.0716	0.0451	0.0843	<u>0.0862</u>	**0.1027**	19.14%
	Recall@10	0.1055	<u>0.1305</u>	0.1061	0.0781	0.0967	0.0804	0.1260	0.1255	**0.1448**	10.96%
	MRR	0.0702	0.0696	0.0759	0.0566	0.0692	0.0392	0.0745	<u>0.0786</u>	**0.0951**	20.99%
Instruments	NDCG@10	0.0633	0.0634	0.0707	0.0468	0.0731	<u>0.0797</u>	0.0694	0.0785	**0.0830**	4.14%
	Recall@10	0.0969	0.0995	0.0972	0.0617	0.1006	<u>0.1110</u>	0.1078	**0.1119**	0.1052	-
	MRR	0.0707	0.0577	0.0677	0.0460	0.0748	<u>0.0755</u>	0.0633	0.0740	**0.0807**	6.89%
Arts	NDCG@10	<u>0.1075</u>	0.0848	0.0942	0.0525	0.0994	0.1026	0.0970	0.0894	**0.1252**	16.47%
	Recall@10	0.1317	0.1342	0.1236	0.0742	0.1209	<u>0.1399</u>	0.1349	0.1333	**0.1614**	15.37%
	MRR	0.1041	0.0742	0.0899	0.0488	0.0941	<u>0.1057</u>	0.0870	0.0798	**0.1189**	12.49%
Office	NDCG@10	0.0761	0.0832	<u>0.0972</u>	0.0500	0.0922	0.0911	0.0865	0.0919	**0.1141**	17.39%
	Recall@10	0.1053	0.1196	0.1205	0.0647	<u>0.1285</u>	0.1186	0.1199	0.1262	**0.1403**	9.18%
	MRR	0.0731	0.0751	0.0932	0.0483	<u>0.0972</u>	0.0957	0.0797	0.0848	**0.1089**	12.04%
Games	NDCG@10	0.0586	0.0547	<u>0.0628</u>	0.0386	0.0600	0.0532	0.0530	0.0580	**0.0684**	8.92%
	Recall@10	0.0988	0.0953	<u>0.1029</u>	0.0479	0.0931	0.0879	0.0844	0.0923	**0.1039**	0.97%
	MRR	0.0539	0.0505	<u>0.0585</u>	0.0396	0.0546	0.0500	0.0505	0.0552	**0.0650**	11.11%
Pet	NDCG@10	0.0648	0.0569	0.0602	0.0366	0.0673	0.0742	<u>0.0754</u>	0.0702	**0.0972**	28.91%
	Recall@10	0.0781	0.0881	0.0765	0.0415	0.0949	<u>0.1039</u>	0.1018	0.0933	**0.1162**	11.84%
	MRR	0.0632	0.0507	0.0585	0.0371	0.0650	<u>0.0710</u>	0.0706	0.0650	**0.0940**	32.39%

Thank You!

Course Website: https://yuzhang-teaching.github.io/CSCE670-S26.html