Rank It, Then Ask It: Input Reranking for Maximizing the Performance of LLMs on Symmetric Tasks

Mohsen Dehghankar Abolfazl Asudeh

University of Illinois Chicago
{mdehgh2, asudeh}@uic.edu

KDD '25: 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 3–7, 2025 – Toronto, ON, Canada

Outline

Motivation

Example 1

Consider a publications dataset in the form of a CSV file, containing the information of papers published across various domains:

Authors	**Title**	**Venue**	**Year**
Alan Turing	Computing Machinery and Intelligence	Mind	1950
...	...	...	...

Suppose one is interested in finding out the number of publications in an "Operations Research" (OR) venue since 2010. They specify their query in the form of a prompt[1] `[how many papers were published in an operations research venue since 2010]`, and pass it alongside the CSV file to an LLM to find the answer.

Motivation

- We study the application of LLMs to symmetric tasks.
- A Symmetric Tasks T is a pair (U, q).
  - U: A (large) bag of items (a set or a multi-set) $\{e_1, e_2, ..., e_n\}$
  - A query q about U.
- Example. Graph Degree Task:
  - U is the list of edges in any order.
  - q is a NL question like `[What is the degree of node 10?]`

Motivation

- We study the application of LLMs to symmetric tasks.
- A Symmetric Tasks T is a pair (U, q).
  - U: A (large) bag of items (a set or a multi-set) $\{e_1, e_2, ..., e_n\}$
  - A query q about U.
- Example. Graph Degree Task:
  - U is the list of edges in any order.
  - q is a NL question like `[What is the degree of node 10?]`

Observation:

- The set elements are not ordered, i.e., their order does not matter.
- LLMs process data in an ordered manner.
  - LLMs pay more attention to some positions
  - May ever forget parts of the input, especially in long prompts.
  - $\Rightarrow$ Forgetting may lead to incorrect responses.

Outline

Problem Formulation

LLM Model

- API, Black box, Access.
- Output Error: $\varepsilon_{\mathcal{L}}(U, q) = \Delta[\mathcal{L}(U, q), \mathcal{O}(U, q)]$.

Problem Definition

- Given a task (U, q) and a large language model $\mathcal{L}$
- Rerank the elements in U to minimize the expected error: $\mathbb{E}(\varepsilon_{\mathcal{L}}(U, q))$.

Our approach is task and query agnostic. In other words, we find the reranking function π^* without using any **explicit knowledge** about the query or the task.

Outline

Modeling as Utility Maximization

We define the *utility* of a reranking function π to capture the expected error $\mathbb{E}\left[\varepsilon_{\mathcal{L}}(U_\pi, q)\right]$.

Relevance

- The function $Rel_q : U \rightarrow [0, 1]$ captures the relevance of each element $e_i \in U$ to the query q.
- $Rel_q(e_i)$ is the relevance of e_i to the query q.

Exposure

- $\mathcal{X}_{\mathcal{L}}(i)$ to show the likelihood that the LLM will not miss an element in position i

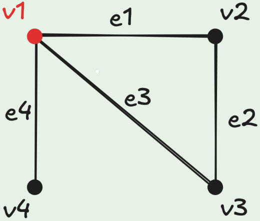
Utility of a ranking π of U

$utility(\pi|q) = \sum_{i=1}^{|U|} \mathcal{X}_{\mathcal{L}}(i) \cdot Rel_q(e_{\pi(i)})$

Modeling as Utility Maximization – Example

Example 2: Node Degree Computation

- Consider the following graph G, given as a list of edges: $\{e_1, e_2, e_3, e_4\}$.
- Let the exposure function be $\mathcal{X}_{\mathcal{L}}(i) = \frac{1}{i}$
- query q: `[compute the degree of` v_1`]`



Example 2 – Max. Utility

- For edges incident to v_1, $Rel_q(e_i) = 1$; for the others the relevance is 0.
- $\Rightarrow$ the utility of the ranking $\pi = \{e_1, e_2, e_3, e_4\}$ is $utility(\pi|q) = 1 + \frac{1}{3} + \frac{1}{4} \simeq 1.58$.
- Note that the ranking with maximum utility puts e_1, e_3, and e_4 at the beginning of the list, and has the utility of $1 + \frac{1}{2} + \frac{1}{3} \simeq 1.83$.

Outline

Relevance Estimation

- We utilize helper LLMs for estimating the relevance values.
- **Warm-up:**
  - Partition the input U into m equally sized chunks: $[P_1, P_2, \cdots, P_m]$.
  - For each chunk, ask the helper LLM to find the relevant ones to the query.
  - `[Which elements in` $[P_i]$ `are more relevant for answering the query` $[q]$`?]`
  - **Issues?**
  - **Alternative?**

Relevance Estimation

- We utilize helper LLMs for estimating the relevance values.
- **Warm-up:**
  - Partition the input U into m equally sized chunks: $[P_1, P_2, \cdots, P_m]$.
  - For each chunk, ask the helper LLM to find the relevant ones to the query.
  - `[Which elements in` $[P_i]$ `are more relevant for answering the query` $[q]$`?]`
- **Modeling as a Bipartite Graph:** An approach inspired by the **peer review process**.

- Randomly shuffle the input list σ times: $U_1, U_2, \cdots, U_\sigma$
- Partition each shuffle U_i into m chunks: $\{P_{i,1}, P_{i,2}, \cdots, P_{i,m}\}$
- **Evaluation:** Ask the helper to give discrete scores to each chunk.
  - Total of $\sigma \cdot m$ evaluations by helper model: $\{\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_{\sigma m}\}$

Bipartite Graph

Left nodes: Estimated scores

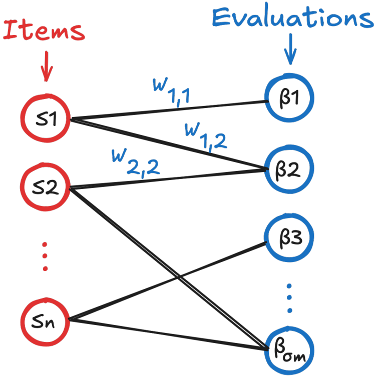
$$\{S_1, \cdots, S_n\}$$

Right nodes: Bias in each evaluation

$$\{\beta_1 \cdots, \beta_{\sigma m}\}$$

Edges: Score assigned to item i in evaluation j:

$$w_{i,j}$$



Relevance Estimation – Bipartite Graph (Cont.)

- Intuition: Each evaluation j, equally under/over estimates all the scores $w_{i,j}$:

$$w_{i,j}^{unbiased} = \frac{w_{i,j}}{\beta_j}$$

- Using the bipartite graph values, the following equations hold:

$$S_i = \frac{1}{\sigma} \sum_{(i,j)} \frac{w_{i,j}}{\beta_j}$$

$$\beta_j = \frac{1}{\lceil \frac{n}{m} \rceil} \sum_{(i,j)} \frac{w_{i,j}}{S_i}$$

- S_i and β_j values are unknown.

Relevance Estimation: Learning the bipartite graph values

1. Initialize $\beta_j^{(0)} = 1, \ \forall j \in [\sigma m]$
2. Iteratively update the values until convergence
    1. $$\bar{S}_i^{(T)} = \frac{1}{\sigma} \sum_{(u_i, v_j) \in E} \frac{w_{i,j}}{\beta_j^{(T-1)}}, \qquad \forall u_i \in U$$
    2. $$\beta_j^{(T+1)} = \frac{1}{\lceil \frac{n}{m} \rceil} \sum_{(u_i, v_j) \in E} \frac{w_{i,j}}{\bar{S}_i^{(T)}}, \qquad \forall u_i \in U$$

Theorem

The bipartite graph value estimation process would eventually converge.

Outline

Highlighted Experiments

Table: Comparing final task error, proximity (absolute value), across methods and helper LLMs. Database query task on IMDB dataset.

Algorithm	DeepSeek	Gemma2	Llama3.1	Mistral	Qwen2
Random (UB)	1.00 (1.18) ↑	1.00 (1.24) ↑	1.00 (1.32) ↑	1.00 (0.90) ↑	1.00 (1.24) ↑
Warm-up	0.56 (0.92)	0.87 (1.12) ↑	0.85 (1.20) ↑	**0.49** (0.60)	0.50 (2.12)
Bipartite	**0.03** (0.60) ↓	**0.29** (0.56) ↓	**0.04** (0.52) ↓	0.69 (0.72)	**0.48** (2.72)
Optimum (LB)	0.00 (0.58) ↓	0.00 (0.28) ↓	0.00 (0.48) ↓	0.00 (0.30) ↓	0.00 (0.42) ↓

For a method with error e, the proximity of the error is:

$$0 \leq Prox = \frac{e - LB}{UB - LB} \leq 1$$

- Error e is the average of $|output - ground_truth|$.

Highlighted Experiments (Cont.)

Table: Comparing the ranking utility of the final reranking generated by different methods and helper LLMs. IMDB dataset for DB Query task. Higher is better.

Algorithm	**DeepSeek**	**Gemma2**	**Llama3.1**	**Mistral**	**Qwen2**
Optimum (UB)	2.76 (100%) ↑	2.60 (100%) ↑	2.69 (100%) ↑	2.52 (100%) ↑	2.67 (100%) ↑
Bipartite	**2.63 (94%)** ↑	2.50 (95%) ↑	**2.48 (90%)** ↑	**2.22 (84%)** ↑	**1.60 (48%)**
Warm-up	1.30 (33%)	**2.58 (99%)** ↑	1.68 (52%)	**2.22 (84%)** ↑	1.50 (44%)
Random (LB)	0.57 (0%) ↓	0.48 (0%) ↓	0.58 (0%) ↓	0.55 (0%) ↓	0.58 (0%) ↓

Highlighted Experiments – Exposures

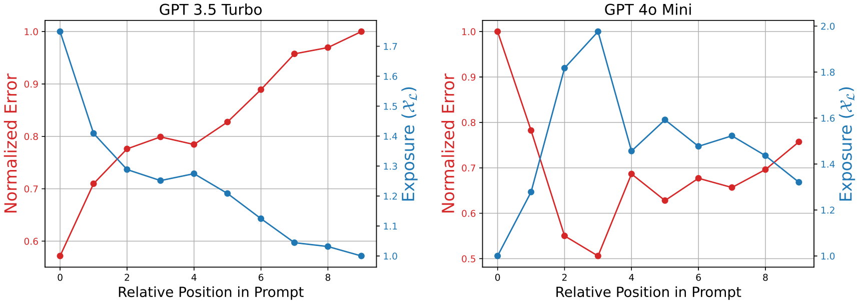


Figure: Token exposures and errors[2] relative to the location in prompt.

[2] The higher the error, the higher likelihood of being forgotten.

Thank you!

Mohsen Dehghankar
mdehgh2@uic.edu

Preprocessing: Exposure Values Discovery

- (Recall) $\mathcal{X}_{\mathcal{L}}(i)$: the likelihood that the model misses a token at position i of the input. (Mohsen: exposure is the reverse of likelihood)
- Learning the exposure values: we consider a sample set of predefined tasks:
  - A query q
  - The input elements $U = [t_1, t_2, \ldots, t_n]$ (consisting of n tokens arranged in sequential order).
  - The ground-truth relevance value.
- We model the relation between the error and the exposures as,

$$\frac{1}{\mathbb{E}[\epsilon_{\mathcal{L}}(U, q)]} \propto \frac{1}{n} \sum_{i=1}^{n} (\mathcal{X}_{\mathcal{L}}(i) \cdot Rel_q(t_i))$$