# Robust In-Context Selection via Online Learned Position-Corrected Attention

Anonymous ACL submission

## Abstract

Large Language Models (LLMs) are often deployed in tasks that require selecting an item from a long list provided in the model's context. LLMs' native selection behavior is brittle: predictions are sensitive to the surface form of the identifiers, their placement within the context, and the ordering of candidate items. We present OLR-heads, a robust method for list selection that harnesses attention patterns available from a single forward call on the LLM. OLR-heads learns the logic for item selection using a few in-context examples, and a simple online position-debiasing mechanism to correct attention distortion. Across multiple database and tool selection benchmarks, OLR-heads consistently improves selection performance over direct generation and prior attention-based methods, while remaining robust to prompt variations and item ordering, and incurring negligible overheads. The LLM's KV cache states are unaffected, and can be reused for subsequent response generation. In contrast, existing approaches either entail additional LLM calls, or task-specific offline learning, or position debiasing methods that modify the attention or encoding rendering the KV states unusable for subsequent generation.

## 1 Introduction

With the prevalence of agents, LLMs are often used in scenarios where they need to select an item, such as a tool, a function, or a database from a long list of options in its prompt, and encoded in its long context. Several studies have reported that LLM's native selection of relevant tools is often unreliable: predictions are sensitive to the ordering of candidate items (Liu et al., 2023; Zhang et al., 2024; Schilcher et al., 2025), and we will show also to the surface form of the identifiers, and their placement within the context.

Existing approaches to fix this limitation can be categorized two ways: (1) Methods that use the LLM to generate the selection either via logits of item identifiers (Gangi Reddy et al., 2024; Hao et al., 2024), or explicit rank permutations (Sun et al., 2024; Liu et al., 2025b) often requiring multiple LLM calls or fine-tuning. (2) Methods that harness the attention from the query to the items in the list. These work only after identification of relevant heads requiring task-specific offline discovery (Zhang et al., 2025; Tran et al., 2025) and position debiasing performed either via a separate LLM with a dummy query (Chen et al., 2025) or modifying position embeddings (Xiao et al., 2025; Yu et al., 2025).

Our goal was to design a method that can allow in-context item selection with existing KV cache states of a long context, without incurring any further LLM calls. Thus, ours is an online, read-only method that does not modify document layout or how attention is computed so that the LLM can reuse the KV cache for further processing needed to generate a response. This is unlike existing methods discussed above that view selection as a standalone step and have one or more of these requirements: additional LLM calls, offline task-specific training, custom attention, or fine-tuning.

We propose OLR-heads which builds upon recent attention-based selection methods by making them online through two simple ideas: (1) harness in-context examples that are typically provided to teach models new tools or functions, to online discover a sparse set of retrieval heads, and (2) correct position bias with in-context anchor tokens separating documents from in-context examples.

Across seven tasks spanning tool selection and database routing, we show that OLR-heads provide better selection accuracy, greater robustness than generative methods to prompt variations, identifier type and placement, and item ordering, while incurring negligible additional overheads beyond a normal LLM forward call.

**Contributions**

1. We propose a low-overhead, read-only, online method of improving the accuracy of list selection from an LLM's context. We propose a very simple method of position debiasing that does not involve a separate tool-call, and thus does not require tampering with the embeddings of the LLM.
2. We compare with existing methods of harnessing LLMs for reranking and list selection on seven datasets, and show our OLR-heads provides competitive accuracy while incurring negligible additional time overheads.
3. We show that LLMs' native selection is brittle: predictions are sensitive to the surface form of the identifiers, their placement within the context, and the ordering of candidate items. In contrast, OLR-heads is robust to these variations.

## 2 Related Work

**LLMs Generate Selections** RankGPT (Sun et al., 2024) is one of the first methods to show that LLMs could perform zero-shot listwise reranking by generating a ranked list of identifiers. For long lists, it is effective only when performed over multiple smaller sliding windows, but this entails multiple LLM calls. A number of alternatives have been proposed to handle this inefficiency but they all require additional training. Gangi Reddy et al., 2024 trains the output logits of the first generated identifier to output probabilities of each candidate ID in a single step. Further, Hao et al., 2024; Yakovlev et al., 2024 proposes to extend the vocabulary with tool names to allow selection via logits. Liu et al., 2025a proposes to compress long documents into a single dense embedding token. Pradeep et al., 2023 distills smaller models for ranking and Liu et al., 2025b trains a long-context models for full ranking.

**Attention for Reranking** In contrast to the above approaches that 'generate' the ranking using the LLM, some recent work harness attention patterns inside LLMs to score items. Wu et al., 2024 mechanistically examine this phenomenon, showing that a small subset of attention heads, called as retrieval heads, are responsible for copying or reusing information from input spans in long-context settings. Zhang et al., 2025 build upon the retrieval-head concept by proposing Query-Focused Retrieval Heads (QR-Heads), with strong query-to-context attention. Using these heads, their QR-Retriever aggregates attention signals to rank candidate passages, achieving significant improvements on long-context reasoning and on IR tasks. Tran et al., 2025 further introduce a contrastive scoring-based method for head selection.

These studies establish that a small number of heads, typically in the middle layers, can deliver state-of-the-art ranking performance. But these methods require offline training for head selection which makes this approach applicable only for tasks where offline training and storage of selected heads is made part of the task-specific pipeline. We build upon these studies, but propose a purely online method applicable on-the-fly on any LLM invocation requiring selection of items from a list.

**Position correction methods** Recent studies (Liu et al., 2023; Zhang et al., 2024; Schilcher et al., 2025) have shown that LLMs exhibit positional bias, where changing the ordering of items in the prompt significantly impacts LLM response, particularly in the long-context. Several approaches have been proposed to mitigate this positional bias. Rotary Position Embeddings (Su et al., 2023) directly encode positional information in a transformer. One category of methods (Xiao et al., 2025; Yu et al., 2025) attenuates that to change the attention values before they are computed. A second category encode documents differently to incorporate position invariance (Wang et al., 2025; Yoon et al., 2025), a third category debias with learned focus directions (Zhu et al., 2025), while others fine-tune for reduced position bias (Zhang et al., 2024). All these methods change the way attention is computed in the LLM, and view item selection as a stand-alone operation. Our method is designed to select an item in-place, where after selection the LLM may use the same KV cache state to do further processing needed to generate a user response. As such, we are interested in read-only methods that do not modify the default LLM further processing. The closest to our approach is the debiasing used in (Zhang et al., 2025) but their method requires an additional LLM call to debias with the attention from a neutral "N/A" query.

## 3 Our method

Let $\mathbf{x}$ denote a user query, and $D(\mathbf{x})$ denote the list of items (e.g. tools or databases or documents) from which we need to select the small subset (typically one) that is relevant to $\mathbf{x}$. We are consider-

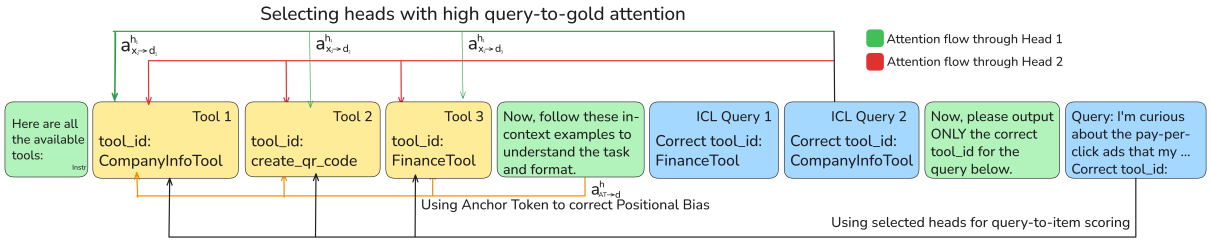


Figure 1: Overview of OLR-heads. The method first identifies a sparse set of retrieval-relevant attention heads by selecting heads with high position-debiased query-to-gold attention averaged over in-context examples. The selected heads are then reused to compute position-debiased attention scores between the test query and candidate items, producing a final ranking.

ing settings where the number of items in $D(\mathbf{x})$ is large, say $O(100)$ We assume access to a small set of labeled in-context examples $\mathcal{E} = \{(\mathbf{x}_i, d_i^\star)\}_{i=1}^K$ where each query is paired with its ground-truth relevant item $d_i^\star \in D(\mathbf{x})$. Typically, $\mathcal{E}$ is small(5-10 examples), and we show that even this small size suffices for our method of item selection. Our objective is to rank items in $D(\mathbf{x})$ using a single forward pass of a frozen model, by exploiting internal attention signals computed during a forward pass over the model.

**Prompt Format.** We assume a prompt containing the candidate pool $D(\mathbf{x})$, the small set $\mathcal{E}$ of in-context examples, and the test query $\mathbf{x}$. Each candidate item, $d \in D(\mathbf{x})$ is represented by a unique identifier followed by its description. Items in the pool are separated using a fixed delimiter(e.g \n\n) to ensure consistent token boundaries across prompts. We use $\mathcal{T}(d)$ to denote the token span of an item $d \in D(\mathbf{x})$, $\mathcal{T}(\mathbf{x}_i)$ to denote the token span of an example in the ICL set and likewise the token space of the target query $\mathcal{T}(\mathbf{x})$.

**Attention Extraction.** A forward pass with the LLM on the above prompt yields an attention matrix between queries and items. Let the model consists of $L$ layers and $H$ attention heads per layer, yielding a total of $\mathbb{H} = L \times H$ attention heads. For each head $h$, the attention matrix is represented by $A^h$. We define a per-head attention score between a query $\mathbf{x}$ and an item $d \in D(\mathbf{x})$ by aggregate attention values from all query tokens to all item tokens as:

$$a^h_{\mathbf{x}\to d} = \frac{1}{|\mathcal{T}(\mathbf{x})|} \sum_{t_x \in \mathcal{T}(\mathbf{x})} \sum_{t_d \in \mathcal{T}(d)} A^h[t_x, t_d] \quad (1)$$

**Position Debiasing.** As several previous studies have shown, the raw attention values exhibit position bias and as discussed in Section 2 several methods are proposed for correcting the bias, including subtracting the attention from a special call with a dummy query (Zhang et al., 2025). We introduce an on-the-fly position debiasing as follows: We denote the instructional tokens between the documents list $D(\mathbf{x})$ and the first ICL example as a special anchor span $AT$ as shown in Figure 1. The attention from $AT$ to each of the documents $(a^h_{AT\to d})$ denotes an example-neutral attention to each document $d$. Using this we define the position-corrected attention as

$$\tilde{a}^h_{\mathbf{x}\to d} = a^h_{\mathbf{x}\to d} - a^h_{AT\to d} \quad (2)$$

Intuitively, this subtraction removes attention mass that arises due to positional bias as the text span $AT$ should have attended uniformly across all items in the absence of position bias.

**Head Detection via In-Context Examples.** We use the in-context examples to identify a small subset of attention heads that encode retrieval signals, i.e., heads that attend strongly from a query to its relevant item. For each example $(\mathbf{x}_i, d_i^\star)$, we compute the position-corrected query-to-gold-item attention for each head $a^h_{\mathbf{x}_i \to d_i^\star}$ using Equation 2. A useful head would attend from the query to the gold item across in-context examples. We therefore define the score of head as average corrected attention across all $K$ in-context examples, and choose the top-$R$ heads as:

$$\mathcal{H}_{top} = TopK(\{\sum_{i=1}^{K} \tilde{a}^h_{\mathbf{x}_i, \to d_i^\star} | h \in \mathbb{H}\}, R) \quad (3)$$

**Test query Scoring.** Now, for the final user query $\mathbf{x}$, we compute the total de-biased attention over the selected heads $\mathcal{H}_{top}$ for each item $d \in D(\mathbf{x})$, choose the item with the highest attention.

$$\hat{d} = \text{argmax}_{d \in D(\mathbf{x})} \sum_{h \in \mathcal{H}_{top}} \tilde{a}^h_{\mathbf{x}\to d} \quad (4)$$

## 4 Experiments

We evaluate OLR-heads with several existing methods on accuracy of selecting the right item, robustness to prompt variation, and running time.

### 4.1 Baselines

We compare OLR-heads against a diverse set of lexical, dense, and LLM-based re-ranking baselines, covering both traditional IR methods and recent generation-based approaches.

**Direct generation (Direct Gen.).** This is the default method, where the LLM generates the selected item from the options in the context.

**LLM-based zero-shot re-rankers.** We compare with these methods that rerank using an LLM without any task-specific training. (1) RankGPT (Sun et al., 2024): ranks documents incrementally using sliding windows, and as such requires $\frac{N-w-s}{w-s}$ calls to rerank $N$ items using windows of size $w$ and stride $s$. We report numbers with $w = N$, that is a single LLM call, and for smaller $w, s$ values chosen so that $w = 2s$ and number of calls is no more than 5. All configurations of sliding window are discussed in Appendix F. (2) In-Context-Re-ranking (ICR; Chen et al., 2025): computes document relevance using attention weights aggregated across all attention heads, using a separate dummy query's attention to do position debiasing.

**LLM-based re-rankers with light training.** QRHEAD: closely follows ICR but performs offline attention head selection using the same labeled set as available to OLR-heads.

Although our method is designed for in-context retrieval, for reference we provide results with standard document retrievals baselines in Appendix C.

### 4.2 Datasets

We consider seven datasets spanning two task categories: database selection and tool selection as summarized in Table 8. For database selection, we use Spider (Yu et al., 2019) and Bird (Li et al., 2023) datasets where Spider contains 166 databases, while Bird contains 80 databases. Each database is described with its name and full schema. The task is to route each query to the one database that can answer this query (Wang et al., 2023). Number of test queries is 1000, randomly sampled from the combined training and development splits. For tool selection, we used five datasets. Each dataset includes a list of tools with a name and a brief textual description, and a user query needs to select one or more tools. The first dataset is ToolE (Huang et al., 2023) where we evaluate on 2000 queries over 199 tools. The remaining four are from the ToolRet repository (Shi et al., 2025). We selected the four largest: CRAFT-TabMWP, CRAFT-Algebra, APIBank and ToolEyes.

**Evaluation Metric.** We evaluate all queries using Recall@1, treating tool retrieval as a single-item selection problem. This design choice allows to assess the robustness of retrieval under long prompts while maintaining a fixed prompt structure consistent with prior datasets and baselines.

**Models and Setup** We experiment with three open-weight, instruction-tuned language models from two families including Qwen2.5(7B and 7B-1M) of Qwen family (Yang et al., 2024), and LLaMA-3.1(8B) of Llama family (Dubey et al., 2024). For both QRHEAD and OLR-heads we select 20 heads and five examples for head selection. Further experimentation details can be found in Appendix A

### 4.3 Accuracy of Selection

Table 1 reports Recall@1 across different methods, datasets, and models. Under the LLaMA backbone, OLR-heads achieves the strongest performance across all datasets. It outperforms direct generation(Direct Gen.), attention-based re-rankers such as ICR and QRHEAD and both versions of RankGPT on every benchmark. These results indicate that retrieval-relevant attention signals in LLaMA are strong and stable, and can be effectively exploited online using just a few in-context examples. RankGPT(without SW) that attempts to output the ranking with a single LLM prompt, is significantly worse than all methods.

For the Qwen-2.5-7B-Instruct model, OLR-heads outperforms Direct Gen., ICR and QRHEAD across datasets. RankGPT(with SW) performs exceptionally well on this model. This behavior is expected, as RankGPT decomposes long contexts into smaller windows that fall within the model's native context length (32K tokens), allowing generation to operate under favorable conditions. However, this comes at the cost of multiple generation calls per query, significantly blowing inference time as we show in Section 4.5. As shown in the second row of the table, except for ToolE, for all other datasets the number of tokens exceeds 25K, and is beyond 32K tokens for four of seven the datasets. This is also reflected in the abysmal performance of this

| Method | ToolE | Spider | Bird | APIbank | Tooleyes | Algebra | Tabmwp |
|---|---|---|---|---|---|---|---|
| **Prompt len** | 5k | 42k | 35k | 25k | 28k | 64k | 54k |
| **Base LM: Llama-3.1-8B-Instruct** | | | | | | | |
| Direct Gen. | 62.2 | 53.3 | 57.3 | 75.3 | 58.8 | 41.1 | **19.0** |
| ICR | 51.3 | 63.4 | 66.2 | 64.7 | 61.3 | 35.2 | 13.0 |
| QRHEAD | 53.9 | **67.3** | 71.4 | 68.2 | 62.5 | 51.7 | 17.0 |
| RankGPT(with SW) | 59.3 | 60.9 | 53.1 | 72.9 | 51.3 | 57.2 | 13.0 |
| RankGPT(without SW) | 15.4 | 6.4 | 11.1 | 48.2 | 13.7 | 0.4 | 0.6 |
| OLR-heads$^{sim}$ | **62.5** | 67.0 | **73.4** | **77.6** | **68.8** | **60.2** | **19.0** |
| **Base LM: Qwen-2.5-7B-Instruct** | | | | | | | |
| Direct Gen. | 55.5 | 42.6 | 48.1 | 69.4 | 42.5 | 11.9 | 4.8 |
| ICR | 44.5 | 31.7 | 27.7 | 41.2 | 52.5 | 11.0 | 8.1 |
| QRHEAD | 51.3 | 51.2 | 38.9 | 49.4 | 58.8 | 15.3 | 17.7 |
| RankGPT(with SW) | **67.4** | **73.5** | **66.6** | **80.0** | **65.0** | **77.1** | **27.2** |
| OLR-heads$^{sim}$ | 59.4 | 59.4 | 65.4 | 76.5 | 55.0 | 52.1 | 17.7 |
| **Base LM: Qwen-2.5-7B-Instruct-1M** | | | | | | | |
| Direct Gen. | 63.1 | 52.9 | 58.5 | 77.6 | 48.8 | 27.5 | 8.8 |
| ICR | 48.7 | 54.5 | 65.1 | 49.4 | 60.0 | 39.4 | 15.6 |
| QRHEAD | 52.5 | 63.7 | 67.8 | 60.0 | 62.5 | 47.5 | 18.4 |
| RankGPT(with SW) | **67.4** | **74.5** | 70.1 | **82.4** | **67.5** | **78.8** | **25.9** |
| RankGPT(constrained SW) | 52.1 | 68.2 | 68.1 | 51.7 | 61.3 | 36.4 | 17.0 |
| RankGPT(without SW) | 37.1 | 37.8 | 41.9 | 51.7 | 42.5 | 30.5 | 6.8 |
| OLR-heads$^{sim}$ | 62.9 | 69.0 | **71.6** | 80.0 | 62.5 | 58.5 | 22.5 |

Table 1: Performance comparison (Recall@1) across database and tool selection datasets under different backbone models. Candidate items are represented using the string-based variant. OLR-heads consistently outperforms direct generation and attention-based re-ranking baselines across backbones. RankGPT with sliding windows performs strongly under Qwen models but requires multiple generation calls; when constrained to 5 calls, performance degrades considerably.

Qwen model compared to the Llama model even in the Direct Gen. setting.

Consequently, we moved to the Qwen-2.5-7B-Instruct-1M, a long-context variant supporting up to 1M tokens. With this model, OLR-heads shows consistent performance improvements across datasets compared to the standard Qwen-2.5-7B model. In contrast,RankGPT(with SW) shows only marginal gains when moving from Qwen-2.5-7B-Instruct to Qwen-2.5-7B-Instruct-1M. On the Craft-Algebra dataset, RankGPT(with SW) makes close to 30 LLM calls, when we constrain it down to 5, RankGPT(constrained SW) sees a massive drop in accuracy.

Overall, attention-based retrieval using OLR-heads remains competitive or superior to generation-based approaches across backbones and datasets and benefits directly from improved long-context modeling.

```
tool_id: WebsiteTool
tool description: Quickly create and deploy
websites, and publish content on them.
```

```
tool_id: [5]
tool name: WebsiteTool
tool description: Quickly create and deploy
websites, and publish content on them.
```

```
tool name: WebsiteTool
tool description: Quickly create and deploy
websites, and publish content on them.
tool_id: [5]
```

Figure 2: Prompt variants used in robustness experiments. Top: `string`. Middle: `id_top`. Bottom: `id_bottom`.

### 4.4 Robustness

We evaluate robustness by measuring performance under prompt variations, while keeping the task, candidate pool, and evaluation fixed. We show that attention patterns in LLM that OLR-heads harnesses are significantly more robust than an LLM's generation pipeline. For these, we compare Direct

Gen. and OLR-heads, under the same prompt.

**Sensitivity to Item Representation.** Table 2 reports Recall@1 under three different item representations as shown in Figure 2. Our default is the string variant where each item is identified by its original name (e.g., database schema names such as `Gymnast` or `Scholar` in Spider) placed at the top. We first switch the identifier to a number (e.g., `[1]`, `[2]`) as shown in the second row of Figure 2, call it the `id_top` variant. Next, we change the position of the numeric identifier to the bottom of the description as shown in the third row and call it the `id_bottom` variant.

As seen in Table 2, LLM's direct generation shows substantial sensitivity to these variations. In particular, Direct Gen.$^{\text{id\_top}}$ leads to large performance drops across all datasets and LLM models. Under LLaMA, generative performance on Spider and BIRD drops from 59.0 and 69.5 (Direct Gen.$^{\text{id\_bottom}}$) to 14.5 and 16.1 (Direct Gen.$^{\text{id\_top}}$), respectively, with similar degradation observed under Qwen. The transition from string to identifier based representations(`id_top` and `id_bottom`) also introduces noticeable performance fluctuations for generation. One likely explanation for the failure of generation under `id_top` variant is that numeric identifiers appearing at the beginning of item descriptions are not contextualized and may instead bias the model toward previously seen identifiers in the prompt. Since OLR-heads does not rely on token generation and instead aggregates attention between query and item tokens, it is substantially less affected by such prompt-level artifacts.

**Sensitivity to Item Ordering.** We compute the *per-query rank variance*[1] of the gold item across six random item orderings on the BIRD dataset and the Llama model. We observed that Direct Gen.$^{\text{id\_bottom}}$ has a substantially higher rank variance of 6.3 compared to OLR-heads$^{\text{id_bottom}}$'s variance of 2.6. This indicates that OLR-heads produces more stable rankings and is less sensitive to changes in item orderings.

Overall, these results demonstrate that OLR-heads is significantly more robust to prompt variations and item ordering than direct generation by the LLM.

[1]Note, it does not make sense to observe change in average accuracy across queries since every ordering is favorable to some query.

| Method | ToolE | Spider | Bird |
|---|---|---|---|
| **Base LM: Llama-3.1-8B-Instruct** | | | |
| Direct Gen. | 62.2 | 53.3 | 57.3 |
| Direct Gen.$^{\text{id\_top}}$ | 40.3 | 14.5 | 16.1 |
| Direct Gen.$^{\text{id\_bottom}}$ | 60.5 | 59.0 | 69.5 |
| OLR-heads | 62.3 | 65.8 | 72.1 |
| OLR-heads$^{\text{id_top}}$ | 60.2 | 64.6 | 69.0 |
| OLR-heads$^{\text{id_bottom}}$ | 63.4 | 65.7 | 70.4 |
| **Base LM: Qwen-2.5-7B-Instruct** | | | |
| Direct Gen. | 55.5 | 42.7 | 48.1 |
| Direct Gen.$^{\text{id\_top}}$ | 30.8 | 13.1 | 19.6 |
| Direct Gen.$^{\text{id\_bottom}}$ | 51.3 | 33.9 | 40.8 |
| OLR-heads | 61.7 | 57.4 | 64.8 |
| OLR-heads$^{\text{id_top}}$ | 55.9 | 53.6 | 60.9 |
| OLR-heads$^{\text{id_bottom}}$ | 62.8 | 55.2 | 63.0 |

Table 2: Recall@1 under different item representations. By default, we use the string-based representation. Direct Gen. exhibits large performance variations across representations: changing the string identifier to a numeric one (`id_top`) causes huge drop in accuracy, and moving the identifier to the bottom (`id_bottom`) recovers the loss. In contrast, OLR-heads remains relatively stable across item representations.

| Method | Spider | Bird |
|---|---|---|
| Direct Gen. | 11.4 | 9.3 |
| QRHEAD | 11.1(+59) | 8.8(+47) |
| RankGPT(with SW) | 58.1 | 37.6 |
| OLR-heads | 11.6 | 9.6 |

Table 3: Inference time (in seconds) comparison on long-context(>35k tokens) datasets. For QRHEAD, an additional one-time overhead for head detection is reported in parentheses.

### 4.5 Running time

Table 3 compares inference latency on long-context inputs (>32K tokens) for Spider and Bird as measured on a single NVIDIA RTX 6000 Ada GPU (48 GB VRAM) using LLaMA-3.1-8B-Instruct. Reported values correspond to average per-query inference time. OLR-heads achieves inference latency comparable to direct generation. When using similarity-based in-context example selection, OLR-heads$^{sim}$ incurs an additional overhead of approximately 0.3 seconds per query due to query encoding and searching. In contrast, RankGPT(with SW) is substantially slower, incurring 5–6× higher latency due to repeated generation calls over item subsets. While QRHEAD shows similar per-query inference time, it requires an additional one-time

| Method | ToolE | Spider | Bird |
|---|---|---|---|
| **Base LM: Llama-3.1-8B-Instruct** | | | |
| OLR-heads$^{\text{rand}}$ | 62.3 | 65.8 | 72.1 |
| OLR-heads$^{sim}$ | 62.5 | 67.0 | 73.4 |
| **Base LM: Qwen-2.5-7B-Instruct** | | | |
| OLR-heads$^{\text{rand}}$ | 61.7 | 57.4 | 64.8 |
| OLR-heads$^{sim}$ | 59.4 | 59.4 | 65.4 |

Table 4: Performance comparison of random versus similarity-based in-context example selection. Similarity-based selection consistently improves Recall@1 across datasets and models.

| $K$ | ToolE | Spider | Bird |
|---|---|---|---|
| 2 | 61.9 | 64.1 | 71.0 |
| 5 | 62.3 | 65.8 | 72.1 |
| 10 | 63.3 | 65.3 | 71.5 |
| 20 | 63.9 | 65.7 | 71.3 |

Table 5: Effect of number of in-context examples ($K$) on Recall@1 across datasets. Performance saturates with a small K, indicating few in-context examples are sufficient to select retrieval-relevant heads.

offline head selection step, which takes approximately 59 seconds on Spider and 47 seconds on BIRD. OLR-heads does not require any such offline computation, making it more suitable for settings where candidate pools change over time.

Overall, these results show that OLR-heads provides competitive latency while avoiding the repeated generation overhead of RankGPT and the offline preprocessing cost of QRHEAD.

### 4.6 Ablation

We conduct ablation studies to analyze the impact of key design choices in OLR-heads. In particular, we examine how in-context example selection, the number of in-context examples $K$, position debiasing method, and the number of selected attention heads (in Appendix E) affect performance. All ablation experiments are conducted on LLaMA-3.1-8B-Instruct. Table 4 shows the effect of in-context example selection: whether examples are selected uniformly at random or based on semantic similarity to the test query. These results indicate that similar ICL examples provide slight edge over randomly selected examples.

We next vary the number of in-context examples $K$ in Table 5. On Spider and BIRD, performance saturates quickly, with as few as five examples sufficient to identify effective heads. In contrast, ToolE benefits from larger values of $K$, showing modest improvements as more examples are added. We attribute this difference to the higher semantic overlap among tools in ToolE, where additional examples help model to more query-tool interactions, leading to better selection of heads.

| Position Debiasing | ToolE | Spider | Bird |
|---|---|---|---|
| both | 62.3 | 65.8 | 72.1 |
| None | 58.4 | 65.9 | 71.5 |

Table 6: Effect of positional debiasing on Recall@1. Positional debiasing is applied during head detection and inference (*both*) or removed entirely (*none*).

Table 6 analyzes the effect of positional debiasing, which is applied both during head detection and at inference time (Section 3). We compare against a variant where positional debiasing is removed in both stages. Removing positional debiasing leads to consistent performance degradation, most notably on ToolE, where Recall@1 drops from 62.3 to 58.4. On Spider and BIRD, the impact on Recall@1 is smaller, reflecting that the correct item is often already ranked near the top in these datasets. To further analyze this effect, we report Recall@k for $k \in [3, 5, 10]$ in Appendix D. These results show that positional debiasing mainly improves the relative ordering of top-ranked items, even when changes in Recall@1 are modest.

## 5 Conclusions

In this paper, we addressed the challenge of unreliable item selection in LLMs with long-context prompts by proposing a robust and efficient alternative that instead harnesses the internal attention patterns of the LLM. Two salient features of our proposed method OLR-heads are: (1) A small number of examples presented in-context suffice to do head selection, in contrast to previous work that depend on offline training. (2) A simple idea of debiasing with in-context anchor tokens suffices for position debiasing, in contrast to previous work that either require a second LLM call or modify default attention computations. Across seven datasets we demonstrate that OLR-heads provides superior accuracy and robustness compared to existing methods that take comparable time. Crucially, our approach is read-only and preserves the existing KV cache, making it seamlessly compatible with standard LLM workflows.

## 6 Limitations

While our work demonstrate the effectiveness and robustness of OLR-heads, some limitations remain. First, although we empirically show that a sparse set of retrieval-relevant attention heads is sufficient for item selection, we do not analyze the internal mechanisms that make these heads effective. A deeper interpretability analysis of how and why specific heads capture retrieval signals is left for future work.

Second, our experiments are limited to English-language datasets for tool and database selection. Since LLM behavior and attention patterns can vary across languages, the effectiveness of OLR-heads in multilingual or cross-lingual settings remains an open question.

Finally, our evaluation focuses on single-item selection, where each query is associated with one correct item. While OLR-heads naturally produces a ranked list of candidates, we do not explicitly evaluate its performance on multi-item selection or full re-ranking tasks. Extending the method to settings that require selecting or ranking multiple relevant items is a promising direction for future work.

## References


Shijie Chen, Bernal Jiménez Gutiérrez, and Yu Su. 2025. Attention in large language models yields efficient zero-shot re-rankers. *Preprint*, arXiv:2410.02642.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models.

Rishi Gangi Reddy, Niranjan Balasubramanian, and 1 others. 2024. FIRST: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2023. Metatool benchmark: Deciding whether to use tools and which to use. *arXiv preprint arXiv: 2310.03128*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357.

Xianming Li and Jing Li. 2024. Angle-optimized text embeddings. *Preprint*, arXiv:2309.12871.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2025a. Leveraging passage embeddings for efficient listwise reranking with large language models. In *Proceedings of the WWW '25: The Web Conference 2025*.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025b. Sliding windows are not the end: Exploring full ranking with long-context large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, abs/1908.10084.

Patrick Schilcher, Dominik Karasin, Michael Schopf, Haisam Saleh, Antonela Tommasel, and Markus Schedl. 2025. Characterizing positional bias in large language models: A multi-model evaluation of prompt order effects. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. Reasonir: Training retrievers for reasoning tasks. *Preprint*, arXiv:2504.20595.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. 2025. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. *arXiv preprint arXiv:2503.01763*.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2024. Is chatgpt good at search? investigating large language models as re-ranking agents. *Preprint*, arXiv:2304.09542.

Linh Ha Tran, Yulong Li, Radu Florian, and Wei Sun. 2025. Contrastive retrieval heads improve attention-based re-ranking.

Tianshu Wang, Hongyu Lin, Xianpei Han, Le Sun, Xiaoyang Chen, Hao Wang, and Zhenyu Zeng. 2023. Dbcopilot: Scaling natural language querying to massive databases. *arXiv preprint arXiv:2312.03463*.

Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham M. Kakade, Hao Peng, and Heng Ji. 2025. Eliminating position bias of language models: A mechanistic approach. In *The Thirteenth International Conference on Learning Representations*.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *ArXiv*, abs/2404.15574.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Zikai Xiao, Ziyang Wang, Wen Ma, Yan Zhang, Wei Shen, Yan Wang, Luqi Gong, and Zuozhu Liu. 2025. Mitigating posterior salience attenuation in long-context llms with positional contrastive decoding. *Preprint*, arXiv:2506.08371.

Konstantin Yakovlev, Sergey Nikolenko, and Andrey Bout. 2024. Toolken+: Improving llm tool usage with reranking and a reject option. *arXiv preprint arXiv:2410.12004*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 39 others. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

Soyoung Yoon, Dongha Ahn, Youngwon Lee, Minkyu Jung, HyungJoo Jang, and Seung-won Hwang. 2025. RoToR: Towards more reliable responses for order-invariant inputs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18739–18760, Vienna, Austria. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *Preprint*, arXiv:1809.08887.

Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. 2025. Mitigate position bias in large language models via scaling a single dimension. *Preprint*, arXiv:2406.02536.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. 2025. Query-focused retrieval heads improve long-context reasoning and re-ranking. In *Proceedings of EMNLP*.

Zheng Zhang, Fan Yang, Ziyan Jiang, Zheng Chen, Zhengyang Zhao, Chengyuan Ma, Liang Zhao, and Yang Liu. 2024. Position-aware parameter efficient fine-tuning approach for reducing positional bias in llms. *Preprint*, arXiv:2404.01430.

Youxiang Zhu, Ruochen Li, Danqing Wang, Daniel Haehn, and Xiaohui Liang. 2025. Focus directions make your language models pay more attention to relevant contexts. *ArXiv*, abs/2503.23306.

## A Experimentation Details

| Dataset | Prompt Len. | Avg. Item Len. | Items | Queries |
|---|---|---|---|---|
| ToolE | 5932 | 28 | 199 | 1982 |
| Spider | 42360 | 253 | 166 | 982 |
| Bird | 35441 | 439 | 80 | 981 |
| APIbank | 25393 | 248 | 101 | 85 |
| Tooleyes | 28633 | 298 | 95 | 80 |
| craft-algebra | 64717 | 228 | 280 | 236 |
| craft-Tabmwp | 54637 | 301 | 180 | 147 |

Table 8: Dataset statistics.

Across all experiments, we use a fixed number of in-context examples unless stated otherwise. For ToolE, Spider, and Bird, we first sample a pool of N=200 queries. For each test query, we select $K = 5$ in-context examples from this pool, either uniformly at random or based on similarity of cosine embeddings calculated using using the `BAAI/bge-small-en-v1.5` model (Xiao et al., 2023).

For the remaining datasets-APIBank, ToolEyes, CRAFT-Algebra, and CRAFT-TabMWP-the number of available queries is smaller. Accordingly, we set the pool size to 16, 15, 44, and 27, respectively, while keeping the number of in-context examples fixed at $K = 5$.

For consistency across attention-based methods, we fix the number of selected heads to 20 for all models and baselines. As shown in Table 10, performance is largely insensitive to this choice.

By default, items are represented using a string-based format, where each item is described by its name followed by its description(Figure 2[Top]). Alternative representations are analyzed in the ablation studies Table 2.

For the QRHEAD baseline, we follow the original protocol of offline head selection. Specifically, for each dataset, we first sample $K = 5$ queries from the same pool of size $N$ used in our experiments. These queries are used once to identify the top attention heads. The selected heads are then fixed and reused for all test queries within the same dataset. No further head selection or adaptation is performed at inference time.

## B Details about per-query rank variance

For generative baselines using the `id_bottom` prompt variant (Figure 2[Bottom]), where each candidate is assigned a unique numeric identifier and the model is instructed to output the identifier at the end of the response, we derive a ranked list directly from the model's next-token probabilities. After detecting the identifier prefix token("["), we rank all candidates by the log-probability assigned to their identifier token under the model's next-token distribution. This yields a top-K item rankings per query.

## C Performance of standard sparse and dense retrieval methods

Although our method is designed for in-context retrieval, for reference we provide results with the following standard document retrievals baselines. (1) BM25 - a classic lexical retriever, (2) strong dense retrievers such as ms-marco-MiniLM-L6-v2 (Reimers and Gurevych, 2019), a retriever trained on MS MARCO, (3) UAE (Li and Li, 2024) - a bidirectional encoder model trained with an additional angular loss to improve embedding discrimination, and (4) ReasonIR (Shao et al., 2025) - a LLaMA-8B backbone trained with bidirectional attention for retrieval tasks. For both ms-marco and UAE, the input length is truncated to 512 tokens.

Results appear in Table 7.

## D Positional Debiasing Analysis

Table 9 shows the effect of positional debiasing using Recall@k metrics for $k \in \{3, 5, 10\}$. As discussed in Section 4.6 positional debiasing is applied both during retrieval head detection and at inference time.

Across all datasets, positional debiasing consistently improves Recall@k. On ToolE, positional debiasing yields improvements across all recall levels, indicating more reliable ranking of the correct tool. On Spider and Bird, while Recall@1 remains sees small gains, Recall@3, Recall@5, and Recall@10 show consistent gains, suggesting that positional debiasing primarily affects the relative ordering among top-ranked candidates.

| Dataset | Debiasing | R@1 | R@3 | R@5 | R@10 |
|---|---|---|---|---|---|
| ToolE | Both | 62.3 | 76.7 | 81.2 | 85.5 |
| ToolE | None | 58.4 | 74.9 | 79.4 | 83.7 |
| Spider | Both | 65.8 | 79.7 | 85.5 | 91.8 |
| Spider | None | 65.9 | 79.5 | 85.1 | 91.4 |
| BIRD | Both | 72.1 | 86.3 | 89.3 | 92.5 |
| BIRD | None | 71.5 | 85.4 | 88.7 | 92.9 |

Table 9: Recall@k comparison with and without positional debiasing.

| Method | ToolE | Spider | Bird | APIbank | Tooleyes | Algebra | Tabmwp |
|---|---|---|---|---|---|---|---|
| **OLR-heads** | | | | | | | |
| Llama-3.1-8B-Instruct | 62.5 | 67.0 | 73.4 | 77.6 | 68.8 | 60.2 | 19.0 |
| Qwen-2.5-7B-Instruct | 61.7 | 59.4 | 65.4 | 76.5 | 55.0 | 52.1 | 17.7 |
| Qwen-2.5-7B-Instruct-1M | 62.9 | 69.0 | 71.6 | 80.0 | 62.5 | 58.5 | 22.5 |
| **Single vector representation** | | | | | | | |
| msmarco-MiniLM | 49.3 | 65.5 | 60.6 | 56.5 | 52.5 | 39.4 | 12.2 |
| UAE-large-v1 | 54.4 | 54.2 | 59.6 | 65.9 | 55.0 | 54.2 | 14.3 |
| reason-IR | 65.7 | 63.4 | 71.2 | 71.8 | 65.0 | 73.7 | 25.2 |
| BM25 | 19.4 | 20.9 | 22.2 | 24.7 | 23.8 | 40.3 | 10.9 |

Table 7: Performance comparison of single-vector representation models across datasets. We also report results for OLR-heads under all three model variants. Algebra and Tabmwp denote the Algebra and TabMWP subsets of the CRAFT benchmark, respectively.

## E Determine the Number of Heads

Table 10 analyzes the sensitivity of OLR-heads to the number of selected attention heads and the number of in-context examples $K$. Varying the number of selected heads has limited impact on performance across the three datasets. Performance saturates with a small number of heads; for example, under the LLaMA backbone, selecting 10–20 heads (out of 1,024 total heads) is sufficient, indicating that retrieval-relevant signals are concentrated in a small subset of attention heads.

| #Heads | ToolE | Spider | Bird |
|---|---|---|---|
| 10 | 62.3 | 65.5 | 71.9 |
| 20 | 62.3 | 65.8 | 72.1 |
| 40 | 62.4 | 65.6 | 71.7 |

Table 10: Performance with different numbers of selected heads.

## F RankGPT Sliding Window Settings

We evaluate three ranking configurations.
(1) RankGPT(with SW): Ranking is performed using a sliding window with window size $w = 20$ and stride $s = 10$.
(2) RankGPT(constrained SW): the window size $w$ and stride $s$ are adjusted per dataset such that the total number of LLM calls does not exceed a fixed budget of 5, as summarized in Table 11.
(3) RankGPT(without SW): Ranking is performed in a single step over the full candidate set, equivalent to setting $w = N$, where $N$ denotes the total number of items.

| Dataset | Items | Window Size ($w$) | Stride ($s$) |
|---|---|---|---|
| ToolE | 199 | 80 | 40 |
| Spider | 166 | 80 | 40 |
| Bird | 80 | 40 | 20 |
| APIbank | 101 | 50 | 25 |
| Tooleyes | 95 | 48 | 24 |
| CRAFT-Algebra | 280 | 100 | 50 |
| CRAFT-TabMWP | 180 | 80 | 40 |

Table 11: Dataset specific sliding window parameters used for incremental ranking.

<|start_header_id|>user<|end_header_id|>
Here are all the available tools:

tool_id: HouseRentingTool
tool description: Tool that provide all sorts of information about house renting

tool_id: RestaurantBookingTool
tool description: Tool for booking restaurant

tool_id: keyplays_football
tool description: Latest live soccer standings, results, commentary, tv stations, keyplays (with and without scores).

...

Now, follow these in-context examples to understand the task and format.

Query: Could you please tell me the current price of Bitcoin according to the latest news?
Correct tool_id: FinanceTool

Query: Can you give me financial information about a UK company?
Correct tool_id: CompanyInfoTool

Query: How many emails have I sent to my colleague named 'John' during past week?
Correct tool_id: EmailByNylas

Now, please output ONLY the correct tool_id for the query below.

Query: I'm curious about the pay-per-click ads that my competitor is using. Can you show me the latest ones?
Correct tool_id:<|eot_id|><|start_header_id|>
assistant<|end_header_id|>

Figure 3: A sample prompt used by OLR-headson the ToolE dataset with Llama-3.1-8B-Instruct. The item representation uses the string variant. The anchor text (AT) corresponds to the text between the final item and the first in-context example (beginning with "Now, follow ...").