Embrace Positional Bias: Place Good Documents in Good Positions

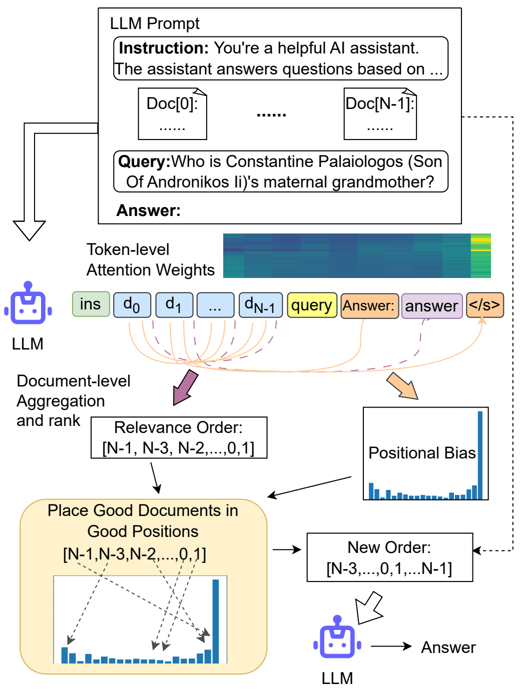
Anonymous ACL submission

Abstract

LLM is unable to treat the information at each position in the prompt fairly, and there is a U-shaped positional bias, which is manifested by paying more attention to the beginning and the end and ignoring the middle, also known as the Lost-in-the-Middle phenomenon. In this paper, we study this phenomenon from the internal state of the model. We examine the effect of different positions on the attention weight of document-level aggregation within the model, both horizontally and vertically, thus reflecting the effect of positional bias on the estimation of document importance in the model. Based on our findings, we propose U-shaped Placement to separate the effects of position and place documents according to positional bias. Combining the U-shaped Placement with the importance estimations of documents within the model, placing good documents in good positions, can improve the model's ability to utilize documents within two iterations. Experimental results prove that our method can outperform other baselines and improve model's ability to utilize documents on various models and datasets. Our codes are submitted with the paper and will be publicly available.

1 Introduction

As large language models(LLM) continue to evolve, they have achieved superior performance in many tasks, especially in Question Answering (QA) tasks (Touvron et al., 2023; Achiam et al., 2023; DeepSeek-AI, 2025). Furthermore, Retrieval Augmented Generation(RAG) has become a widely recognized paradigm by supplementing the model with external knowledge in the form of context, which helps to improve the factualness and accuracy of the answers (Gao et al., 2023; Asai et al., 2023). However, the quality of input documents is variable (Shi et al., 2023; Yoran et al., 2024; Wu et al., 2024) due to the inadequate performance of the retriever (Yan et al., 2024) or the alignment gap between the retriever and the generator (Ke et al., 2024; Li and Ramakrishnan, 2025).



LLM Prompt
Instruction: You're a helpful AI assistant. The assistant answers questions based on ...
Doc[0]: ......
......
Doc[N-1]: ......
Query: Who is Constantine Palaiologos (Son Of Andronikos Ii)'s maternal grandmother?
Answer:
Token-level Attention Weights
LLM
ins
d0
d1
...
dN-1
query
Answer:
answer
</s>
Document-level Aggregation and rank
Relevance Order: [N-1, N-3, N-2,...,0,1]
Positional Bias
Place Good Documents in Good Positions
[N-1,N-3,N-2,...,0,1]
New Order: [N-3,...,0,1,...N-1]
LLM
Answer

Figure 1: The framework of our proposed pipeline. See introduction and section 4 for more details.

How to improve a model's ability to utilize documents with inputs of varying quality is a challenging and realistic research topic, and this is also part of the model robustness problem (Shi et al., 2023; Yoran et al., 2024; Zhou et al., 2025). Previous works improve the robustness of model by incorporating irrelevant and interfering documents into the supervised fine-tuning process (Pan et al., 2024; Yoran et al., 2024; Tu et al., 2025), which is customized and requires additional training resources. Instead of direct training, we focus on the model's properties of prompt utilization, especially the ability to use documents in different positions.

The retrieval results reflect the importance of different documents through the retrieval ranking and positional order in the prompt (Gao et al., 2023). But the language model has a positional bias for information in the prompt, which is characterized by a U-shaped curve, i.e., it pays more attention to information at the beginning and the end and tends to ignore those in the middle. This phenomenon was first proposed in Liu et al. (2024) and verified in Cuconasu et al. (2024) and Wu et al. (2024) from a performance perspective in RAG tasks. However, there is a lack of research on U-shaped curves from the internal state of the model.

In this paper, we first investigate the bias towards documents in different positions from the perspective of the internal characteristic of the LLM. We examine the effect of positions on the attention weights of the LLM from both horizontal and vertical perspectives, by constructing different prompts and exploring attributes of different layers of the model. The value of attention weight as a whole reflects the document relevance and positional influence on the target (Peysakhovich and Lerer, 2023; Chen et al., 2024; Liu et al., 2025). We separated the effect of position and verified that it also conforms to the U-curve characteristic .

Based on the above findings, we propose U-shaped Placement to reorder the documents to conform to the positional bias. And we combine it into a two-round iterative generation method that modifies the inputs based on the internal states of the model, as illustrated in figure 1. Specifically, we calculate the importance scores of different documents within the model based on the attention weights from the first round of generation, obtain the positional bias of the model at the same time, and then collaboratively use the importance score and positional bias to modify the input prompts for the second round so that the model can maximize the use of the documents it considers most relevant. We conduct comprehensive experiments on several multi-document QA datasets utilizing several commonly used LLMs, demonstrating that our method consistently outperforms baselines and achieves superior response quality. Our method requires no additional training and can be easily migrated to any dataset and model.

2 Related Works

2.1 Retrieval Augmented Generation

Retrieval-augmented generation (RAG) has exhibited significant effectiveness in addressing issues such as hallucinations by introducing external knowledge into context or training objectives (Gao et al., 2023; Asai et al., 2023; Tu et al., 2025; Luo et al., 2024, 2025). However, irrelevant and distracting information can adversely affect the generated results (Shi et al., 2023; Yoran et al., 2024; Wu et al., 2024). Previous work has explored various improvement strategies, such as improving the retriever (Shi et al., 2024), designing new rerankers (Kim and Lee, 2024), investigating gaps between the retriever and generator (Ke et al., 2024; Li and Ramakrishnan, 2025), and improving the robustness of the LLM, especially interference resistance (Xiang et al., 2024; Yoran et al., 2024). Rather than directly adding documents to the training or supervised fine-tuning process (Pan et al., 2024; Yoran et al., 2024; Tu et al., 2025), we study the internal utilization characteristic of documents at different positions and dynamically modify inputs based on positional bias to improve robustness.

2.2 Document Relevance

In RAG pipeline, the external retriever or reranker will give a relevance score, and the different importance would be reflected mainly by the positional order, rather than the value itself (Gao et al., 2023; Shi et al., 2024; Kim and Lee, 2024). Including the relevance score into the prompts may affect the generated results (Pan et al., 2024), but this requires a high level of the instruction-following ability. In addition to utilizing externally given relevance scores, there are also some works that let the model itself give a judgment on the relevance of documents through prompt engineering (Qin et al., 2024; Sun et al., 2023; Niu et al., 2024), adding probing structures (Baek et al., 2024; Wang et al., 2024), or internal attention weight (Peysakhovich and Lerer, 2023; Chen et al., 2024; Liu et al., 2025). We also use attention weights as the basis for model importance estimation for documents, but we compute them differently and further combine them with positional bias to optimize the inputs.

2.3 Positional Bias

The LLMs are unable to treat the information from each position in the prompt equally and have a positional bias, which is part of the model's prompt-sensitive properties (Xie et al., 2024). It tends to pay more attention to information at the beginning and the end, and to ignore those in the middle, which is characterized by a U-shape curve. This "Lost in the Middle" phenomenon was first proposed in Liu et al. (2024). To date, many RAG

Prompt	Vicuna-7b			Llama-3.1-8b			Qwen2.5-7b			Qwen2.5-7b-ins		
	H	M	W	H	M	W	H	M	W	H	M	W
concat	0.392	0.238	0.452	0.292	0.192	0.35	0.376	0.298	0.39	0.468	0.458	0.48
rerank	0.4	0.312	0.482	0.302	0.238	0.358	0.408	0.342	0.41	0.472	0.5	0.526

Table 1: Zero-shot model performance in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024). Concat means placing documents in random order, while rerank means sorting by retrieval scores and placing the relevant ones close to the questions.

and long text-related works (Cuconasu et al., 2024; Wu et al., 2024; Xu et al., 2024) have investigated this positional bias issue from a downstream performance perspective. We study this phenomenon from internal attention weight of the LLM both horizontally and vertically, offering a new perspective to investigate the U-shaped positional bias, and we propose a method to take advantage of positional bias during the generation process.

3 Investigation on Positional bias

In this section, we take an in-depth look at the model's positional bias and relevance assessment for documents at different prompt locations. We build on intuitive performance differences to further investigate positional bias and relevance in terms of the internal states of the model, especially the attention weights.

3.1 Notations

For each data, we use q to present the question. The retrieval documents are denoted as $D = \{d_0, d_1, ..., d_{N-1}\}$, where d_i is a single document, and N is the total number of documents. $x = \{x_0, x_1, ..., x_{k-1}\}$ is the input of large language models, which is constructed based on q, D, and a certain prompt template T, where k is the number of tokens contained in the input, i.e., the token length. And the output is indicated as $y = \{y_0, y_1, ..., y_{m-1}\}$ while the large language model is presented as θ and generates each token in answer y with auto-regressive style, and m is the token length of the output answer.

3.2 Preliminary Experiments

Rather than manually adjusting the position of relevant documents in the prompt as in previous work (Cuconasu et al., 2024; Wu et al., 2024), we focus on examining the difference in model performance when the documents are not ordered and when they are ordered by external correlation, as these are the two most common settings for the RAG task and real-world scenarios.

Datasets Therefore, we apply the datasets processed by Pan et al. (2024), as they provide processed randomized (denoted as *concat*) and sorted (denoted as *rerank*) versions, which can reduce the randomness introduced in the processing process. Due to resource constraints, we performed experiments on only three of these datasets, HotpotQA (Yang et al., 2018), Musique (Trivedi et al., 2022) and 2WikiMHQA (Ho et al., 2020), which are popular open-domain RAG datasets with multiple documents as input. The details of datasets can be found in the original paper or Appendix A.1.

Models We test four popular open-source LLMs: Vicuna-7B (Chiang et al., 2023), Llama-3.1-8B (Dubey et al., 2024), Qwen2.5-7B and Qwen2.5-7B-Instruct (Yang et al., 2024).

Metrics We follow Pan et al. (2024) and utilize Exact Match (EM) as the primary evaluation metric by checking whether the short answers provided are exact substrings of the generation. The hyperparameters such as temperature and instruction are the same as in Pan et al. (2024). However, we conduct the experiments on zero-shot setting to investigate the positional bias and relevance assessment inside of the model. See the appendix A.2 for prompt templates and implementation details.

The results of the EM values are presented in Table 1. The main reason for the inconsistency with the results in Pan et al. (2024) is that we used a zero-shot setting, which is not the same as the original paper that used several Q&A pairs as demonstration, and there are also issues with different versions of huggingface and devices. The results show that although different models perform differently on different datasets, either model is unable to utilize the information from each position equally, and positional bias exists. From the dataset perspective, the HotpotQA is less sensitive to document order, while the Musique dataset has 20 documents and changing the document order has a significant impact. An ordered placement approach such as placing relevant documents close to the questions

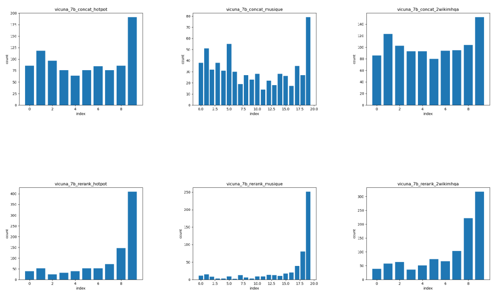


Figure 2: The MIRAGE results of vicuna-7b model. The different columns represent different datasets: HotpotQA, Musique, 2wikiMultiHopQA. The first and second lines represent the concat and rerank inputs. See the section 3.2 for the exact meaning of each figure.

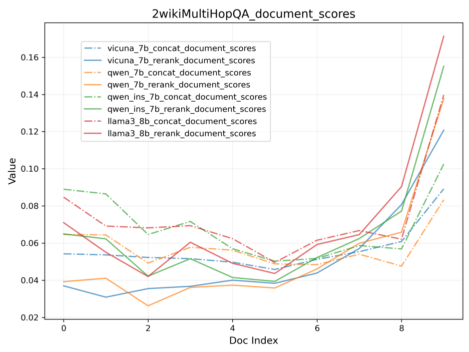


Figure 3: The document scores $S_{y,L_{all}}$ of all models on 2wikiMultiHopQA datasets. The solid line - corresponds to the rerank input, the dashed -. line corresponds to the concat input, and the results of the same model are shown in the same color.

is a powerful baseline (i.e., rerank), but we want to make better use of the model's positional bias.

Therefore, we first explore the dependency of the generated answer token on the context documents using the library developed by Qi et al. (2024). MIRAGE detects context-sensitive answer tokens and pairs them with retrieved documents based on model internals, and we count the position of the context on which the answer token depends. For better illustration, we only present the result of vicuna-7b model in figure 2 and leave the complete in the appendix B. In each figure, the horizontal axis coordinates 0 through $N-1$ indicate the different positions of the documents in the prompt, with 0 placed at the beginning and $N-1$ placed at the end, closest to the question. The vertical bar is the result of the count, indicating how many answer tokens depend on the document at this position.

The results show that under rerank input, it is common sense to depend on the documents near the question. In contrast, it shows a clear positional bias towards the beginning and the end under the concat input, which matches the Lost-in-the-middle (Liu et al., 2024) phenomenon in performance. And this is more evident on the Musique which has a larger number of documents.

3.3 Attention Weight

The MIRAGE metric reflects the positional bias of the model in a U-shaped curve by examining the dependency of the answer and context tokens, but it is a back-end metric that needs complete generation before analysis, so it cannot reflect internal state during the generation. Attention weights essentially reflect, at the token level, the influence of context tokens on answer tokens during the generation. In order to obtain the influence of context on answer tokens at document level, we aggregate the attention weight as follows:

$$s_{i,y,L_s} = \frac{1}{m}\frac{1}{H}\frac{1}{|L_s|}\sum_{l\in L_s}\sum_{h=1}^{H}\sum_{j\in y} a_{i,j}^{l,h} \quad (1)$$

$$S_{d_i,y,L_s} = \frac{1}{|d_i|_t}\sum_{i\in d_i} s_{i,y,L_s} \quad (2)$$

where $a_{i,j}^{l,h}$ denotes the attention weight from the token i (from the document d_i whose token length is $|d_i|_t$) to the token j (from the answer y whose token length is m) by the attention head h at layer l, H is the total number of attention heads, and L_s is the set of selected layers. After obtaining the influence score of each token in the document on the answer s_{i,y,L_s} at token level we then aggregate and normalize by removing the influence of length to obtain attention weight value at document level for the answer S_{d_i,y,L_s}. $S_{y,L_s} = \{S_{d_0,y,L_s}, ..., S_{d_{N-1},y,L_s}\}$ is the overall set of document scores.

We examine the effect of position on document scores from both horizontal and vertical perspectives. We hope to synthesize the two dimensions to get the effect of different positions of documents on the answers within the model. Specifically, we examine the difference between the document scores of different positions under different inputs, which is horizontal. We first set L_s to all layers and calculate $S_{y,L_{all}}$. We randomly sample 50 samples from the 2wikiMultHopQA dataset to average the results and visualize them in figure 3 for better illustration. The complete results of all models

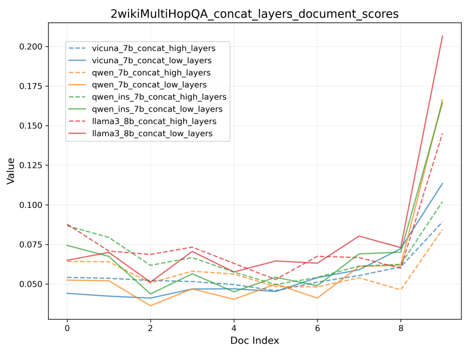


Figure 4: The document scores S_{y,L_s} of all models with different selected layers on 2wikiMultiHopQA datasets under concat input. The solid - and dotted – lines are used to distinguish the low and high half of layers.

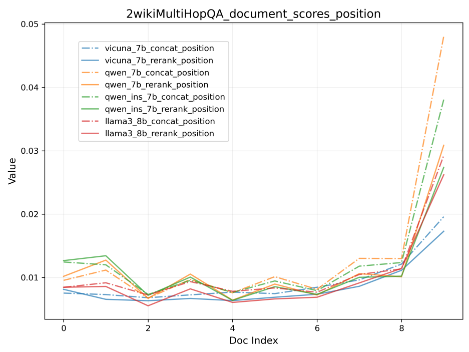


Figure 5: The positional scores $S_{\{t_b,t_e\},L_l}$ on 2wikiMultiHopQA datasets which are calculated by aggregating the document scores of the previous token t_b and the terminating token t_e of the answer tokens.

on all datasets are presented in the appendix C.1. The results show that document scores $S_{y,L_{all}}$ follow a U-shaped curve in different positions under rerank and concat input, but under rerank it is more skewed towards the back end near the question, with a steeper U-shape, while under concat the U-shape is more gentle, and the differences between the beginning, the end, and the middle exist but are not as significant as under rerank. This means that the model's estimation of document importance is internally affected by positional bias in a U-shape, with the magnitude of the U-shape varying with the order of input documents.

And we examine the effect of different selected layers under the same prompt, which is vertical. We split all layers into two halves, low and high, and examine the differences in document scores with different selected layers. For better illustration, we present the result of concat input on 2wikiMultiHopQA datasets in figure 4 as document relevance has less impact on concat than rerank and the U-shaped positional bias is more obvious under concat as said before, and the complete results are presented in the appendix C.2. The results show that for the same model, the document scores computed with the lower and higher half layers have different values but have a similar U-shaped trend, with the distinction between different positions being more significant at the lower layers. This matches generally accepted consensus that the model processes semantic information at higher layers and is more affected by positions at lower layers.

4 Separate and Utilize Positional Bias

The attention weight reflects the overall influence of context tokens on answer tokens, including semantic relevance and positional influence. Previous analysis utilized concat to shuffle the order of documents to exclude semantic influence, but how to directly obtain the positional influence in the generation process is a problem worth studying.

Previous work mentioned using meaningless query (Chen et al., 2024), but this requires one more round of LLM calls, and we are aggregating answer tokens instead of query tokens. Therefore, we choose to aggregate the previous token and the terminating token of the answer, because we have observed that the previous token of the answer token is always a token with no semantic information (e.g. *is* or *:*). As mentioned before, the lower layers reflect the position information more significantly, so we choose the lower layers (i.e., L_l) for aggregation. Therefore, we get the positional information related to the beginning and end of the answer tokens, and combine them to represent the overall positional information of whole answer tokens. We similarly visualize the scores $S_{\{t_b,t_e\},L_l}$ of 2wikiMultiHop in figure 5. See appendix C.3 for results of other datasets. And the results show no significant difference between the concat and rerank inputs compared to figure 3, suggesting that the combination of the beginning and end can somewhat represent the overall positional information of whole answer, independent of the document order.

After separating the positional bias, we hope to use it to improve the model's ability to utilize documents. Placing documents directly according to the obtained $S_{\{t_b,t_e\},L_l}$ from preferred to non-preferred is a feasible approach, but there will be a length problem. $S_{\{t_b,t_e\},L_l}$ is the result of length normalization, the length of the position corresponding to each score is not necessarily equal, and direct placement will be problematic. For example, the position with the highest score may only contain 100 token positions, and placing a document with more than 100 tokens will use the positions corresponding to other scores. In order to solve the length issue, we arrange documents directly based on token level scores $s_{i,\{t_b,t_e\},L_l}$ instead of document level $S_{\{t_b,t_e\},L_l}$. The motivation is placing good documents in good positions. The specific algorithm is as follows, considering the documents from relevant to irrelevant, using the U-shaped feature to determine the position of each document, calculating the scores placed at the beginning(left) and the end(right) of the remaining length, and placing the document on whichever side has a higher score until all the documents are placed in the U-shape. The U-shaped Placement is summarized in the form of pseudocode in Algorithm 1.

Algorithm 1 U-shaped Placement

```
Input Relevance ranking $R$, Attention weight $A_\theta$, previous token $t_b$ and the terminating
      token $t_e$ of the answer tokens, the collection of token lengths for all documents
      $T_l = \{T_0, ..., T_{N-1} | T_i = |d_i|_t\}$.
 1: Ensure that the relevant ones in $R$ come first;
 2: Get $S_{pbe}$ = $\{s_{i,\{t_b,t_e\},L_l}\}$ based on $A_\theta$;  ▷ (1)
 3: Initialization: $l = 0, r = \sum_{i=0}^{N-1} T_i, l_{idx} = 0, r_{idx} = N-1, R_u = [0] * N$;
 4: for i $\in R$ do
 5:     $T_i = T_l[i]$;
 6:     Left_Scores = $S_{pbe}[l : l + T_i]$.sum();
 7:     Right_Scores = $S_{pbe}[r - T_i : r]$.sum();
 8:     if Right_Scores $\geq$ Left_Scores then
 9:         $R_u[r_{idx}] = i, r_{idx} = r_{idx} - 1, r = r - T_i$;
10:     else
11:         $R_u[l_{idx}] = i, l_{idx} = l_{idx} + 1, l = l + T_i$;
12:     end if
13: end for
Output: The new ranking $R_u$
```

The U-shaped Placement approach can be combined with all kinds of document ranking methods. We combine it with the previously obtained document scores that are aggregated based on the answer tokens, and modify the inputs for the next round, thus improving the overall ability of the model to utilize documents. The whole pipeline is summarized in the form of pseudocode in Algorithm 2.L_l and L_h denote the low and high half of all layers, respectively.

Algorithm 2 Place Good Documents in Good Positions

```
Input Prompt Template $T$, LLM $\theta$, Question $q$,
      Documents $D = \{d_0, ..., d_{N-1}\}$.
 1: Construct input: $x = T(q, D)$;
 2: Get the output from LLM: $y, A_\theta = \theta(x)$;
 3: Calculate $S_{y,L_h}$; ▷ (2)
 4: Rank the documents based on $S_{y,L_h}$ as $R_a$;
 5: Get token lengths of each document from $x$ to
    construct $T_l$, and locate $t_b$ and $t_e$ from $x$;
 6: $R_u$= U-shaped Placement($R_a, A_\theta, t_b, t_e, T_l$);
 7: Reconstruct the input based on $R_u$: $x_u = T(q, R_u(D))$;
 8: Get the final answer: $y = \theta(x_u)$
Output: The output answer $y$
```

The algorithm is essentially two rounds of iterations of the LLM, using the attention weight from the first round to obtain the model's ranking of the documents and positional bias, and then placing good documents in good positions according to the U-shape, and reconstructing the inputs to generate the final answer in the second round.

5 Experiments

5.1 Baselines

The basic setting of the experiment is the same as preliminary experiments in section 3.2, including the datasets, models, metrics, and so on.

Our work is essentially a two-round iteration of the LLM, so we mainly consider similarly set-up baselines for fair comparison, and the following is a brief description of the baselines we consider: **(1)Vanilla**: The most basic baseline, generating answers directly based on inputs. **(2)RankGPT** (Sun et al., 2023): Two rounds of iteration, the first round uses the model to sort the documents in list-wise style and the second round generates the answer. The prompt template used in the first round is shown in the appendix D. **(3)Attention Sorting** (Peysakhovich and Lerer, 2023): Two rounds of iteration, average per-document attention is computed for the first generated token in the first round, and then documents are sorted based on the attention scores for the second round. **(4)ICR** (Chen et al., 2024):Two rounds of iteration, the first round aggregates the contextual attention weight corresponding to all query tokens and calibrates it with the meaningless query to get the document order, and the second round generates the answers based on the reordered document. **(5)SELFELICIT** (Liu

Prompt	Methods	Vicuna-7b			Llama-3.1-8b			Qwen2.5-7b			Qwen2.5-7b-ins		
		H	M	W	H	M	W	H	M	W	H	M	W
concat	Vanilla	0.392	0.238	0.452	0.292	0.192	0.35	0.376	0.298	0.39	0.468	0.458	0.48
	RankGPT	0.401	0.22	0.46	0.278	0.196	0.334	0.372	0.343	0.402	0.466	0.51	0.526
	AttentionSort	0.386	0.291	0.484	0.296	0.216	0.336	0.398	0.375	0.4	0.462	0.495	0.484
	ICR	**0.421**	0.305	0.498	0.298	0.237	0.368	0.379	0.385	0.384	0.469	0.511	0.522
	SELFELICIT	0.378	0.257	0.462	**0.308**	0.229	0.359	0.393	0.298	0.43	0.417	0.311	0.382
	Our	0.414	**0.31**	**0.506**	0.302	**0.245**	**0.372**	**0.402**	**0.393**	**0.434**	**0.482**	**0.513**	**0.536**
rerank	Vanilla	0.4	0.312	0.482	0.302	0.238	0.358	0.408	0.342	0.41	0.472	0.5	0.526
	RankGPT	0.403	0.28	0.502	0.284	0.21	0.342	0.391	0.358	0.416	0.493	0.516	0.53
	AttentionSort	0.404	0.3	0.474	0.294	0.218	0.342	0.408	0.385	0.432	0.492	0.485	0.518
	ICR	0.413	0.307	**0.512**	0.301	0.254	0.353	0.406	0.378	0.4	0.487	0.537	0.504
	SELFELICIT	0.393	0.314	0.482	**0.314**	0.261	0.372	0.411	0.342	0.432	0.417	0.447	0.372
	Our	**0.426**	**0.315**	0.51	0.304	**0.267**	**0.378**	**0.412**	**0.401**	**0.436**	**0.495**	**0.555**	**0.542**

Table 2: Zero-shot model performance in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024). See section 5.1 for more details on baselines.

et al., 2025): Two rounds of iteration, average per-sentence attention is computed for the first generated token in the first round and then important sentences are selected to be emphasized with special token in the input for the second round.

5.2 Main Results

The results are shown in Table 2. The results show that: (1) Our method outperforms previous baselines on most datasets and most models whether it is the concat input or the rerank input. (2) The improvement under concat input is larger than that under rerank input, partly because there is more room for improvement under concat, and the results obtained by our method based on concat input can be superior to the vanilla rerank baseline based on the external retrieval ranking, which shows that our method can obtain an effective order of documents from disordered input. Further improvement under rerank input indicates that our method can improve the ability of the model to utilize documents. (3) From the dataset point of view, the improvement of our method is more evident on datasets with a larger number of documents (such as Musique). In terms of models, the improvement is more evident on Qwen models compared to Vicuna-7b model, which is related to the basic capability of the LLM. After all, the information obtained based on the internal state is more reliable if the model capability is strong. However, our method achieves a steady performance gain on various models and datasets.

6 Analysis

Our method consists of two parts: obtaining document order and positional bias based on internal state, and the effect of combining them has been demonstrated in main experiments, so we focus on their roles separately in this section.

6.1 The Influence of Positions

We propose U-shaped Placement to arrange document positions in compliance with positional bias, and it can be combined with any document relevance ordering obtained by arbitrary method.In this section we combine it with external sorting results (i.e., rerank) to better represent its role.

For a better comparison, we consider four inputs: rerank, U-shaped Placement, and the reverse versions of these two methods. In the original versions, the goal is to place good documents in default good positions, while in the reverse versions, default good positions are prioritized by bad documents. All four inputs use the same prompt template, with only the order of the documents differing, and this comparison shows the importance of placement on the results. We've included an example in the appendix E for better explanation. And the results of four inputs are presented in Table 3.

The results show that: (1) Our proposed U-shaped Placement in accordance with positional bias is a better placement method that improves the model's ability to utilize documents. Comparing the results with those in Table 2, it is found that the results on the vicuna-7b and llama-3.1-8b models can approach or even exceed those in Table 2, while there is still a distance for qwen series, which is consistent with the previously mentioned model's capabilities, and document relevance ranking obtained by the qwen series is more reliable. (2) Among the reversed versions, the reversed version of U-shaped Placement resulted in more decreases, indicating importance of embracing positional bias.

Placement	Setting	Vicuna-7b			Llama-3.1-8b			Qwen2.5-7b			Qwen2.5-7b-ins		
		H	M	W	H	M	W	H	M	W	H	M	W
rerank	Original	0.4	0.312	0.482	0.302	0.238	0.358	0.408	0.342	0.41	0.472	0.5	0.526
	Reverse	0.378	0.24	0.464	0.28	0.2	0.348	0.37	0.323	0.39	0.456	0.477	0.5
U-shaped (Our)	Original	0.397	0.314	0.524	0.31	0.252	0.376	0.416	0.369	0.414	0.486	0.509	0.522
	Reverse	0.367	0.216	0.446	0.268	0.186	0.338	0.368	0.3	0.39	0.454	0.435	0.476

Table 3: Results of different placements with relevance ranking based on external retrieval. Rerank means placing the relevant ones close to question, while U-shaped is our method in accordance with positional bias. Different settings indicate whether a good position is preferentially occupied by a good (original) or bad (reverse) document.

Ranking	Aggregation	H	M	W
Retrieval	-	0.4	0.312	0.482
Attention Weight	First Token	0.404	0.291	0.48
	Query	0.398	0.301	0.49
	Answer	0.406	0.305	0.494
	Answer(qwen)	0.412	0.329	0.512

Table 4: Results of the different document relevance sorting methods of vicuna-7b model under rerank input.

6.2 Attention Aggregation

In previous analyses, we aggregated the attention weight of context tokens over answer tokens as document scores as a basis for documents estimation. In this section we explore the impact of different aggregation methods, mainly considering aggregation based on the first generated token or query tokens involved in the previous works.

To exclude positional effect, we use the default rerank placement, placing the relevant ones close to the question. The results of vicuna-7b under rerank input are displayed in Table 4 as a representative.

The results show that: (1) Answer tokens based aggregation is better than query tokens, first token based, because it reflects the direct influence on the answer, although obtaining the attention weight of the context on the full answer tokens requires more computational cost. (2) As mentioned before, the document relevance ranking based on the Qwen models is more reliable, and can improve the performance of vicuna by offering the document relevant ranking to vicuna. This suggests a new way of combining two models into the generation.

6.3 Something We Tried

The complete pipeline of our proposed algorithm is embodied in Algorithm 2, while in this section we briefly describe some additional attempts at details.

The first is the estimation of document relevance. In previous experiments, we directly use the document scores as the basis. The calibration proposed in Chen et al. (2024) is inspiring, so we also try to remove the positional effect from the attention scores. Specifically, we similarly subtract the scores indicating position from the document scores. However, this does not result in a significant improvement, probably because we are using the scores from the answer aggregation, while the position is a representation of the beginning and end, which does not strictly correspond to each other. The results are presented in appendix F.1.

We place the documents according to the U-shape in our proposed method, however, the positional bias does not exactly fit the U-shape and there may be zigzag in the middle, as shown in previous analysis. Aggregating the token-level position scores by document and then placing the document directly according to the result of document-level has no zigzag problem, but it has length problem as said in section 4. Is the length issue more important or the zigzag issue? The results in appendix F.2 show that placement according to the U-shape is more in line with the positional bias, and the length mismatch has a greater impact on performance compared to the zigzag problem.

7 Conclusion

In this paper, we investigate the positional bias based on model's attention weight, both horizontally and vertically. We find that the model's estimation of document importance is also internally affected by positional bias in a U-shape, with the magnitude of the U-shape varying with the order of input documents. In addition, the lower layers reflect the position information more significantly.

And we propose U-shaped Placement to separate and utilize positional bias. Combining it with the importance estimation of documents within the model, placing good documents in good positions, can improve the model's ability to utilize documents within two iterations. Our approach requires no training, and can work on any open-source model and dataset.

Limitations

There are several limitations of our current work that we plan to address in the future:

(1) Our work is based on attention weight to determine the importance judgment of documents and positional bias within the model, so access to the model internals is needed. Consequently, its applicability may be limited to white-box models. Closed-source models, such as ChatGPT and GPT4, may not be compatible with our method due to the lack of access to internal attention weight.

(2) Our work is essentially a two-round iteration of LLM that utilizes the internal information from the first round to modify the inputs of the second round, which increases the inference cost compared to the normal generation process. How to utilize the internal state during one round of generation and modify the generated results directly is our future research direction. In addition, from another direction, it is also possible to increase the number of iterations to make the results more stable and reliable, however, due to resource constraints, we did not do experiments with multiple rounds of iterations in this paper, and we will carry out related research when we have the conditions to do so.

(3) Our method itself also has some points that can be studied in depth, such as the way of aggregation of attention weight, and there may exist better ways of aggregation, like sentence level, and so on. Moreover, we use the meaningless tokens at the beginning and end of the answer as the source of position information in this paper, in which there may be better ways of choosing meaningless tokens, such as aggregating meaningless tokens in the whole answer, constructing meaningless answers, meaningless query, and so on. In this paper, we only propose one feasible way, and there may be more feasible ways to be explored.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and 260 others. 2023. Gpt-4 technical report.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *ArXiv*, abs/2310.11511.

Ingeol Baek, Hwan Chang, Byeongjeong Kim, Jimin Lee, and Hwanhee Lee. 2024. Probing-rag: Self-probing to guide language models in selective document retrieval. *arXiv preprint arXiv:2410.13339*.

Shijie Chen, Bernal Jiménez Gutiérrez, and Yu Su. 2024. Attention in large language models yields efficient zero-shot re-rankers. *Preprint*, arXiv:2410.02642.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for RAG systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 719–729. ACM.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10438–10451, Bangkok, Thailand. Association for Computational Linguistics.

Kiseung Kim and Jay-Yoon Lee. 2024. RE-RAG: improving open-domain QA performance and interpretability with relevance estimator in retrieval-augmented generation. In *Proceedings of the 2024*

Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024, pages 22149–22161. Association for Computational Linguistics.

Sha Li and Naren Ramakrishnan. 2025. Oreo: A plug-in context reconstructor to enhance retrieval-augmented generation. *CoRR*, abs/2502.13019.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173.

Zhining Liu, Rana Ali Amjad, Ravinarayana Adkathimar, Tianxin Wei, and Hanghang Tong. 2025. Selfelicit: Your language model secretly knows where is the relevant evidence. *CoRR*, abs/2502.08767.

Wen Luo, Tianshu Shen, Wei Li, Guangyue Peng, Richeng Xuan, Houfeng Wang, and Xi Yang. 2024. Halludial: A large-scale benchmark for automatic dialogue-level hallucination evaluation. *CoRR*, abs/2406.07070.

Wen Luo, Feifan Song, Wei Li, Guangyue Peng, Shaohang Wei, and Houfeng Wang. 2025. Odysseus navigates the sirens' song: Dynamic focus decoding for factual and diverse open-ended text generation. *CoRR*, abs/2503.08057.

Tong Niu, Shafiq Joty, Ye Liu, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Judgerank: Leveraging large language models for reasoning-intensive reranking. *CoRR*, abs/2411.00142.

Ruotong Pan, Boxi Cao, Hongyu Lin, Xianpei Han, Jia Zheng, Sirui Wang, Xunliang Cai, and Le Sun. 2024. Not all contexts are equal: Teaching LLMs credibility-aware generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19844–19863, Miami, Florida, USA. Association for Computational Linguistics.

Alexander Peysakhovich and Adam Lerer. 2023. Attention sorting combats recency bias in long context language models. *CoRR*, abs/2310.01427.

Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. 2024. Model internals-based answer attribution for trustworthy retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 6037–6053. Association for Computational Linguistics.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 1504–1518. Association for Computational Linguistics.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Huai hsin Chi, Nathanael Scharli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 8371–8384. Association for Computational Linguistics.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Yiteng Tu, Weihang Su, Yujia Zhou, Yiqun Liu, and Qingyao Ai. 2025. Rbft: Robust fine-tuning for retrieval-augmented generation against retrieval defects. *CoRR*, abs/2501.18365.

Yuhao Wang, Ruiyang Ren, Junyi Li, Xin Zhao, Jing Liu, and Ji-Rong Wen. 2024. REAR: A relevance-aware retrieval-augmented framework for open-domain question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 5613–5626. Association for Computational Linguistics.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024. How easily do irrelevant inputs skew the responses of large language models? In *First Conference on Language Modeling*.

Chong Xiang, Tong Wu, Zexuan Zhong, David A. Wagner, Danqi Chen, and Prateek Mittal. 2024. Certifiably robust RAG against retrieval corruption. *CoRR*, abs/2405.15556.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations*.

Huichi Zhou, Kin-Hei Lee, Zhonghao Zhan, Yue Chen, Zhenhao Li, Zhaoyang Wang, Hamed Haddadi, and Emine Yilmaz. 2025. Trustrag: Enhancing robustness and trustworthiness in RAG. *CoRR*, abs/2501.00879.

A Details about Preliminary Experiments

A.1 Datasets

We applied the datasets processed by Pan et al. (2024) in our paper. Due to resource limitations, we mainly focus on several open-domain variants of the datasets.

HotpotQA (Yang et al., 2018) and 2WikiMHQA (Ho et al., 2020) both require reasoning across multiple documents, and feature a high proportion of distracting documents. Importantly, the data from HotpotQA is extracted from the dev subset, whereas our training dataset is derived from the train subset. Musique (Trivedi et al., 2022) questions are of higher complexity, with up to 90% of distracting passages.

A.2 Implementation Details

we will list the details of hyperparameters we used in the experiments. The seed is set to 42. The generation configuration includes, temperate is set to 0.01 and the number of max_new_tokens is 300. The same prompt template is used for all datasets and all models in the experiments to exclude template interference, which is presented as follows:

```
You're a helpful AI assistant. The assistant answers questions based on given passages.

Docs:{{d_0.title}}:{{d_0.text}}
{{d_1.title}}:{{d_1.text}}
{{d_2.title}}:{{d_2.text}}

(more passages) ...

Question: {{question}}

Answer:
```

B Mirage Results

The complete Mirage Results are presented in this section. The results of llama3-8b, qwen-7b, and qwen-7b-instruct are presented in Fig 6, 7, and 8, respectively.

The different rows represent the results on different datasets: HotpotQA, Musique, 2wikiMultiHopQA. The different columns represent the different ways of composing the prompt: concat, rerank, corresponding to Table 1. In each figure, the horizontal axis coordinates 0 to N-1 indicate different positions of the document in the prompt, with 0 placed at the beginning, furthest from the question, and N-1 placed at the end, closest to the problem. The vertical axis is the counting result, indicating how many answer tokens have dependency on the document at this position.

C Attention Weight Results

The complete Attention Weight Results are presented here.

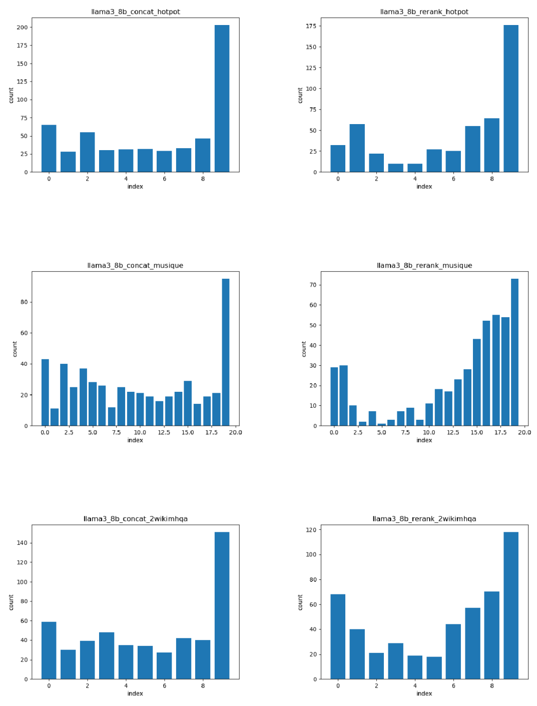


Figure 6: The MIRAGE results of llama3-8b model. See the section 3.2 for more details and analysis.

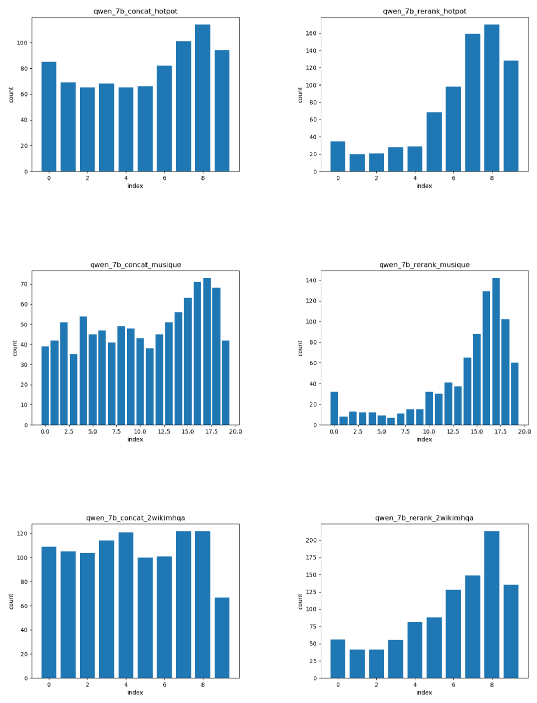


Figure 7: The MIRAGE results of qwen-7b model. See the section 3.2 for more details and analysis.

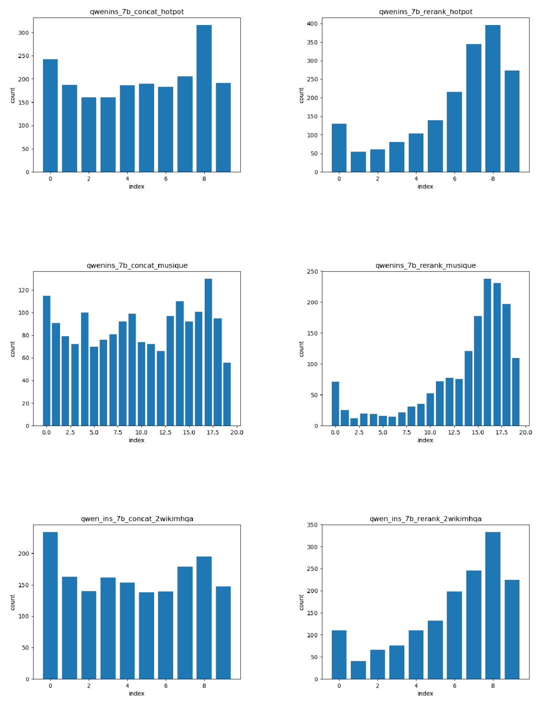


Figure 8: The MIRAGE results of qwen-7b-Instruct model. See the section 3.2 for more details and analysis.

C.1 Horizontal Results

We present the complete results of the difference between the score of different documents in different order of documents on this section. The results on HotpotQA dataset is presented in figure 9, and the results on Musique dataset is presented in figure 10.

C.2 Vertical Results

We present the complete results of different selected layers in this section. See figure 11,12, 13,14,15 for more information.

C.3 Positional Scores

We present the complete results of postional scores on all datasets in this section. See figure 16,17 for more information.

D RankGPT

The prompt template used during the first round of RankGPT generation is as follows, based on which the prompts are constructed to allow LLM to perform listwise document sorting.

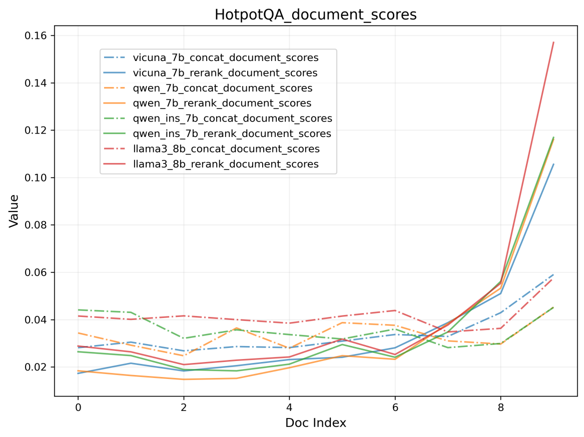


Figure 9: The document scores S of all models on HotpotQA datasets. The solid line - corresponds to the rerank input, the dashed -. line corresponds to the concat input, and the results of the same model are shown in the same color.

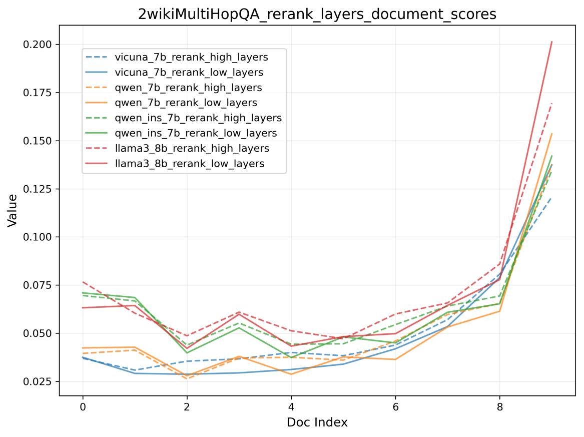


Figure 11: The document scores S of all models with different selected layers on 2wikiMultiHopQA datasets under rerank input. The solid - and dotted – lines are used to distinguish the first and the last half of layers.

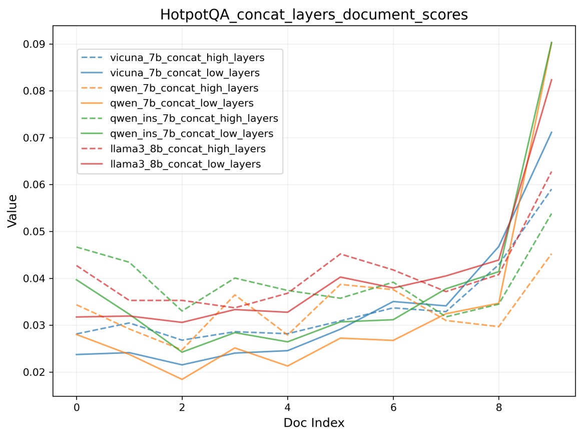


Figure 12: The document scores S of all models with different selected layers on Hotpot datasets under concat input. The solid - and dotted – lines are used to distinguish the first and the last half of layers.

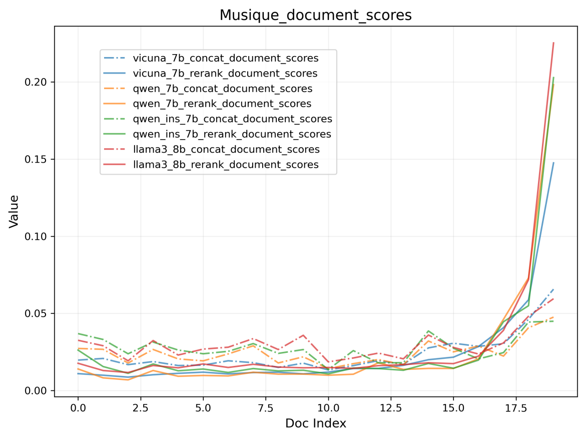


Figure 10: The document scores S of all models on Musique datasets. The solid line - corresponds to the rerank input, the dashed -. line corresponds to the concat input, and the results of the same model are shown in the same color.

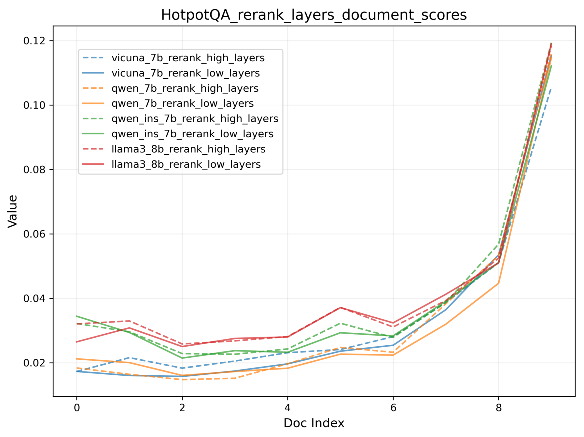


Figure 13: The document scores S of all models with different selected layers on Hotpot datasets under rerank input. The solid - and dotted – lines are used to distinguish the first and the last half of layers.

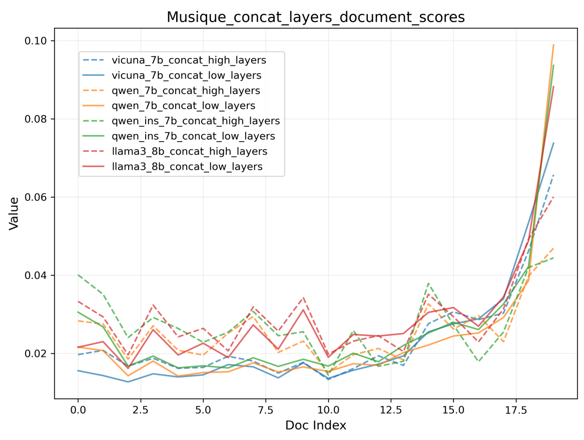


Figure 14: The document scores S of all models with different selected layers on Musique datasets under concat input. The solid - and dotted – lines are used to distinguish the first and the last half of layers.

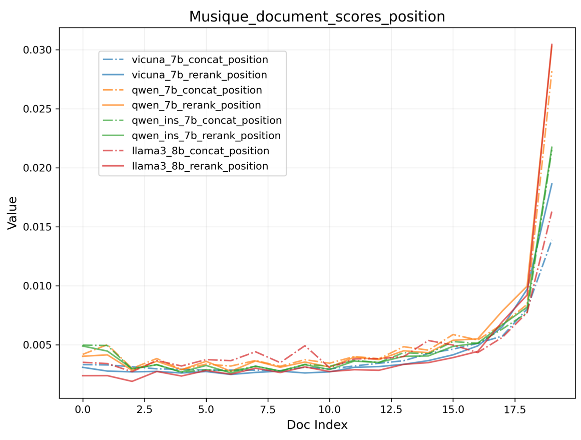


Figure 17: The positional scores S of all models on Musique datasets, which are calculated by aggregating the document scores of the previous token and the terminating token of the answer tokens.

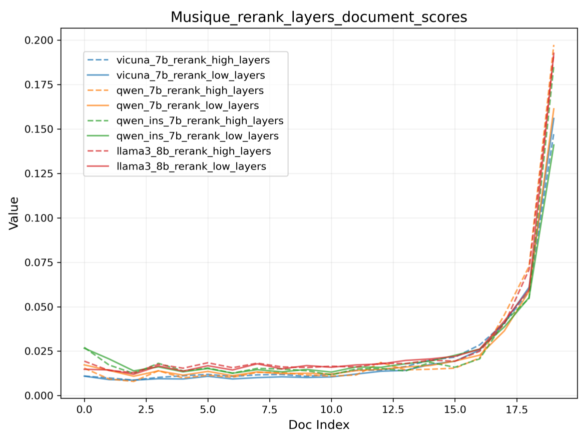


Figure 15: The document scores S of all models with different selected layers on Musique datasets under rerank input. The solid - and dotted – lines are used to distinguish the first and the last half of layers.

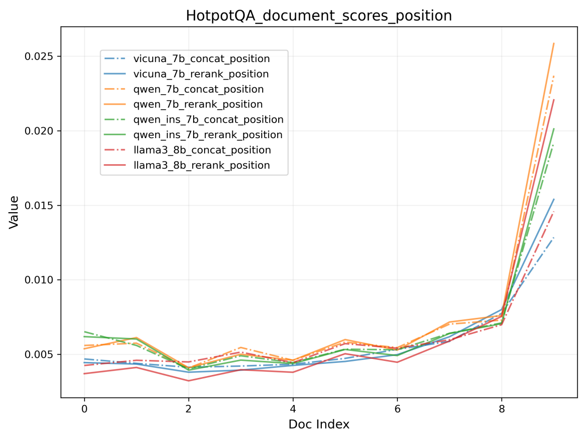


Figure 16: The positional scores S of all models on HotpotQA datasets, which are calculated by aggregating the document scores of the previous token and the terminating token of the answer tokens.

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:

E Examples of Different Ordering

The goal of original rerank and U-shaped Placement is to place good documents in good positions, but the default good positions are different. As an example, if the dataset has 10 documents, the or-

Ranking	Aggregation	H	M	W
Retrieval	-	0.4	0.312	0.482
Attention Weight	answer	0.406	0.305	0.494
	calibration	0.398	0.279	0.496

Table 5: Results of the different document relevance sorting methods of vicuna-7b model under rerank input. Calibration means subtracting positional influence from attention scores.

der of documents under the rerank input is [0,1,.... ,9], the question is placed at the end, document 9 has the best relevance, and document 0 has the worst relevance. After placing the documents according to the positional bias under the U-shaped Placement, the order of documents may become [6,5,4,2,0,1,3,7,8,9], and the question is placed at the end as well. While the reverse version of rerank has the input document order as [9,8,... ,0], and the reverse version of U-shaped Placement has the document order [3,4,5,7,9,8,6,2,1,0], with bad documents prioritized to occupy the default good placements in each reverse order.

F Something We Tried

Due to space limitations, the results of this part of the experiment could not be placed in the main text and are presented below.

F.1 Calibration

As in section 6.2, the vicuna model was also used in the experiments under the rerank input and the results are presented in Table 5.

F.2 Zigzag

The results are presented in Table 6.

Placement	Vicuna-7b			Llama-3.1-8b			Qwen2.5-7b			Qwen2.5-7b-ins		
	H	M	W	H	M	W	H	M	W	H	M	W
rerank	0.4	0.312	0.482	0.302	0.238	0.358	0.408	0.342	0.41	0.472	0.5	0.526
U-shaped (Our)	0.397	0.314	0.524	0.31	0.252	0.376	0.416	0.369	0.414	0.486	0.509	0.522
Direct-U	0.39	0.291	0.49	0.31	0.232	0.356	0.406	0.361	0.406	0.472	0.495	0.516

Table 6: Results of different placements after sorting them for relevance based on external search scores. Rerank means directly placing the relevant ones close to the questions, while U-shaped is our proposed method in accordance with positional bias. Direct-U means aggregating token-level position scores by document and then placing the document directly according to the result of document-level.