Attention Weights as an Indicator: Analyzing and Improving Document Utilization in Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

The generation of Retrieval-Augmented Generation (RAG) models is affected by factors such as the quality and order of input documents, indicating that their ability to utilize documents remains underdeveloped. This ability encompasses not only identifying useful documents from inputs but also minimizing positional bias and filtering irrelevant documents. To achieve this, key challenges include the model's internal estimation of document importance and positional bias. In this paper, we conduct a comprehensive study on the properties of attention weights, examining the impact of factors like aggregation methods, document quality, document position, token type, and so on. Based on our findings, we propose strategies to enhance document utilization from three perspectives: document ranking, placement, and filtering. Comprehensive experiments show that our method outperforms baselines and improves document utilization effectiveness in a training-free manner. Our code is submitted with the paper and will be publicly available.

1 Introduction

As large language models(LLMs) advance, Retrieval Augmented Generation(RAG) has emerged as a helpful paradigm that improves factual accuracy and reliability by integrating external knowledge as context (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 ineffective retrievers (Yan et al., 2024) or misalignment between the retriever and generator (Ke et al., 2024; Li and Ramakrishnan, 2025).

How to improve a model's ability to utilize documents with inputs of varying quality is a significant research challenge, closely associated with model robustness (Shi et al., 2023; Yoran et al., 2024; Zhou et al., 2025). Previous works improve document utilization by integrating irrelevant and interfering documents during supervised fine-tuning (Pan et al., 2024; Yoran et al., 2024; Tu et al., 2025), which requires additional training resources. In contrast, we examine the internal mechanisms of document utilization without requiring training.

In traditional RAG models, documents are placed in sequence according to scores from an external retriever or reranker (Gao et al., 2023). This approach, however, faces several limitations. First, external rankings may not align optimally with LLM generation due to a gap between the retriever and generator (Ke et al., 2024; Li and Ramakrishnan, 2025). Second, the model is prompt-sensitive and exhibits positional bias, preventing it from fairly utilizing information from all positions (Liu et al., 2024; Cuconasu et al., 2024; Wu et al., 2024). Furthermore, irrelevant documents can negatively impact results (Cuconasu et al., 2024; Yan et al., 2024). The ideal scenario involves identifying important documents, filtering out irrelevant ones, and prioritizing useful information to maximize document utilization.

In this paper, inspired by Peysakhovich and Lerer (2023), we investigate the distinct properties of attention weights and propose strategies to enhance document utilization from three perspectives: document ranking, placement, and filtering. The framework is shown in Figure 1. We first analyze the impact of different aggregation methods on ranking results and establish that aggregation based on answer tokens is more effective. Subsequently, we explore the influence of positional bias on attention weights both horizontally (input-level) and vertically(layer-level). Based on this analysis, we propose a method for dynamically placing documents to adapt to positional bias. Additionally, we examine the feasibility of filtering documents using attention weights. The strategies we propose are orthogonal. Each can individually enhance performance, and they can also be combined to comprehensively improve overall document utilization.

We conduct comprehensive experiments on several multi-document QA datasets using various widely adopted LLMs. The results demonstrate that our strategies consistently outperforms baselines and enhance response quality. Our approach requires no additional training and can be directly applied to diverse models and datasets.

2 Preliminaries

2.1 Notations

We formulate the task as generating the answer based on a given question and retrieved documents, following standard RAG settings. For each sample, let q denote the question. The retrieval documents are denoted as $D = \{d_0, d_1, ..., d_{N-1}\}$, where d_i is a document, and N is the total number of documents. $x = \{x_0, x_1, ..., x_{k-1}\}$ is the input, and k is the number of tokens contained in the input, i.e., token length. The input x is constructed from q, D, and a prompt template T. The output answer is $y = \{y_0, y_1, ..., y_{m-1}\}$, where m is the token length of y. The language model, denoted by θ, generates y in an auto-regressive style.

2.2 Preliminary Experiments

We first evaluate model performance under two standard configurations: unordered documents and externally ranked documents. These settings reflect common practices in both RAG evaluation and real-world applications.

Datasets We utilize datasets processed by Pan et al. (2024), which include both randomized (*Unordered*) and retrieval-ranked (*Ordered*) versions to minimize processing-related randomness. Due to computational constraints, our experiments are conducted on three widely-used open-domain multi-document QA benchmarks: HotpotQA (Yang et al., 2018), Musique (Trivedi et al., 2022), and 2WikiMHQA (Ho et al., 2020). The details can be found in the original paper or Appendix A.1.

Models and Metrics We test four 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). Following Pan et al. (2024), we employ Exact Match (EM) as the metric, which checks whether the answers provided are exact substrings of the generation.

Results We conduct experiments under a zero-shot setting, and the details of our implementation are provided in Appendix A.2. The results of the

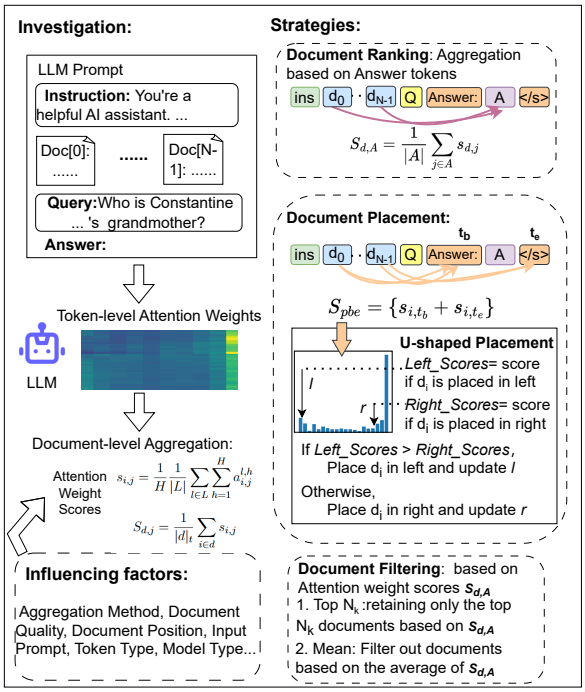

Figure 1: The overall framework.

EM values are presented in Table 1. The discrepancy between our results and those reported by Pan et al. (2024) is likely due to differences in evaluation settings, specifically our use of a zero-shot setup, as well as potential variations in huggingface versions and hardware environments. The results indicate that all models exhibit sensitivity to document order. They fail to utilize document equally across different positions and struggle to capture useful information under varying document arrangements. This suggests that models' ability to utilize documents requires improvement.

2.3 Attention Weight Aggregation

Attention weights capture the influence of document tokens on answer tokens at the token level during generation. Prior work (Peysakhovich and Lerer, 2023) introduced a document-level aggregation of attention weights to derive a relevance score for each document. Formally, the score for a document is computed as follows:

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

$$S_{d,j} = \frac{1}{|d|_t}\sum_{i\in d} s_{i,j} \quad (2)$$

where $a_{i,j}^{l,h}$ denotes the attention weight from token i (belonging to document d whose token length is

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
Unordered	0.392	0.238	0.452	0.292	0.192	0.35	0.376	0.298	0.39	0.468	0.458	0.48
Ordered	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: Original zero-shot model performance in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024).

$|d|_t$) to token j by attention head h at layer l. H is the total number of attention heads, and L is the set of selected layers. $s_{i,j}$ is the attention weights of token i relative to token j. The aggregated score $S_{d,j}$ combines the contributions of all tokens in document d and performs length normalization to produce the document-level score on token j. $S_j = \{S_{d_0,j}, ..., S_{d_{N-1},j}\}$ is the overall set of document scores towards token j. Using this aggregation method, we explore the impact of various factors and further applications of the resulting scores.

3 Investigation on Attention Weights

In this paper, we investigates three key dimensions for improving document utilization: document ranking, placement, and filtering. Our proposed strategies are based on attention weight properties and aim to enhance LLMs' document utilization in a training-free manner. This section presents the individual contributions of each strategy, while Section 4 demonstrates their combined effectiveness. The framework is shown in figure 1.

3.1 Document Ranking

Prior work (Peysakhovich and Lerer, 2023) introduced the use of aggregated attention weight scores to estimate document importance for subsequent ranking. In their implementation, token j corresponds to the first generated token. Later studies proposed aggregating over query tokens (Chen et al., 2024; Liu et al., 2025). However, we argue that aggregating solely over query tokens or the first generated token does not accurately reflect a document's influence on the final answer. A more direct approach is to aggregate over all answer tokens. Therefore, we compare three aggregation methods: aggregation over query tokens, aggregation over the first generated token, and aggregation over answer tokens. For query and answer tokens, we sum the individual token-level contributions and compute the average to ensure a fair comparison.

The CAGB benchmark (Pan et al., 2024) provides data in various arrangements along with manual annotations of document quality, categorized as *high*, *medium*, or *low*. Therefore, we sample 100 examples to visualize the relationship between the scores obtained through different aggregation methods and corresponding document quality. Due to space constraints, only results for the Qwen2.5-7B model are included in the main text, and results for the other models are provided in Appendix H.1.

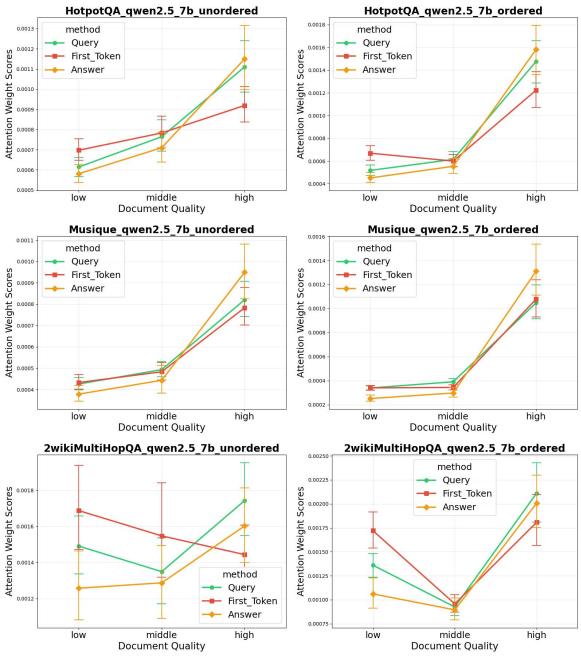

Figure 2: The relationship between attention weight scores obtained through different methods and the corresponding document quality of Qwen2.5-7B model.

Figure 2 supports several conclusions. First, attention weight scores partially reflect document quality, indicating that ranking documents based on these scores can capture the model's internal assessment of document importance. Second, the order of input documents influences the results, with document quality being more difficult to distinguish when documents are unordered. The effect of document arrangement will be analyzed in the next section. Most importantly, among the aggregation methods compared, aggregating over answer tokens (orange) more effectively distinguishes document quality than aggregating over query tokens or the first generated token.

In addition to visual analysis, we further compare different aggregation methods based on downstream task performance. Experimental results for Vicuna-7B in Table 2 confirm that the answer-based aggregation method outperforms prior approaches. Results for other models are provided in Appendix H.2. Intuitively, the reliability of this attention-weight-based ranking method is related to the model's inherent capabilities. We validated this by having Vicuna-7B generate answers using rankings produced by Qwen2.5-7B. The observed performance improvement suggests the promise of hybrid approaches that leverage multiple models during generation. We also conducted experiments on token types within documents, as detailed in Appendix B.

Prompt	Method	H	M	W
Unordered	First Token	0.386	0.291	0.484
	Query	0.386	0.265	0.468
	Answer	0.406	0.293	0.474
	Answer(qwen)	0.408	0.295	0.488
Ordered	First Token	0.404	0.3	0.474
	Query	0.398	0.301	0.49
	Answer	0.406	0.305	0.494
	Answer(qwen)	0.412	0.329	0.512

Table 2: Results of the different aggregation methods of vicuna-7b model.

Aggregating answer tokens can yield better performance, but the gains from altering aggregation methods are limited. Unless otherwise specified, the attention weight scores mentioned in the following sections are based on aggregation over answer tokens.

3.2 Document Placement

This section investigates the impact of different document arrangements under the assumption that a predefined order of document relevance is provided, such as one derived from retrieval ranking, attention-weighted ranking, or any other relevance ranking methods.

As shown in Figure 2, for the same dataset, ordered inputs (right) allocate higher attention weights to high-quality documents placed at the end compared to unordered inputs (left). And under ordered conditions, the distinction in attention weights between middle- and low-quality documents becomes less clear. Low-quality documents even receive higher weights than middle-quality ones in some datasets. This pattern, also observed in other models (see Appendix H.1), highlights the effect of positional bias. As noted by Liu et al. (2024), language models tend to exhibit U-shaped positional bias, assigning greater attention to the beginning and end of the input while often overlooking content in the middle.

To examine this phenomenon, we analyze it from both horizontal and vertical perspectives. Horizontal analysis investigates the relationship between attention weight scores and document positions under ordered and unordered input. Vertical analysis involves controlling the selected attention layers by dividing all layers into upper and lower halves. Given that positional bias was most evident on the 2wikiMHQA dataset in prior experiments, we visualize results from various models on this dataset in Figure 3. Results for other datasets are provided in Appendix H.3. Figure 3(a) presents the horizontal comparison, revealing a U-shaped distribution across document positions under both ordered and unordered input conditions. Figure 3(b) shows the vertical comparison, indicating that positional distinctions are more pronounced in the lower layers. This observation aligns with the widely accepted view that lower layers are more sensitive to positional information, while higher layers focus on processing semantic content. In summary, attention weight scores are influenced not only by document relevance but also by positional bias. Isolating positional effects from attention weights and strategically leveraging favorable positions constitute the focus of our subsequent research.

To eliminate the impact of semantic relevance, we consider aggregating attention weights from tokens that carry minimal semantic meaning. Since the goal is to obtain the positional information of answer tokens, we examine the tokens preceding and following the answer tokens. In the construction of prompt templates, the final token preceding the answer often serves a structural rather than semantic role, such as *"Answer:"*, *":"* or *"is"*. Similarly, the generated output typically ends with a special token that does not convey semantic content. Therefore, we combine the attention weights of these two tokens into a positional score, focusing on lower layers where positional signals are more distinct. The results are visualized in Figure 3(c). Comparing Figure3 (a) and (c) reveals that the scores derived from tokens surrounding the answer vary little across different inputs. They consistently display the same U-shaped trend under both ordered and unordered conditions. To as-

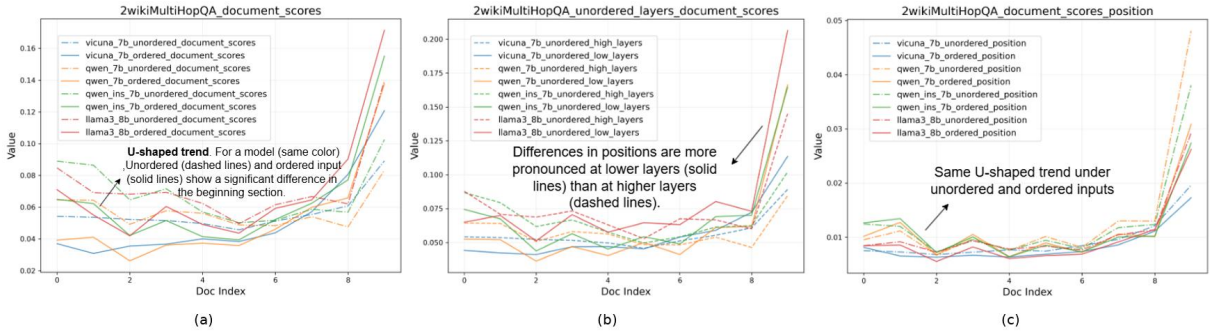

Figure 3: Relationship between attention weight scores and document positions on the 2wikiMultiHopQA dataset. Figure (a) shows comparison between ordered and unordered inputs(horizontal). Figure (b) compares different layers under unordered inputs(vertical).Figure (c) compares extracted position scores under ordered and unordered.

sess robustness, we repeated the experiments using different random seeds for unordered input. The results, provided in the Appendix C, demonstrate that the calculated outcomes remain stable even with multiple randomizations of document order.

Algorithm 1 U-shaped Placement

Input Relevance ranking R, attention weight A_θ, preceding token t_b and terminating token t_e of answer, the collection of token lengths for all documents $T_l = \{T_0, ..., T_{N-1} | T_i = |d_i|_t\}$.

Ensure that the relevant ones in R come first;
Get S_{pbe} = $\{s_{i,\{t_b,t_e\}}\}$ based on A_θ; ▷ (1)
Initialization: $l = 0, r = \sum_{i=0}^{N-1} T_i, l_{idx} = 0, r_{idx} = N-1, R_u = [0] * N$;
for i $\in R$ **do**
  $T_i = T_l[i]$;
  Left_Scores = $S_{pbe}[l : l + T_i]$.sum();
  Right_Scores = $S_{pbe}[r - T_i : r]$.sum();
  if Right_Scores $\geq$ Left_Scores **then**
    $R_u[r_{idx}] = i, r_{idx} = r_{idx} - 1, r = r - T_i$;
  else
    $R_u[l_{idx}] = i, l_{idx} = l_{idx} + 1, l = l + T_i$;
  end if
end for

Output: The new ranking R_u

After obtaining the positional scores which are relevance-independent, we aim to enhance the model's ability to utilize documents by placing higher-quality documents in more favorable positions. A direct strategy would be to rank documents in descending order of aggregated positional scores. However, this approach faces a practical limitation due to length variability. Because attention weight scores are length-normalized (Equation 2) to remove document length effects, the actual token capacity corresponding to each score can vary significantly. For instance, the highest-scoring position may accommodate only 100 tokens, while longer documents would occupy positions associated with lower scores. To address this issue, we operate directly on token-level scores (Equation 1) rather than document-level aggregates. Additional experiments on this issue are presented in the appendix L. The underlying principle remains consistent: to place high-quality documents in positions that receive higher attention. Specifically, we propose a placement strategy that processes documents in descending order of relevance and uses the observed U-shaped attention pattern to place each document at either the beginning or end of the available context. At each step, the algorithm evaluates whether placing the document on the left or right side of the remaining prompt space yields a higher token-level score, and assigns it accordingly. This continues until all documents are placed, resulting in a dynamic U-shaped arrangement that aligns with the model's inherent attention bias. The complete procedure, termed **U-shaped Placement**, is formalized in pseudocode in Algorithm 1.

The U-shaped Placement approach is compatible with all kinds of document ranking methods. We first integrate it with retrieval ranking to illustrate the effect of placement alone. The results are presented in Table 3. The results of combining it with attention-weight ranking will be presented in section 4. "Default" refers to placing relevant documents close to the question, while different settings indicate whether favorable positions are assigned to high-quality (original) or low-quality (reverse) documents (see Appendix D for details). The results show that: (1) The proposed U-shaped Placement, which aligns with the model's posi-

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
Default	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.

tional bias, constitutes an effective arrangement that enhances the model's ability to utilize documents. (2) Among reverse-placement configurations, the reversed U-shaped Placement leads to the most pronounced performance degradation, highlighting the importance of leveraging positional bias appropriately.

3.3 Document Filtering

In addition to placing good documents in advantageous positions, removing irrelevant documents is also a way to enhance document utilization. This section explores the removal of documents based on attention weight scores while preserving their original relative order.

We consider two heuristic filtering methods. The first retains a fixed number of documents: given a predefined number N_k, we compute attention weight scores for each document and keep only the top N_k highest-scoring documents. The second uses a dynamic threshold. As illustrated in Figure 2, the attention weight scores for high-quality documents are higher than those for other quality levels, and there is no significant difference in the number of documents across different quality levels. We therefore filter out documents whose attention weight score falls below the average score across all documents, resulting in a variable number of retained documents. The actual document count for this method is detailed in the Appendix E.

Prompt	Method	H	M	W
Unordered	No Filter	0.392	0.238	0.452
	Random N_k	0.318	0.195	0.368
	Top N_k	0.396	0.275	0.428
	Mean	0.394	0.268	0.4
Ordered	No Filter	0.4	0.312	0.482
	Random N_k	0.308	0.208	0.366
	Top N_k	0.388	0.316	0.472
	Mean	0.412	0.316	0.424

Table 4: Results of the different filtering methods of vicuna-7b model.

We experimentally investigate the effectiveness of the filtering methods, with the results shown in Table 4. The results for other models are presented in Appendix H.4. For the fixed-document method, N_k is set to half the original number of documents ($N_k = \frac{1}{2}N$). The results show that both heuristic filtering methods based on attention weight scores significantly outperform random filtering, confirming that attention weight scores reliably distinguish between documents of varying quality. Between the two methods, Top N_k yields more stable performance, whereas the Mean method retains fewer documents (see Appendix E). Across datasets, on HotpotQA and Musique datasets, our filtering approach achieves performance comparable or superior to using the full document set, suggesting that irrelevant documents notably affect these two datasets. In contrast, 2wikiMHQA benefits from retaining more documents, indicating a greater reliance on comprehensive information extraction.

4 Main Experiments

In this section, we explore the combined effect of our proposed strategies. The experimental setup follows the preliminary experiments in Section 2.2.

Since our approach essentially involves a two-round iteration of the LLM, we compare it with similarly structured baselines for a fair evaluation. The baselines include: Vanilla, RankGPT (Sun et al., 2023), Attention Sorting (Peysakhovich and Lerer, 2023), ICR (Chen et al., 2024), and SELF-ELICIT (Liu et al., 2025). Descriptions of each baseline are provided in Appendix F.

4.1 Results Without Filtering

We first keep the number of documents fixed and do not apply filtering. To demonstrate the overall effect, we combine our proposed document ranking (denoted as DR) with the U-shaped Placement strategy (denoted as U). The complete pipeline is formalized as pseudocode in Appendix I.

The results are shown in Table 5. The results show that: (1) Our method outperforms previous

Prompt	Methods	Vicuna-7b			Llama-3.1-8b			Qwen2.5-7b			Qwen2.5-7b-ins			All
		H	M	W	H	M	W	H	M	W	H	M	W	A
Unordered	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	0.365
	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	0.376
	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	0.385
	ICR	**0.421**	0.305	0.478	0.298	0.237	0.368	0.379	0.385	0.384	0.469	0.511	0.522	0.396
	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	0.352
	Our DR	0.406	0.293	0.474	0.298	0.236	0.36	0.398	0.378	0.424	0.476	0.515	0.52	0.398
	Our DR + DP	0.414	**0.31**	**0.506**	0.302	**0.245**	**0.372**	**0.402**	**0.393**	**0.434**	**0.482**	**0.513**	**0.536**	**0.409**
Ordered	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	0.396
	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	0.394
	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	0.396
	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	0.404
	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	0.379
	Our DR	0.406	0.305	0.494	0.302	0.257	0.366	0.405	0.387	0.428	0.49	0.525	0.53	0.408
	Our DR + DP	**0.426**	**0.315**	0.51	0.304	**0.267**	**0.378**	**0.412**	**0.401**	**0.436**	**0.495**	**0.555**	**0.542**	**0.420**

Table 5: Zero-shot model performance in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024). **A** is the average of all models and datasets' performance.

baselines on most datasets and models, under both unordered and ordered input settings. This demonstrates the effectiveness of the ranking method, and combining it with placement strategy that align with positional bias can further improves document utilization. (2) Improvements are more substantial under unordered inputs. This can be partly attributed to the greater potential for enhancement in unordered settings. Notably, our approach applied to unordered inputs can surpass the performance of the vanilla ordered baseline that relies on external retrieval rankings, demonstrating its ability to infer an effective document order even without prior ranking. The gains under ordered inputs further confirm that our method enhances the model's capacity to utilize documents effectively. (3) Improvements are more pronounced on datasets with more documents, such as Musique. Our method can be applied to settings with varying document counts, as verified in Appendix J. In terms of models, greater gains are observed on the Qwen series compared to Vicuna-7b, which may be related to the base capability of the model: stronger models provide more reliable internal state signals. Results for the larger model (13B) are presented in the Appendix K. We also experimented with a calibration method involving position score subtraction, detailed in Appendix M.

Nevertheless, our method delivers consistent performance improvements across diverse models and datasets.

4.2 Results With Filtering

In this section, we combine document filtering with ranking to achieve further performance improvements. Again, we employ two rounds of iteration: document filtering and ranking based on the attention weight scores from the first round. Since the position scores obtained in the first round correspond to the original N documents and cannot be directly transferred to the filtered subset, we employ the default placement method here rather than U-shaped placement. The setup primarily evaluates the combined effect of document filtering and ranking.

Following the setup in Section 3.3, for methods using a fixed number of documents, N_k is set to half the original number of documents ($N_k = \frac{1}{2}N$). Under both ordered and unordered inputs, we consider four scenarios: (1) Vanilla-N: Results before filtering, where the full set of N documents is used. (2) Retrieval-N_k: The top N_k documents are selected according to the retrieval ranking. (3) DR + Top N_k: Documents are filtered by attention weight scores, retaining the top N_k documents, which are then sorted by score. (4) DR + Mean: Documents with attention weight scores below the average are filtered out and the remaining documents are sorted by score. For scenarios (3) and (4), filtering and ranking is applied to the original N documents.

The results in Table 6 show that: (1) Compared to the results in Table 4, incorporating ranking can further improve performance without additional inference cost. (2) Overall, on the HotpotQA and Musique datasets, filtering by external retrieval degrades performance, whereas our method outperforms the unfiltered baseline while using fewer documents. This demonstrates that filtering irrelevant documents based on attention weight scores is more effective than using an external retriever, high-

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
Unordered	Vanilla-N	0.392	0.238	0.452	0.292	0.192	0.35	0.376	0.298	0.39	0.468	0.458	0.48
	Retrieval-N_k	0.394	0.232	0.428	0.29	0.174	0.288	0.36	0.298	0.388	0.452	0.371	0.428
	DR + TopN_k	**0.398**	**0.292**	0.431	**0.294**	**0.242**	0.292	**0.382**	**0.402**	**0.396**	0.474	**0.494**	0.436
	DR + Mean	0.396	0.261	0.422	0.284	0.241	0.296	**0.382**	0.391	0.384	**0.476**	0.487	0.416
Ordered	Vanilla-N	0.4	0.312	0.482	0.302	0.238	0.358	0.408	0.342	0.41	0.472	0.5	0.526
	Retrieval-N_k	0.384	0.256	0.462	0.304	0.194	0.29	0.4	0.296	0.396	0.458	0.413	0.446
	DR + TopN_k	0.404	**0.318**	0.478	0.288	0.236	0.288	0.38	**0.396**	0.394	0.47	**0.511**	0.46
	DR + Mean	**0.426**	0.312	0.426	**0.304**	**0.246**	0.318	**0.418**	0.376	0.406	0.462	0.492	0.442

Table 6: Results of filtering methods in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024).

lighting a gap between retrieval ranking and LLM generation needs. (3) On 2wikiMHQA dataset, performance degrades as the number of documents decreases, which is a characteristic inherent to the dataset. Our approach alleviates this decline and achieves better results than retrieval-based filtering under the same conditions.

5 Related Works

Retrieval-Augmented Generation RAG has proven effective in mitigating 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 quality (Shi et al., 2023; Yoran et al., 2024; Wu et al., 2024). Previous works have explored various improvement strategies, such as improving the retriever (Shi et al., 2024), designing new rerankers (Kim and Lee, 2024), bridging the gap between the retriever and generator (Ke et al., 2024; Li and Ramakrishnan, 2025), and improving model robustness to interference (Xiang et al., 2024; Yoran et al., 2024). In contrast to methods that directly incorporate documents into training or fine-tuning (Pan et al., 2024; Yoran et al., 2024; Tu et al., 2025), we study how language models internally utilize documents and dynamically modify inputs based on attention weights to improve performance.

Document Utilization The relevance and arrangement of prompt significantly influence the output generated by LLM(Cuconasu et al., 2024). In the RAG workflow, documents are conventionally ranked using scores from external retrievers or rerankers (Gao et al., 2023; Shi et al., 2024; Kim and Lee, 2024). There are also some works that enable the model itself to assess document relevance 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). Filtering operations are generally also based on document relevance scores(Cuconasu et al., 2024). The default placement method is sequential placement based on relevance. However, the LLMs are unable to treat the information in the prompt equally and have a positional bias. This "Lost in the Middle" phenomenon was first identified in Liu et al. (2024). Subsequent studies in RAG and long-context modeling have examined performance variations due to this bias (Cuconasu et al., 2024; Wu et al., 2024; Xu et al., 2024) In this paper, we leverage attention weights not only for ranking purposes but also to investigate the impact of different aggregation methods, the utilization of positional bias, and the role of filtering techniques. These approaches collectively enhance the model's ability to effectively utilize documents.

6 Conclusion

In this paper, We conduct our research based on attention weights and their document-level aggregation. We first investigate the impact of different aggregation methods, demonstrating that aggregating over answer tokens yields superior ranking performance. Then, leveraging the nature of positional bias, we propose a placement method called U-shaped Placement that aligns with positional bias. Additionally, we introduce two heuristic methods to filter irrelevant documents. Each of the three distinct strategies can independently improve performance in a training-free manner, and when combined, they comprehensively enhance the model's ability to utilize documents. Our approach requires no training and can be applied in a plug-and-play fashion to 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, for example, finer aggregation granularity, from document level to sentence level. 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

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.

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.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop 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.

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 of Datasets and Implementation

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. The statistics on the number of documents included in the dataset are shown in Table 7.

	HotpotQA	Musique	2WikiMHQA
N	10	20	10

Table 7: N is the number of documents for each sample.

See original paper (Pan et al., 2024) for more details.

A.2 Implementation Details

Hyperparameters including temperature and instruction format remain consistent with the setting of Pan et al. (2024). Unlike their work, however, we conduct experiments under a zero-shot setting to better isolate and examine the model's intrinsic positional bias and relevance assessment mechanisms. The placement of documents in the prompt adheres to Pan et al. (2024), positioning the most relevant documents closest to the question when documents are ordered. Previous researches (Cuconasu et al., 2024; Liu et al., 2025) have also confirmed that this placement is a widely applied paradigm and strong baseline.

Furthermore, the seed is set to 42. The 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 Token Type

In this paper, we aggregate all tokens within the document to obtain document-level attention weight scores. However, not all tokens in a document carry semantic information, and there will always be stopwords such as punctuation marks that lack semantic meaning. In this section, we discuss the impact of removing stopwords such as punctuation marks during the aggregation of attention weight scores. The results are shown in Table 8. The results indicate that removing stopwords such as punctuation marks yielded outcomes similar to the original, with no significant improvement in effectiveness.

C Random Variation of Positional Scores

To eliminate the impact of random variation while enhancing the credibility of our conclusions, we conduct repeated experiments using different random number seeds for experiments in Figure 3. We plotted the range of variation in the results from five repeated experiments as shaded areas in the figure 4 .

The results demonstrate that even with multiple randomizations of document order, the calculated outcomes exhibit a certain degree of stability.

Prompt	Methods	Vicuna-7b			Qwen2.5-7b		
		H	M	W	H	M	W
Unordered	Answer	0.406	0.293	0.474	0.398	0.378	0.424
	Answer(Remove Stopwords)	0.4	0.285	0.482	0.406	0.368	0.426
Ordered	Answer	0.406	0.305	0.494	0.405	0.387	0.428
	Answer(Remove Stopwords)	0.394	0.303	0.5	0.396	0.391	0.42

Table 8: Results of removing stopwords within documents.

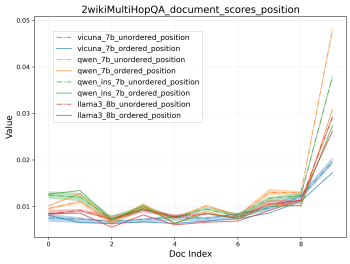

Figure 4: The results from five repeated experiments.The shaded area indicates the range of score variations across these five random experiments.

D Placement Strategies

We evaluate four placement strategies: placing relevant documents near the question, U-shaped Placement, and the reverse variants of both. In the original versions, higher-relevance documents are assigned to positions that inherently receive more attention, while the reverse versions deliberately assign lower-relevance documents to these favored positions. All four configurations use identical prompt templates, varying only in document order, thereby isolating the effect of placement on model performance.

As an example, if the dataset has 10 documents, the order of documents under the ordered 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 ordered 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.

E Document Count Statistics

We propose two filtering methods: one with a fixed number of documents and another that dynamically determines the filtering threshold (Mean). To facilitate comparison, we statistically analyzed the number of documents retained by the Mean method and present the results in Table 9.

F Baselines

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 listwise style and the second round generates the answer. The prompt template used in the first round is shown in the Appendix G. **(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 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.

G 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.

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
No Filter	10	20	10	10	20	10	10	20	10	10	20	10
Top N_k	5	10	5	5	10	5	5	10	5	5	10	5
Mean (Unordered)	3.45	6.58	3.84	3.44	6.09	3.74	3.23	5.72	3.62	3.18	5.55	3.76
Mean (Ordered)	2.93	5.15	3.34	2.88	4.72	3.42	2.61	4.32	3.19	2.73	4.40	3.42

Table 9: The number of retaining documents of filtering methods in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024).

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:

H Full Experimental Results

Due to space constraints in the main text, we have included the detailed experimental results for each model across various datasets in this section.

H.1 Relation Between Scores and Quality

The visualizations of the relationship between the scores obtained through different methods and document quality for Vicuna-7b, Llama-3.1-8B and Qwen2.5-7B-Instruct are shown in Figure 5, 6, 7 .

H.2 Attention-Weighted Ranking

In Section 3.1, we explored the impact of different aggregation methods on results and discussed the influence of token types. The complete experimental results are shown in Table 10.

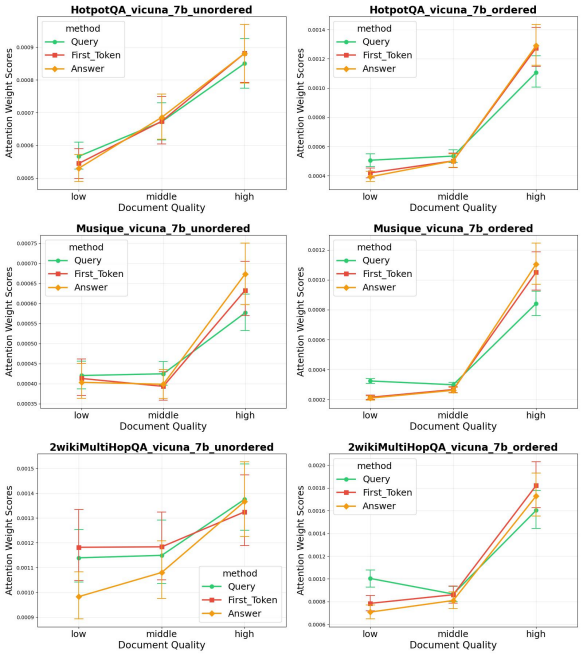

Figure 5: The relationship between attention weight scores obtained through different methods and the corresponding document quality of Vicuna-7B model.

H.3 Relation Between Scores and Quality

The visualizations of relationship between attention weight scores and document positions on the HotpotQA and Musique dataset are shown in Figure 8 and 9.

H.4 Document Filtering

In Section 3.3, we explored document filtering based on attention weights. The complete experimental results are as shown in Table 11.

I Pipeline

The workflow combining attention-weighted document ranking with U-shaped placement is as follows:

Prompt	Methods	Vicuna-7b			Llama-3.1-8b			Qwen2.5-7b			Qwen2.5-7b-ins			All
		H	M	W	H	M	W	H	M	W	H	M	W	A
Unordered	First Token	0.386	0.291	0.484	0.296	0.216	0.336	0.398	0.375	0.4	0.462	0.495	0.484	0.385
	Query	0.386	0.265	0.468	0.304	0.214	0.358	0.404	0.363	0.41	0.474	0.479	0.502	0.386
	Answer	0.406	0.293	0.474	0.298	0.236	0.36	0.398	0.378	0.424	0.476	0.515	0.52	**0.398**
Ordered	First Token	0.404	0.3	0.474	0.294	0.218	0.342	0.408	0.385	0.432	0.492	0.485	0.518	0.396
	Query	0.398	0.301	0.49	0.29	0.232	0.352	0.396	0.363	0.412	0.464	0.503	0.496	0.391
	Answer	0.406	0.305	0.494	0.302	0.257	0.366	0.405	0.387	0.428	0.49	0.525	0.53	**0.408**

Table 10: Results of different aggregation methods in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024). **A** is the average of all models and datasets' performance.

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
Unordered	No Filter	0.392	0.238	0.452	0.292	0.192	0.35	0.376	0.298	0.39	0.468	0.458	0.48
	Random N_k	0.318	0.195	0.368	0.213	0.145	0.298	0.301	0.223	0.312	0.412	0.391	0.409
	Top N_k	0.396	0.275	0.428	0.288	0.222	0.276	0.38	0.358	0.39	0.471	0.481	0.428
	Mean	0.394	0.268	0.4	0.278	0.231	0.281	0.381	0.341	0.381	0.468	0.48	0.402
Ordered	Vanilla-N	0.4	0.312	0.482	0.302	0.238	0.358	0.408	0.342	0.41	0.472	0.5	0.526
	Random N_k	0.308	0.208	0.366	0.221	0.154	0.291	0.298	0.245	0.321	0.422	0.402	0.411
	TopN_k	0.388	0.316	0.472	0.272	0.212	0.281	0.39	0.403	0.384	0.468	0.502	0.448
	Mean	0.412	0.316	0.424	0.293	0.241	0.302	0.4	0.371	0.392	0.468	0.481	0.437

Table 11: Results of filtering methods in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark (Pan et al., 2024).

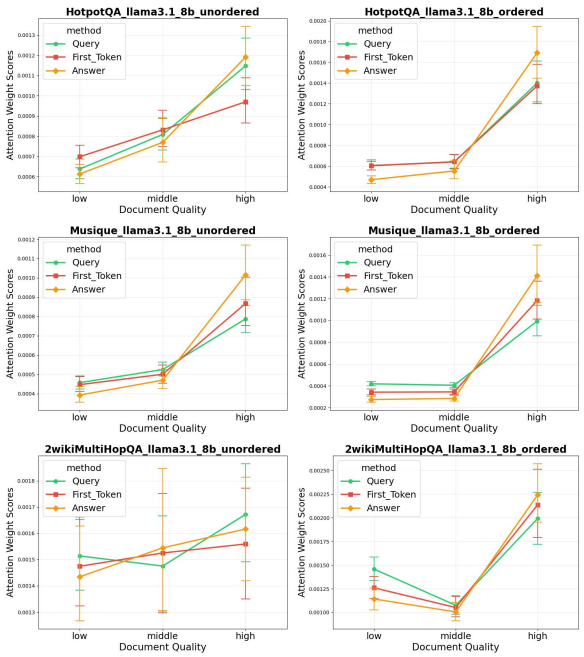

Figure 6: The relationship between attention weight scores obtained through different methods and the corresponding document quality of Llama-3.1-8B model.

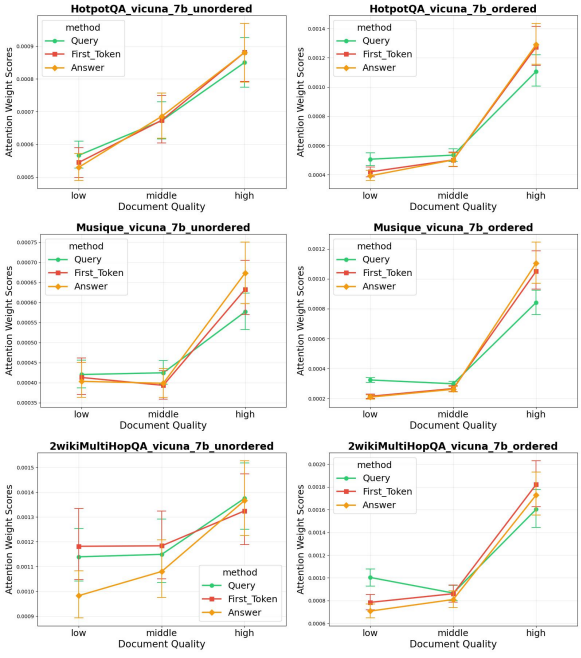

Figure 7: The relationship between attention weight scores obtained through different methods and the corresponding document quality of Qwen2.5-7B-Instruct model.

Figure 8: Relationship between attention weight scores and document positions on the HotpotQA dataset. Figure (a) shows comparison between ordered and unordered inputs(horizontal). Figure (b) compares different layers under unordered inputs(vertical).Figure (c) compares extracted position scores under ordered and unordered.

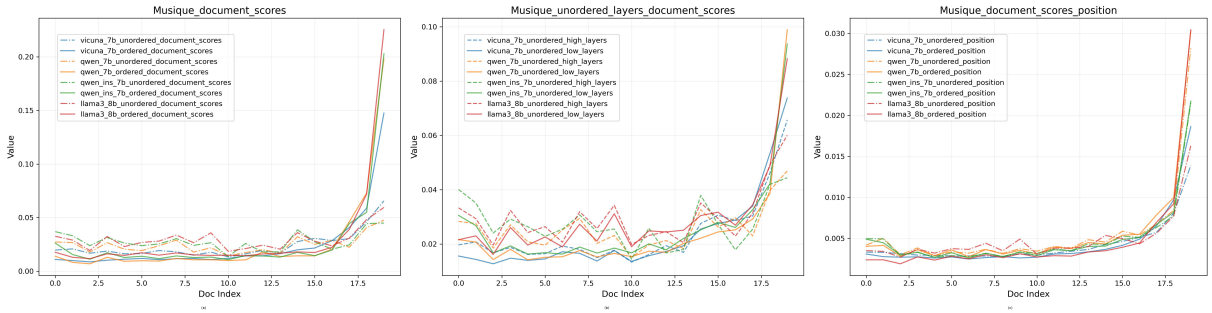

Figure 9: Relationship between attention weight scores and document positions on the Musique dataset. Figure (a) shows comparison between ordered and unordered inputs(horizontal). Figure (b) compares different layers under unordered inputs(vertical).Figure (c) compares extracted position scores under ordered and unordered.

Algorithm 2 Attention Weight Based Ranking and U-shaped Placement

Input Prompt Template T, LLM θ, 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

J Number of Documents

Our method can be applied to experiments involving any number of documents. While the effectiveness of our method has been demonstrated on datasets of 10 and 20 documents in the main experiments, we have also conducted experiments with a small number of input documents to prove the method's versatility and efficacy. We take the original ten-document 2wikiMultiHopQA dataset and only intercepted the first three and five documents for the experiment. Experimental results in the Table 12 demonstrate that our proposed ranking method and U-shaped Placement remain high efficiency across varying numbers of documents.

K Results on 13B Model

To demonstrate that our approach is effective across different model scales, we conduct experiments on Vicuna-13B. The results are shown in Table 13. Due to computational resource constraints, we were unable to conduct experiments on larger model scales, such as 70B.

The results are similar to those in Table 5, demonstrating that our method outperforms the previous baseline on the 13B model and enhances the model's ability to utilize documentation.

L Length Issue

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

Prompt	Methods	Vicuna-7b			Qwen2.5-7b-ins		
		3doc	5doc	10doc	3doc	5doc	10doc
Unordered	Vanilla + Default	0.286	0.296	0.452	0.316	0.334	0.48
	Vanilla + U-shaped	0.334	0.35	0.48	0.34	0.358	0.501
	Our Ranking + Default	0.344	0.358	0.474	0.342	0.35	0.51
	Our Ranking + U-shaped	**0.356**	**0.374**	**0.506**	**0.35**	**0.36**	**0.536**
Ordered	Vanilla + Default	0.38	0.424	0.482	0.366	0.428	0.526
	Vanilla + U-shaped	0.398	**0.47**	**0.524**	0.37	0.464	0.522
	Our Ranking + Default	0.39	0.456	0.494	0.376	0.444	0.53
	Our Ranking + U-shaped	**0.401**	0.463	0.51	**0.394**	**0.466**	**0.542**

Table 12: Results of varying number of input documents.

Prompt	Method	H	M	W
Unordered	Vanilla	0.386	0.264	0.442
	RankGPT	0.388	0.303	0.438
	AttentionSort	0.41	0.305	0.458
	ICR	0.406	**0.313**	0.479
	SELFELICIT	0.382	0.294	0.447
	Our DA + DU	**0.42**	0.311	**0.484**
Ordered	Vanilla	0.406	0.313	0.482
	RankGPT	0.408	0.301	0.456
	AttentionSort	0.408	0.301	0.474
	ICR	0.405	0.343	0.475
	SELFELICIT	0.402	0.334	0.478
	Our DA + DU	**0.414**	**0.357**	**0.497**

Table 13: Zero-shot model performance in HotpotQA(H), Musique(M) and 2WikiMHQA(W) datasets of CAGB benchmark on Vicuna-13B model.

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 3.2. Is the length issue more important or the zigzag issue? We compare these two placement and list the result in Table 14. The results 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.

M Calibration

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. Another explanation is that relevance estimation and positional bias jointly form the attention weight, but they do not follow a simple additive relationship; instead, they exhibit a more complex combinatorial relationship. The results are presented in Table 15.

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
Default	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 14: Results of different placements after sorting them for relevance based on external search scores. Default 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.

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	0.486

Table 15: Results of the different document relevance sorting methods of vicuna-7b model under ordered input. Calibration means subtracting positional influence from attention scores.