CAUS-RAG: Algorithmic Interceptor for Mitigating Positional Bias in Technical Code Generation

Mithun Martin
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Chennai, India
mithun.martin.work@gmail.com

Aritra Roy
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Chennai, India
aritraroy775@gmail.com

T. Grace Shalini
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Chennai, India
gracesht@srmist.edu.in

***Abstract*—Retrieval-Augmented Generation (RAG) revolutionised the fields of Large Language Models (LLMs) and Natural Language Processing by helping to fetch the latest external knowledge while generating responses. However, standard RAG faces a critical challenge when it comes to technical code generation, as it depends solely on semantic similarity, which often leads to failure in retrieving the correct lexical API tokens. The "Lost-in-the-Middle" phenomenon in RAG further causes the model to neglect information placed in the middle of a long query prompt. The proposed framework, CAUS-RAG (Confidence-Aware Adaptive U-Shape), solves this problem by being an algorithmic interceptor that helps in achieving accurate retrieval and effective context placement. It works based on two principles. Firstly, the framework makes use of a Hybrid Confidence Scoring (HCS) mechanism that aggregates both semantic scores and lexical cues to achieve more code-oriented document retrieval. Secondly, the architecture employs a model-aware U-shaped position permutation tuned through a Position Sensitivity Index (PSI) score in order to place the retrieved chunks based on the model's positional attention bias. Experiments conducted based on a PyTorch API benchmark across different models such as Phi-3.5 (3.8B), Mistral (7B), and Qwen-2.5 (7B) show a significant performance improvement measured in terms of Exact Match (EM). CAUS-RAG boosts the Exact Match accuracy of Qwen-2.5 from 17.3% to 86.5% and Phi-3.5 from 11.5% to 71.2%.**

***Index Terms*—Retrieval-Augmented Generation, Positional Bias, Code Generation, Large Language Models, Lost in the Middle, Hybrid Retrieval, BM25**

I. Introduction

Although Large Language Models (LLMs) can automatically write functional code, they rely on static parametric knowledge that requires retraining to incorporate updates, which is expensive and time-consuming [3]. In constantly updating documentations like PyTorch, this can lead to wrong code generation by an LLM due to the calling of a non-existing function.

RAG tackles this problem by introducing the use of external, non-parametric information during the inference stage [2]. Instead of relying solely on the information stored in the model's internal memory, RAG can retrieve the relevant documentation from an up-to-date corpus. Therefore, it becomes feasible, in principle, to produce correct code based on the current version of the API. However, as the retrieved documents increase, the model doesn't allocate equal attention to each piece of the document. "Lost-in-the-Middle" refers to the precise scenario where the attention mechanism of transformers prioritizes attending to the first and last tokens of the context window while ignoring the others. In a scenario where PyTorch functions are generated using documentation, the retrieval module will likely find the documentation containing the desired function. However, the retrieved document might be the third or fourth document out of ten, thus placing itself in the middle of the context. Therefore, the generated code will resort to retrieving function calls from the parametric memory, which might be outdated or entirely fabricated, as conceptually illustrated in Fig. 1.

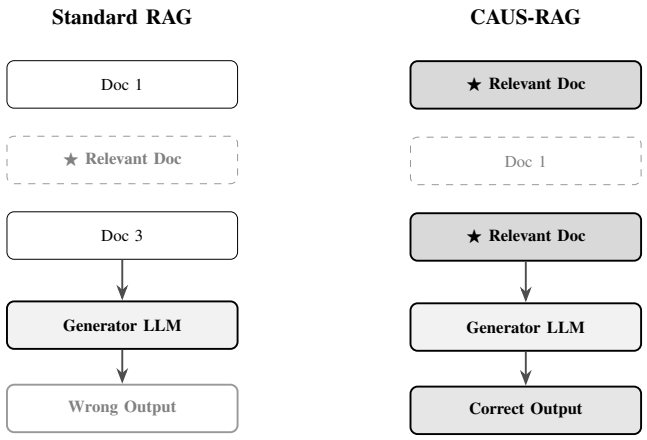

Fig. 1. Conceptual comparison of context allocation in Standard RAG and the proposed CAUS-RAG.

The current methods to solve this limitation in RAGs fail, as they consider the quality of data retrieved and the position of context placed to be two distinct tracks independent of one another. Techniques like position-agnostic training and attention calibration [8], [9] try to make the model pay attention to the middle part of a long query, but they require computationally expensive fine-tuning tailored for individual large language model architectures, which cannot be transferred directly across different models like Mistral [12] and Qwen [11]. Conversely, standard RAG architectures are capable of finding relevant information but feed it blindly to the model's middle context, where the model ignores it. **The critical fault in existing works is that they fail to give**

importance to where the retrieved information is placed as much as they give importance to accurately retrieving it. When it comes to domains like programming, fetching the correct information is of no use if the model ends up ignoring it altogether.

The proposed framework **CAUS-RAG** (Confidence-Aware Adaptive U-Shape) overcomes these challenges by acting as an interceptor that works between the information retrieval and output generation stages. Compared to prior works, our framework focuses equally on the accuracy of data retrieved as well as where it should go, considering the attention strength of individual model architectures.

The main contributions introduced in this paper are the following:

1) The placement of information retrieved is a core problem to solve, considering that large language models tend to prefer certain positions and show that both retrieval and placement must be considered as a single joint track for quality output generation.
2) A PSI (Position Sensitivity Index) score is presented, which calculates the attention bias of individual architectures, without changing any model parameters.
3) A U-shaped method of arrangement is introduced, which arranges retrieved information based on the Hybrid Confidence-Scoring mechanism [5] so that the most relevant information is placed where the model pays the most attention [1].
4) The experiments conducted show significant improvement in syntactic accuracy without any training and across different models, boosting the Exact Match performance of models such as Qwen-2.5 from 17.3% to 86.5%.

II. Related Work

The main challenge in accurate code generation via RAGs exists because of the positional bias present in transformer architectures. This section highlights the previous work done in this field and points out their limitations, which are addressed by CAUS-RAG.

A. Evolution of RAG and Hybrid Retrieval

Retrieval-Augmented Generation (RAG) [2] helps Large Language Models to fetch external non-parametric knowledge rather than relying only on built-in parametric memory while generating output responses. This is made possible in a standard RAG through embedding-based retrieval, where both queries and documents are translated into vector embeddings and then ranked based on similarity scores.

Although this method proves beneficial under normal circumstances like open-domain question answering, such retrievals lack lexical accuracy, often retrieving syntactically incorrect documents in technical domains [6]. BM25, a hybrid retrieval method, solves this problem by combining the semantic understanding of dense embeddings with the exact keyword matching of sparse techniques [5]. CAUS-RAG works fundamentally based on this principle by introducing a confidence-aware fusion of semantic and syntactic scores and context-aware placement of retrieved pieces of information.

B. Positional Bias and Mitigation Strategies

The "Lost-in-the-Middle" effect [1] shows that transformer architectures struggle to pay attention to the middle of a long input context, thus creating a U-shaped performance curve where priority is given to tokens at the beginning (primacy bias) and end (recency bias) of the input. This acts as a major obstacle in code generation, in which, in spite of retrieving correct documentation, it may be placed in low-attention regions.

Current techniques to reduce positional bias in transformer-based LLMs only focus at the model or prompt level. Model training-based techniques, such as position-agnostic training [9] and attention calibration [8], are computationally expensive and do not work uniformly when transferred across various architectures. Prompt-based methods [10] are more flexible but are currently inconsistent with the results. Conversely, the proposed method CAUS-RAG reduces positional bias by directly optimizing context placement based on the attention strength of individual large language models.

C. Syntactic Constraints in API Generation

Code creation requires strict syntactic accuracy, where even a small mistake, for example, in a function name or argument, can cause failure in execution. This is correctly pointed out by the Gorilla framework [7], which shows that semantic correctness alone is insufficient for error-free technical code generation. Further, traditional metrics such as BLEU and ROUGE are incapable of being used for benchmarking since they fail to capture exact token-to-token matches.

Since the Exact Match (EM) metric [14] demands token-to-token identicality, even small differences are treated as errors. This makes positional bias a critical factor, as it means that if relevant information is not placed in the most attended regions of the prompt, it leads to wrong output generation.

*Summary:*Existing works in this field consider retrieval quality and positional bias as two separate issues [4]. CAUS-RAG introduces a unified approach through a training-free interceptor framework that jointly optimizes retrieval scoring and context placement.

III. Proposed Methodology

The CAUS-RAG architecture is an algorithmic interceptor that operates between the information retrieval and output generation stages. This section defines the input-to-output process via the proposed pipeline and explains the two-stage (dual-gear) process behind this refinement.

A. System Architecture Overview

CAUS-RAG operations take place in two stages. In the first step, a hybrid confidence score, based on semantic and syntactic match, is assigned to each retrieved document. In

the second stage, it ranks these documents into a U-shaped prompt structure, placing the chunks with the highest scores in areas where the model pays the most attention. A Positional Sensitivity Index (PSI) probe is employed to find the positional bias behaviour of individual model architectures. The CAUS-RAG architecture is shown in Fig. 2.

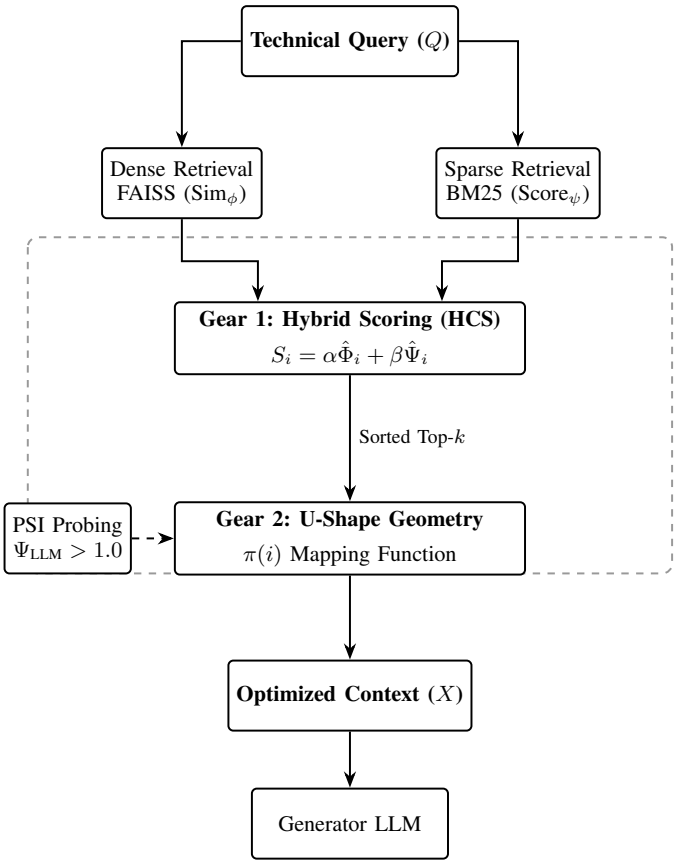

Fig. 2. CAUS-RAG architecture showing the dual-gear interceptor: Gear 1 performs hybrid confidence scoring, and Gear 2 applies PSI-based U-shape context reranking.

As shown in Fig. 2, the two retrieval pipelines , FAISS and BM25, run in parallel for the same query, and their respective scores are combined in Gear 1. The merged result score is then passed to Gear 2, which rearranges them using a permutation mapping $\pi(i)$. This reranking is performed only if the PSI probing step detects a U-shaped positional bias in the target model architecture. The final optimized context X is then forwarded directly to the generator LLM.

B. Objective Function and Problem Formalization

Let Q be a technical query and $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ be the set of top-k documents retrieved from a corpus $\mathcal{C}$. The objective is to find an optimal permutation π^* from the symmetric group S_k that reorders $\mathcal{D}$ into a context X such that:

$$\pi^* = \arg\max_{\pi \in S_k} P(R = R^* \mid Q, X_\pi), \quad X_\pi = f_{\text{CAUS}}(Q, \mathcal{D}, \pi) \tag{1}$$

where R^* is the ground-truth API syntax. The function f_{CAUS} minimizes the impact of positional decay [1] with the placement of high-confidence-score documents to the context regions the model pays attention to. The number of possible permutations grows exponentially with k, which makes exploring all of them impractical. However, the hybrid scoring step addresses this problem by reducing them into a simpler ranking task. The score S_i acts as an indicator of each document's relevance, and the U-shaped mapping then assigns positions based on this ranking.

C. Numerically Stable Hybrid Confidence-Scoring (HCS)

Combining dense and sparse scores requires scale normalization, as FAISS similarities [4] are bounded, $[-1, 1]$ while BM25 scores [5] are unbounded. Simply averaging the scores can make the signals with high variance dominate. To ensure both signals contribute fairly, Min–Max normalization is used. For document d_i, the hybrid score S is:

$$\begin{aligned} S(Q, d_i) = \alpha \cdot \frac{\text{Sim}_\phi(i) - \min(\Phi)}{\max(\Phi) - \min(\Phi) + \epsilon} \\ + \beta \cdot \frac{\text{Score}_\psi(i) - \min(\Psi)}{\max(\Psi) - \min(\Psi) + \epsilon} \end{aligned} \tag{2}$$

where Φ and Ψ are the dense and sparse score sets for the top-k documents, and $\epsilon = 10^{-7}$ prevents division by zero. The condition $\alpha + \beta = 1$ is enforced, and $\beta > 0.5$ is set to prioritize the lexical signal. In API generation, producing executable codes depends on getting the exact token sequences right. Methods like BM25 prioritize this exact term overlap, especially when compared to relying only on embedding-based retrieval. [7].

D. Positional Sensitivity Index (PSI)

Before Gear 2 applies reordering, the interceptor measures the model's sensitivity to the "Lost-in-the-Middle" effect using the Positional Sensitivity Index (Ψ_{LLM}):

$$\Psi_{\text{LLM}} = \frac{\mu(EM_{\text{top}}) + \mu(EM_{\text{bot}})}{2 \cdot \mu(EM_{\text{mid}}) + \epsilon} \tag{3}$$

where $\mu(EM_{\text{top}})$, $\mu(EM_{\text{mid}})$, and $\mu(EM_{\text{bot}})$ denote the mean Exact Match scores when the ground-truth document is placed at the beginning, middle, and end of a fixed-length context, respectively. The PSI is calculated once per architecture and treated as a model-level constant.

$\Psi_{\text{LLM}} > 1.0$ shows that model is paying higher importance to its edge positions, confirming the presence of a U-shaped attention behaviour. In such cases, Gear 2 applies the reordering of contexts. Otherwise, if $\Psi_{\text{LLM}} \leq 1.0$, the documents are simply used in their ranked order without any reordering.

E. U-Shape Permutation Mapping

Given the sorted document list $\mathcal{D}_{\text{sorted}}$ where $S_1 \geq S_2 \geq \cdots \geq S_k$, Gear 2 applies the mapping $\pi(i)$:

$$\pi(i) = \begin{cases} 2i - 1, & \text{if } 1 \leq i \leq \lceil k/2 \rceil \\ 2(k - i + 1), & \text{if } \lceil k/2 \rceil < i \leq k \end{cases} \tag{4}$$

This U-shaped mapping ensures that high-confidence documents are kept at the end positions and that lower-confidence documents are placed in middle positions. For example, if $k = 3$ and the ranking is $d_1 > d_2 > d_3$, then the re-ranked arrangement of the documents will be $[d_1, d_3, d_2]$, which ensures the most important documents are placed in positions where the model is more likely to pay attention.

F. Computational Complexity and Algorithm

Algorithm 1 shows how the CAUS-RAG interceptor combines hybrid confidence scoring and model position bias for context reordering. Let $\mathcal{D}_{\text{pool}}$ be the merged confidence results and $\mathcal{D}_{\text{top}}$ be the top-k ranked retrieved chunks.

Algorithm 1 CAUS-RAG Interceptor Logic

Input: The method receives a technical query Q, the document collection $\mathcal{C}$, a chosen retrieval depth k, the hybrid scoring weights α and β, and the model's Positional Sensitivity value Ψ_{LLM}. **Output:** Optimized Context Sequence X

Phase 1: Hybrid Confidence Scoring (Gear 1)
$\mathcal{D}_{\text{dense}}, \Phi \leftarrow \text{DenseSearch}(Q, \mathcal{C}, k)$
$\mathcal{D}_{\text{sparse}}, \Psi \leftarrow \text{SparseSearch}(Q, \mathcal{C}, k)$
$\mathcal{D}_{\text{pool}} \leftarrow \mathcal{D}_{\text{dense}} \cup \mathcal{D}_{\text{sparse}}$

for each $d_i \in \mathcal{D}_{\text{pool}}$ **do**
    $\hat{\Phi}_i \leftarrow \text{MinMaxNorm}(\Phi_i, \Phi)$
    $\hat{\Psi}_i \leftarrow \text{MinMaxNorm}(\Psi_i, \Psi)$
    $S_i \leftarrow \alpha\hat{\Phi}_i + \beta\hat{\Psi}_i$
end for

$\mathcal{D}_{\text{sorted}} \leftarrow \text{SortDescending}(\mathcal{D}_{\text{pool}} \text{ by } S)$
$\mathcal{D}_{\text{top}} \leftarrow \mathcal{D}_{\text{sorted}}[1 \dots k]$ ▷ Select top-k candidates

Phase 2: Position-Aware U-Shape Mapping (Gear 2)
if $\Psi_{\text{LLM}} > 1.0$ **then** ▷ Positional bias detected
    Initialize array X of length k
    $left \leftarrow 1, \quad right \leftarrow k$
    for $i \leftarrow 1$ **to** k **do**
        if i is odd **then**
            $X[left] \leftarrow \mathcal{D}_{\text{top}}[i]$
            $left \leftarrow left + 1$
        else
            $X[right] \leftarrow \mathcal{D}_{\text{top}}[i]$
            $right \leftarrow right - 1$
        end if
    end for
else ▷ No positional bias detected
    $X \leftarrow \mathcal{D}_{\text{top}}$
end if

return X

The final sequence X is the reordered context passed to the generator LLM. Normalization, hybrid scoring, and permutation take $\mathcal{O}(k)$ time, while sorting takes $\mathcal{O}(k \log k)$. These operations are independent of model parameters, which thereby add negligible cost compared to LLM inference, making CAUS-RAG practical for code generation tasks.

IV. Experimental Results

A. Experimental Setup

The experiments were conducted on a PyTorch API benchmark comprising 52 queries, each requiring exact matches for method names, signatures, and arguments. These queries are divided into three difficulty levels, ranging from basic operations to advanced APIs whose signatures change from time to time. The retrieval dataset consists of over 1,500 documentation chunks, indexed with FAISS [4] and generated using `nomic-ai/nomic-embed-text-v1` embeddings.

The evaluation includes three open-weight models: **Phi-3.5 (3.8B)** [13], **Mistral-7B-v0.3** [12], and **Qwen-2.5-7B** [11]. For CAUS-RAG, the parameters are set to $\alpha = 0.3$, $\beta = 0.7$, with a retrieval depth of $k = 10$. Performance is measured using Exact Match (EM) [14] and Keyword Match (KW), where EM is the primary evaluation metric.

B. Comparative Performance Analysis

To study the contribution of each component individually, three different architectural methods are defined:

1) **Base RAG:** Ranking of retrieved chunks only based on the semantic similarity using FAISS [4].
2) **Static U-RAG:** U-shape context rearrangement applied directly to the semantically ranked retrieved documents, without hybrid scoring or PSI.
3) **CAUS-RAG (Proposed):** An interceptor with Hybrid Confidence Scoring that creates a U-shape mapping based on the model's PSI value.

Table I shows the results across all three architectures across all three methods.

TABLE I
Comparative performance analysis demonstrating Exact Match (EM) and Keyword Match progression across Base RAG, Static U-RAG, and CAUS U-RAG frameworks.

Model	RAG Type	Exact Match	Keyword Match
Phi-3 (3.8B)	Base RAG	11.5%	21.2%
	Static U-RAG	13.5%	25.0%
	CAUS U-RAG	**71.2%**	**75.0%**
Mistral (7B)	Base RAG	15.4%	44.2%
	Static U-RAG	17.3%	53.8%
	CAUS U-RAG	**78.8%**	**80.8%**
Qwen-2.5 (7B)	Base RAG	17.3%	73.1%
	Static U-RAG	21.2%	78.8%
	CAUS U-RAG	**86.5%**	**90.4%**

Table I shows three patterns. First, Base RAG performs poorly across all models, with an average Exact Match (EM) of about 14.7%. This shows that the issue is not in retrieving the right information, but in how that information is placed before being passed to the generator LLM. Secondly, moving from Base RAG to Static U-RAG without changing the retrieval step makes Mistral improve from 15.4% to 17.3%, and Qwen from

17.3% to 21.2%. These large gains are thanks to simply placing important information at the edges, supporting the "Lost-in-the-Middle" effect [1]. Third, moving to the full CAUS-RAG setup further boosts performance, which clearly indicates the importance of hybrid scoring. Qwen-2.5-7B reaches 86.5% EM, a 69.2% absolute improvement. Incorporating BM25 [5] helps capture exact token matches. This helps in ranking important documents higher before placing them in attention-dominant edge positions of the model. Overall, these results show that both where information is placed and how it is ranked are critical, and therefore they need to be addressed jointly. Experimentations conducted on Phi-3.5 also confirm this pattern. Starting at 11.5% EM with Base RAG, it suffers more because of poor context ordering than larger models. Static U-RAG offers only a slight improvement (13.5%), whereas CAUS-RAG boosts it to 71.2%, indicating that both accurate retrieval and optimal placement are necessary for smaller models.

C. Ablation Study I: Hybrid Scoring Weights (α, β)

To find the optimal balance between the semantic (embedding-based) and lexical signals in Gear 1, an ablation study was run on Qwen-2.5-7B with $k = 10$, averaged across all 52 queries. The results are shown in Table II.

TABLE II
IMPACT OF HYBRID CONFIDENCE-SCORING WEIGHTS ON QWEN-2.5-7B GENERATION ACCURACY ($k = 10$).

Configuration Focus	Dense (α)	Sparse (β)	EM
Dense-Heavy	0.9	0.1	82.7%
Balanced Fusion	0.5	0.5	86.5%
Sparse-Heavy (Optimal)	**0.3**	**0.7**	**88.5%**

As Table II shows, even when the configuration is heavily biased toward the semantic signal ($\alpha = 0.9$), the model still achieves 82.7% EM. This surpasses the Base RAG and Static U-RAG results in Table I, which shows that the U-shaped method of arrangement is capable of doing a good job when it comes to increasing the precision in output generation without even performing PSI scoring. When the weighting given to both semantic and lexical signals is equal ($\alpha = \beta = 0.5$), performance rises to 86.5%. Shifting further toward the lexical side ($\beta = 0.7$) pushes it up to 88.5%. The roughly 5.8-point difference between the dense-heavy and sparse-heavy setups confirms that while semantic retrieval helps identify the correct topic, it often fails to capture the exact token sequences needed for accurate API generation. To solve this dilemma, BM25 [5] helps to boost the Exact Match by fetching documents that contain the precise tokens needed. These documents are then given more weight and ranked higher in the U-shape mapping to increase their effective usage.

D. Ablation Study II: Retrieval Depth (k) Analysis

A second ablation study shows how CAUS-RAG behaves when the number of retrieved documents varies ($k = 3$ to $k = 15$). In standard RAG, increasing k often reduces its performance by involving more irrelevant chunks of information. Table III shows that CAUS-RAG performance is not affected under such circumstances.

TABLE III
PERFORMANCE OF CAUS-RAG ACROSS VARIOUS RETRIEVAL DEPTHS ON QWEN-2.5-7B.

Depth (k)	**3**	**5**	**7**	**10**	**12**	**15**
EM	84.6%	84.6%	88.5%	84.6%	90.4%	86.5%

As shown in Table III, performance remains stable across all depths, with a minimum of 84.6% EM and a peak of 90.4% at $k = 12$. This is made possible because of U-shape mapping $\pi(i)$ that places documents with high hybrid scores at edge positions of the model where it pays most attention. Further, it is clear that increasing k does not displace the relevant information, which shows that the CAUS-RAG framework performance is unaffected even when the optimal retrieval depth is unknown.

V. RESULTS AND DISCUSSIONS

The experiments conducted clearly depict a major flaw in standard RAG pipelines when it comes to code generation [2]. As seen from the initial baselines, Base RAG performs poorly across all models. This is not because the system fails to retrieve relevant information; Keyword Match (KW) scores (21–73%) show that the right tokens are often present, but rather due to wrong placement. When important documentation is placed in the middle of the context, it is given lesser priority, causing the model to rely back on its internal (parametric) knowledge. The keyword accuracy comparison is shown in Fig. 3.

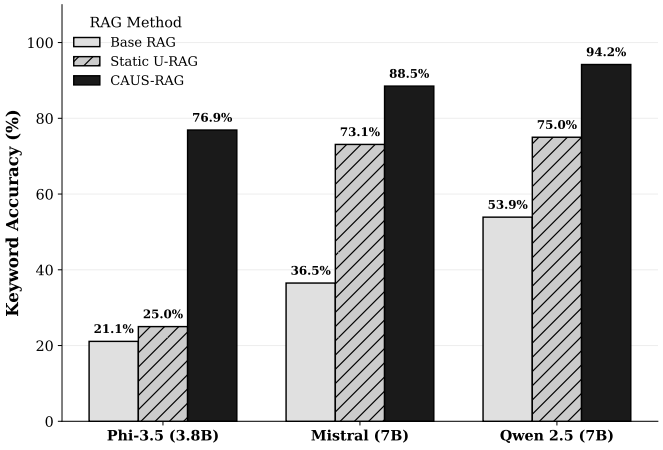

Fig. 3. Keyword accuracy performance in Base RAG, Static U-RAG, and CAUS-RAG configurations for Phi-3.5, Mistral, and Qwen models.

The relationship between keyword match and exact match is illustrated in Fig. 3 which shows that in Base RAG, there is a big gap between KW and EM (e.g., Qwen: 73.1% vs 17.3%), which means that the model is capable of retrieving the right tokens but fails to efficiently make use of them due to incorrect placement. Conversely, this gap significantly reduces in CAUS-RAG (90.4% vs 86.5%).

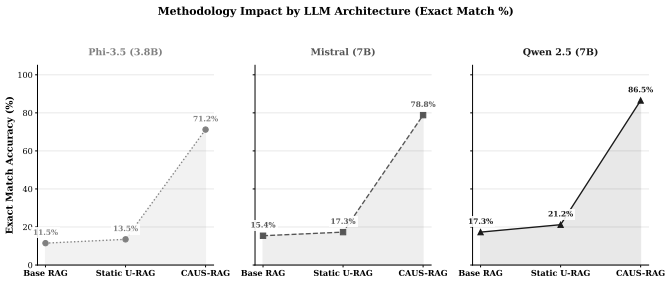

Fig. 4. Exact Match (%) trajectories across Base RAG, Static U-RAG, and CAUS-RAG configurations in Phi-3.5, Mistral, and Qwen models.

Fig. 4 shows that EM improves across all setups. Larger models like Mistral and Qwen show the most gain, while Phi-3.5 improves mainly at the final stage, where both retrieval quality and context placement are jointly optimized.

The usage of $\alpha = 0.3$ and $\beta = 0.7$ in Gear 1 gives more importance to the role of lexical signals. This, in turn, causes the 69.2% improvement in models like Qwen ($p < 0.01$), indicating that the improvement is not caused by variations in the dataset but rather due to the combined effect of correct retrieval based on hybrid scoring and their correct placement. Also, as the interceptor operates between retrieval and generation, it can be used with any LLM without changing model parameters. The overall overhead is also minimal compared to LLM inference, about $\mathcal{O}(k \log k)$ for sorting and permutation [7].

A. Error Analysis

There are two patterns seen in CAUS-RAG's error analysis. The first is *near-miss errors*, where the model identifies the correct function family but makes small mistakes in the arguments, for example, generating `torch.linalg.eigvals` instead of `torch.linalg.eig`. These contribute to 60% of the remaining EM failures and explain why the gap between KW and EM does not completely vanish. The second type is *confabulation errors*, where the model relies on its internal parametric knowledge instead of using the retrieved context, leading to outdated or incorrect API calls. This often happens in complex queries (e.g., distributed training APIs), where strong prior knowledge can override the prompt information given in the prompt.

VI. Conclusion and Future Work

This paper introduces Confidence-Aware Adaptive U-Shape (CAUS), a framework introduced to the RAG pipeline in which an algorithmic interceptor operates between the retrieval and generation stages. It solves two key challenges: the limitation of retrieval based on semantic similarity alone when it comes to capturing lexically accurate API details, and the tendency of transformer models to pay less attention to information placed in the middle of the context. The combination of hybrid confidence scoring and U-shaped arrangement of the retrieved context, based on a Positional Sensitivity Index, ensures that relevant documents are not only retrieved accurately but also placed correctly.

Experimental results across the three models show that both components play a clear role in the overall improvement. The U-shaped arrangement alone increases Exact Match compared to Base RAG, while the complete CAUS-RAG approach leads to much larger gains. For example, Qwen-2.5-7B improves from 17.3% to 86.5%, Mistral-7B from 15.4% to 78.8%, and Phi-3.5 from 11.5% to 71.2%. Further analysis through ablation studies shows that giving more weight to the lexical signal ($\beta = 0.7$) is the right choice when it comes to technical code generation (Table II). It also shows that performance stays above 84% EM even when the retrieval depth increases (Table III), indicating that the method remains stable even when the retrieval depth increases.

Limitations: CAUS-RAG has limitations when it comes to the retrieval behaviour. If an exact match is not present in the top-k results, reranking is meaningless. The approach also assumes a consistent U-shaped attention pattern, which may not be true for newer architectures. Further, the current evaluation is limited to code generation tasks.

Future work: As for the next step, moving to a query-adaptive weighting instead of a fixed α/β ratio using PSI and extending CAUS-RAG are considered. Applications to various other domains such as legal QA, SQL generation, and medical QA is also explored.

References

[1] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, 2024.

[2] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. NeurIPS*, 2020, pp. 9459–9474.

[3] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[4] J. Johnson, M. Douze, and H. J'egou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[5] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, 2009.

[6] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," in *Proc. ACL*, 2024.

[7] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive APIs," in *Proc. ICLR*, 2024.

[8] C.-Y. Hsieh *et al.*, "Found in the middle: Calibrating positional attention bias improves long context utilization," in *Findings ACL*, 2024, pp. 14982–14995.

[9] J. He *et al.*, "Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training," in *Proc. ACL*, 2024, pp. 13628–13642.

[10] M. Zhang, Z. Meng, and N. Collier, "Can we instruct LLMs to compensate for position bias?" in *Findings EMNLP*, 2024, pp. 12545–12556.

[11] Qwen Team, "Qwen2.5 technical report," arXiv preprint arXiv:2412.15115, 2024.

[12] A. Q. Jiang *et al.*, "Mistral 7B," arXiv preprint arXiv:2310.06825, 2023.

[13] M. Abdin *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," arXiv preprint arXiv:2404.14219, 2024.

[14] M. Chen *et al.*, "Evaluating large language models trained on code," arXiv preprint arXiv:2107.03374, 2021.