Fortify the Shortest Stave in Attention: Enhancing Context Awareness of Large Language Models for Effective Tool Use

Yuhan Chen[1*], **Ang Lv**[1*], **Ting-En Lin**[2], **Changyu Chen**[1]
Yuchuan Wu[2] , **Fei Huang**[2], **Yongbin Li**[2†] and **Rui Yan**[1†]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Alibaba Group
{yuhanchen, anglv, chen.changyu, ruiyan}@ruc.edu.cn
{ting-en.lte, shengxiu.wyc, f.huang, shuide.lyb}@alibaba-inc.com

arXiv:2312.04455v1 [cs.CL] 7 Dec 2023

Abstract

Recent advancements in large language models (LLMs) have significantly expanded their functionality and skills as tool agents. In this paper, we argue that a waveform pattern in the model's attention allocation has an impact on the tool use performance, which degrades when the position of essential information hits the trough zone. To address this issue, we propose a novel inference method named *Attention Buckets*. This approach enables LLMs to handle context by conducting parallel processes, each featuring a unique RoPE angle base that shapes the attention waveform. *Attention Buckets* ensures that an attention trough of a particular process can be compensated with an attention peak of another run, reducing the risk of the LLM missing essential information residing within the attention trough. Our extensive experiments on the widely recognized tool use benchmark demonstrate the efficacy of our approach, where a 7B-parameter open-source model enhanced by *Attention Buckets* achieves SOTA performance on par with GPT-4. [1]

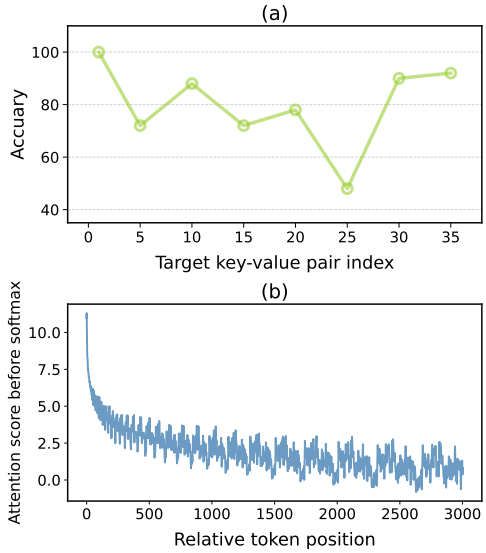

Figure 1: (a) Feeding the Llama JSON key-value pairs, we adjust the target key's index within, observing substantial variations in key-value retrieval accuracy based on this index. (b) Disregarding semantics, the attention score before softmax fluctuates, referred to as the "attention waveform" in this paper.

1 Introduction

Recent works that augmenting large language models (LLMs, e.g., GPT series [1, 2, 3]) with tools have achieved advancements in various fields, such as human-computer interactions [4, 5], automating multi-modal tasks [6, 7], and enhancing the overall efficiency of language-related applications [8]. In this paradigm, upon receiving a user's intent, a large language model accesses multiple tools (APIs), selects the most suitable one by referring to the relevant tool documentation, and provides an accurate and suitable response. Given the incorporation of vast information into the context, tool use tasks demand a high level of understanding and awareness from LLMs towards complex contexts.

Despite the achievements made by current LLM-based tool use frameworks, our observations indicate that LLMs exhibit varied levels of awareness towards different positions within the context, which can potentially impact their performance in tool use. This observation stems from a simple experiment following [9], in which we provided the Llama-2-7B [10] with multiple key-value pairs in JSON format and prompted it to generate the corresponding value for a given key. Figure 2 presents a test case for this task. When we altered the position of the target key within the context, there were

*Equal contribution.
†Corresponding authors.
[1]We release our code at https://github.com/AlibabaResearch/DAMO-ConvAI/attention-buckets.

noticeable variations in accuracy, as illustrated in Figure 1(a). We posit that these performance differences are tied to the model's fluctuating attention allocation. Specifically, we noticed that the upper bound of the attention score takes a waveform pattern, as depicted in Figure 1(b). If essential information coincides with a trough in this attention waveform, the model may overlook it, potentially leading to decreased accuracy. We confirm this hypothesis in §2.2.

We argue that by shifting essential information away from the attention waveform's trough zone, we can reduce the risk of LLMs[2] missing key details, thus enhancing the efficacy of tool use. Given the challenge of pinpointing important information within a context in real-world scenarios, we suggest altering the waveform pattern to capture potentially important data. To achieve this, we modify the RoPE's rotary angle base, a crucial factor influencing the waveform pattern (see §2.1 for details). Our approach involves processing the context through several parallel executions, each characterized by a unique rotary angle base of the RoPE, resulting in varied attention allocations. By interleaving the attention waveform peaks from different runs, we enable the LLM in each run to focus on various context positions that collectively provide a comprehensive understanding. We aggregate the output distributions from all these parallel executions and compute their weighted sum. This sum is then decoded to produce the final prediction token.

An analogy can aid in understanding our approach: Imagine a wooden bucket with some shorter staves, which allow water to leak out. Similarly, the attention mechanism, at each angle base, has limited awareness of specific positions in the context. We utilize models to process the context with different angle bases. This results in the troughs of one attention wave being fortified by the peaks of another, analogous to how the longer staves in one bucket compensate for the shorter staves in another. Consequently, we name our proposed method *Attention Buckets*.

We achieve the state-of-the-art on the largest tool use benchmark [4], where we augment the performance of a 7B LLM to levels competitive with those of GPT-4 [3]. In addition to our achievements in tool use, we also demonstrate our method's potential in general retrieval-augmented generation (RAG) tasks, which also demand a high degree of contextual awareness. In summary, we make three major contributions:

(1) For LLMs with RoPE, we propose and verify an explanation for the variation in their awareness of different positions within the context. We establish a relationship between this variation and the fluctuation in the upper bound of attention score.

(2) By leveraging the insights from our proposed explanation, we develop a novel approach *Attention Buckets* to enhance LLMs' context awareness.

(3) Through extensive experiments, we empirically validate the efficacy of our proposed method.

[2]In this paper, we focus on LLMs based on Transformer models [11] and rotary position embeddings (RoPE [12]). This family of LLMs include many popular models like Llama [13, 10], Qwen [14], Baichuan [15], etc.

Input Context

Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"1e0029ce- … -f1ed9642d893": "3d678cff- … -2950de83f31c",
"da448545- … -bcb1d03a2254": "89e4a63e- … -c7c96e4c8b9d",
"**a5d4c9ee- … -54ea28f8bf21**": "dc9708f9- … -c6aeb3606222",
…
"0194ec2b- … -728bab87b4f7": "5daf5bed- … -9c134fb7745a",
"2ac4aebd- … -76a333d48489": "3945b582- … -3dbb44d3162b"}

Key: "**a5d4c9ee-14e6-4f48-9ad0-54ea28f8bf21**"
Corresponding value:

Figure 2: A task illustration of generating the value of the given key. The model needs to retrieve the correct key from the long context and precisely generate it.

2 Attention Waves Impact on Context Awareness

2.1 Preliminaries

Rotary position embedding (RoPE) [12] stands as a prevalent technique for position encoding in large language models with Transformer backbone [11]. During the attention calculation, given a query or key vector at position m in the sequence, RoPE serves to encode the position information into the vector via a d-dimensional rotation matrix denoted as $R_{\theta,m}$. This matrix $R_{\theta,m}$ is structured as a block diagonal matrix consisting of blocks with dimensions of 2×2, totaling $d/2$ such blocks. Specifically, the i-th block is defined as:

$$R_{\theta_i,m} = \begin{bmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{bmatrix}. \quad (1)$$

where $\theta_i = B^{-\frac{2i}{d}}$, with B is termed as the base of the rotary angle.

In each Transformer layer, after multiplying the query vector q_m at position m and the key vector k_n

at position n with the rotation matrix, the relative position is incorporated in their inner product (the attention score before softmax):

$$(R_{\theta,m}q_m)^\top(R_{\theta,n}k_n) = q_m^\top R_{\theta,n-m}k_n. \quad (2)$$

When the relative distance $n - m$ increases, the upper bound of the attention score before softmax demonstrates a long-term decay, i.e., the value generally decreases as the relative distance grows. This trend is accompanied by a waveform, as depicted in Figure 1(b). The derivation of the upper bound is presented in the appendix A.

As a widely-used position embedding technique in LLMs, many researchers found RoPE has a substantial impact in LLM's context utilization and awareness. RoPE has favorable properties that enhance the model in various aspects, such extrapolation [16, 17, 18, 19], efficient long-context model training [20, 21], and better understanding of training data [22]. In this paper, we leverage the attention waveform introduced by position embeddings, to enhance the context awareness of LLMs. We hypothesize that these waveform patterns might affect the model's context awareness. Intuitively, tokens located at troughs of the attention waveform would receive less focus. If such tokens are important for the current prediction, this could hamper the performance. In this section, we design an experiment to test this hypothesis.

2.2 Hypothesis Verification

Task and Data To test our hypothesis, we conducted an in-context retrieval test [9, 23]. We feed the Llama-2-chat-7B [10] with K synthetic key-value pairs in JSON format. Each key and value is a distinct UUID string [24]. We then prompt the model to retrieve the value corresponding to the key we specify. We evaluated the model's context awareness based on the accuracy of the value it generates. Figure 2 shows a test example.

Experiment Design We varied the RoPE base within the model from $10,000$ to $30,000$ in increments of $5,000$. For each base value, we calculate the positions of the peaks and troughs in the corresponding upper bound of attention score. The specific calculations are provided in the appendix B. Each test sample undergoes two evaluation rounds: In the first round, we position the target key-value pair at an attention peak in the middle of the context. In the second round, we move the target pair to the nearest attention trough. By comparing accuracy differences between the two rounds, we aimed to answer how much attention waveform patterns impact the model's context awareness. The experiment was conducted with varying context lengths by setting different K (40 and 50, respectively). We provide more experimental details in appendix C.

Table 1: The results of the in-context key-value retrieval. The generation accuracy provides insight into the model's awareness of information at both the peaks and troughs of the attention waveform.

Base	K=40		K=50	
	Peak Acc	Trough Acc	Peak Acc	Trough Acc
10,000	79.8	76.8	47.6	44.0
15,000	96.2	96.6	75.8	75.2
20,000	85.2	85.0	82.6	80.4
25,000	70.8	70.0	59.2	55.6
30,000	62.2	57.6	51.8	24.4

2.3 Results and Analysis

Our experimental findings, as detailed in Table 1, reveal a performance trend associated with varying RoPE base values: we observe an initial rise followed by a subsequent fall. Notably, placing a key-value pair at the peak of the attention waveform consistently yields better outcomes than positioning it at the trough. This holds true across different context lengths (as defined by K) and base values. Also, the results suggest that the optimal base values differ depending on the context length. For example, with K set at 40, the best performance is achieved with a base of 15,000, while a base of 20,000 is most effective when K is 50.

Based on these results, we can draw the following insight and its associated challenge:

Insight: Enhancing the attention to information positioned at the troughs of the attention waveform could make the context awareness of large language models more robust, potentially leading to improved overall performance.

Challenge: In practical applications, pinpointing the location of critical information is difficult. This makes it challenging to select a RoPE base that ensures attention to the crucial information.

3 Enhancing Context Awareness via Interleaving Attention Waveform

Based on the insights presented above, we introduce a novel approach to sidestepping the above challenge, with the goal of improving the LLM

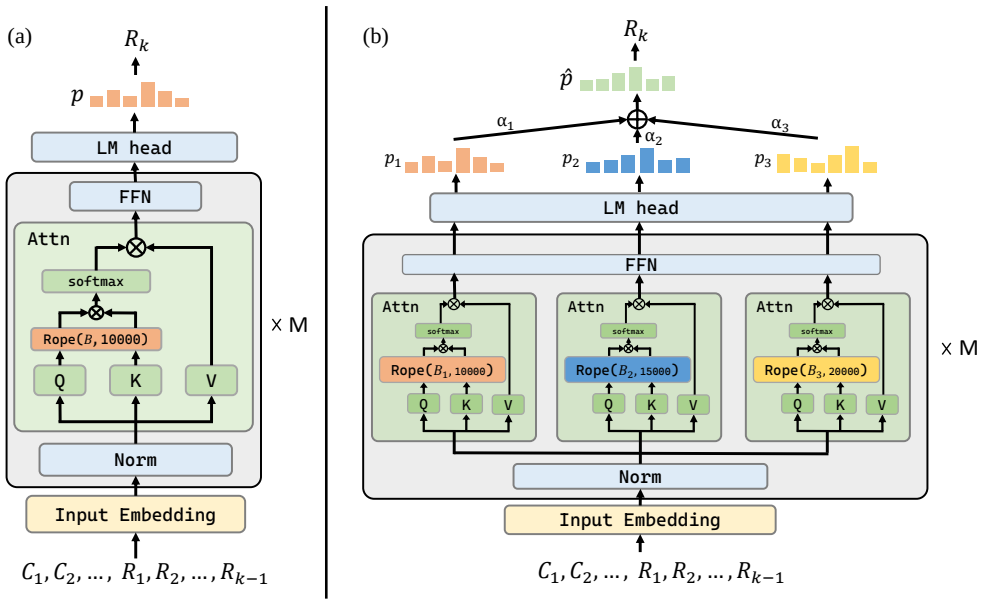

Figure 3: (a) The overview of how a typical Transformer-based Large Language Model (LLM) generates the next token based on context $\mathcal{C}$. This LLM comprises M layers, though for simplicity, only the inner workings of a single layer are shown. (b) The overview of our proposed *Attention Buckets* augmenting the context awareness of LLMs: Upon receiving context $\mathcal{C}$, it creates N (specifically 3 in this example) parallel copies for processing. Each parallel stream employs a distinct RoPE base. The resulting output distributions p_j are weighted and summed based on the prediction confidence α_j, culminating in the final predicted distribution $\hat{p}$ used for decoding the next token.

performance in tool use by enhancing its context awareness. Our method focuses on the inference stage of LLMs and does not require training. We first provide preliminary definitions, followed by a detailed introduction to our approach.

3.1 Preliminaries

In tool use, fulfilling a user's intent typically involves multiple turns, such as selecting tools, calling for APIs, and engaging with the user across multiple interactions. In this section, our introduction focuses on one single turn since the multi-turn scenario is a simple amalgamation of this single-turn scenario. We consider all information from previous turns, including API responses, tool execution outcomes, and user feedback, as the model's context for the current turn, which we represent as $\mathcal{C}$. Subsequently, the LLM proceeds to generate a response, denoted as $\mathcal{R}$, in an autoregressive manner based on $\mathcal{C}$. To denote the specific tokens within $\mathcal{C}$ or $\mathcal{R}$, we employ the notation $\mathcal{C}_k$ or $\mathcal{R}_k$, respectively, where k represents the token's index.

3.2 Method

Given an input context $\mathcal{C}$, our approach involves duplicating this context into N copies, forming a batch that allows for parallel processing by the LLM. In each parallel, each of these N copies is individually processed with a distinct RoPE base B_j from a base set $\mathcal{B}_c$, resulting in N corresponding predicted distribution p over the vocabulary V. Our selection of $\mathcal{B}_c$ guarantees that an attention trough in one parallel is compensated by a peak in another, effectively reducing the possibility of the LLM missing essential information residing within an attention trough. We will delve into the details of determining $\mathcal{B}_c$ in § 3.3.

We posit that in the parallel run indexed by j, if the model focuses its attention on crucial information it currently requires, it has more confidence to make accurate predictions for the next token in the response $\mathcal{R}$. We quantify the model's confidence on prediction α_j as:

$$\alpha_j^{'} = \max_{v \in V} p(\mathcal{R}_k = v | \mathcal{C}, B_j, \mathcal{R}_{1:k-1}),$$
$$\alpha_j = \frac{e^{\alpha_j^{'}}}{\sum_{i=1}^{N} e^{\alpha_i^{'}}}. \tag{3}$$

Next, we compute a weighted sum of each run's output distribution p_j to derive the final predicted distribution $\hat{p}$. The weighting of each p_j depends on its corresponding confidence score α_j:

$$\hat{p} = \sum_{j}^{n} \alpha_j * p_j. \tag{4}$$

Algorithm 1 The searching algorithm of $\mathcal{B}_c$.

Input:
- P_c and T_c: Sets containing the peak and trough positions in attention waveforms corresponding to items in $\mathcal{B}_c$. These positions are calculated by functions f_p and f_t (see appendix B), respectively.
- Searched set $\mathcal{B}_c$, initialized as $\{B_{\text{train}}\}$.
- Search space $\mathcal{B}_s$, initialized using Eq. 5.

$P_c \leftarrow f_p(B_{\text{train}}), T_c \leftarrow f_t(B_{\text{train}})$
while $|\mathcal{B}_c| < N$ **do**
  for B_j in $\mathcal{B}_s$ **do**
    $P_j \leftarrow f_p(B_j), T_j \leftarrow f_t(B_j)$
    $d_j \leftarrow \sum_{\substack{p_{i,j} \in P_j \\ t_{i,c} \in T_c}} |p_{i,j} - t_{i,c}| + \sum_{\substack{t_{i,j} \in T_j \\ p_{i,c} \in P_c}} |t_{i,j} - p_{i,c}|$
  end for
  $\mathcal{B}_c \leftarrow \mathcal{B}_c \cup \{B'_j$ with the minimum $d_j\}$.
  $P_c \leftarrow P_c \cup f_p(B'_j), T_c \leftarrow T_c \cup f_t(B'_j)$
end while
Output: $\mathcal{B}_c$.

We decode a predicted token from $\hat{p}$. This token is incorporated into the preceding context, and this auto-regressive process persists until the current turn ends.

3.3 The Searching of $\mathcal{B}_c$

This section details our methodology for searching $\mathcal{B}_c$, an appropriate set of RoPE bases. Our goal is to develop strategies ensuring that the attention waveform troughs of any given base overlap with peaks from different bases, and vice versa. We do not consider peak and trough positions exceeding the maximum pre-trained context length in this paper. Firstly, we define a discrete base search space, denoted as:

$$\mathcal{B}_s = \left\{ B_i \middle| B_i = B_{\text{min}} + i \times T,\ i \in \left(0, \frac{B_{\text{max}} - B_{\text{min}}}{T}\right] \right\}, \tag{5}$$

where B_{min} and B_{max} represent the minimum and maximum base values, and T is the search stride. In our experimental setup, we set B_{min} equal to B_{train}, the base used during model pre-training. This decision is grounded in the consideration [18] that opting for a smaller base compared to the one used during pre-training could potentially introduce out-of-distribution (OOD) positional information, particularly in scenarios where the context spans a considerable length, as discussed in detail in the appendix D.

At the beginning of the search, we initialize $\mathcal{B}_c$ to $\{B_{\text{train}}\}$. For the following $N-1$ iterations, we search for a candidate base in each round to be included in $\mathcal{B}_s$. This candidate selection is based on the proximity of the candidate base's peak positions to the trough positions in the attention waveforms of the bases already in $\mathcal{B}_c$. We also expect the troughs of the candidate base to align closely with the peaks in the attention waveforms of the existing bases. The candidate that best matches these criteria is then incorporated into $\mathcal{B}_c$. Our searching algorithm is detailed in Algorithm 1.

3.4 Method Visualization

We present a visual demonstration of Algorithm 1 depicted in Figure 4(a). The illustration showcases the initial round of the search, where $\mathcal{B}_c$ contains only one item. For simplicity, let's assume that there are two search candidates being considered. From the figure, it is clear that the $d_1 = |P_1 - T_c| + |T_1 - P_c|$ is smaller compared to $d_2 = |P_2 - T_c| + |T_2 - P_c|$. Hence, candidate 1 is chosen in this case.

In Figure 4(b), we demonstrate the searched $\mathcal{B}_c$ with the hyper-parameters $B_{\text{min}} = B_{\text{train}} = 10,000$, $B_{\text{max}} = 30,000$, $T = 500$, and $N = 6$. The values in our searched $\mathcal{B}_c$ consist of $\{10,000, 17,500, 18,000, 19,000, 20,000, 25,000\}$. In this figure, we use a staggered arrangement of attention scores along the vertical axis to provide a clear demonstration. A highlighted area, enclosed within a parallelogram, vividly showcases how the waveforms of various bases interweave within the context. By imagining flattening this parallelogram, readers can better understand how the peaks and troughs of these waveforms align closely with each other in a practical scenario.

3.5 Inference Cost

In *Attention Buckets*, parallel processing of the context with different base values introduces additional memory overhead but does not affect the inference speed. All experiments described in this paper were executed on a single NVIDIA A100-80G GPU. In addition to being collated as a micro-batch on a single GPU, the parallel runs can be assigned to distributed across multiple GPUs, which means there is no constraint on the parallel number N as long as there is enough available memory.

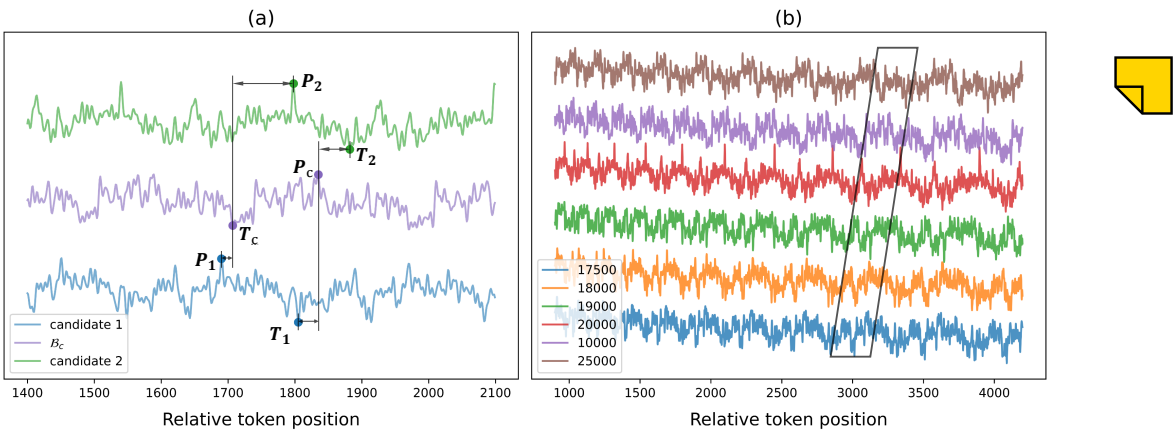

Figure 4: (a) RoPE Base Searching: we measure the distance from the candidate bases' attention peaks (or troughs) to the attention troughs (or peaks) corresponding to items in $\mathcal{B}_c$. For demonstration clarity, we illustrate only a partial context that corresponds to one waveform period. (b) The depiction of the attention score before softmax with $\mathcal{B}_c$ searched by hyperparameters detailed in §3.3. We use a parallelogram to showcase that the waveform peaks and troughs of various bases are compensated within the context.

4 Experiments

Benchmark Up to the time of this paper, Toolbench [4] stands as the largest benchmark for evaluating the tool use proficiency of large language models. It has extensive resources, including 3,451 tools, 16,464 APIs, 126,486 instances, and 469,585 API calls. All api calls in Toolbench are real and sampled from Rapid API.

The Toolbench evaluation consists of three distinct levels and three specific scenarios, each offering its own set of challenges. The three evaluation levels include **Inst.**: testing the model's response to new instructions for tools already covered in the training data; **Tool.**: measuring performance with unfamiliar tools within the same tool categories as those in the training dataset; and **Cat.**: examining the model's ability to handle tools from completely new categories not represented in the training data. The scenarios are **I1**: single-tool instructions, **I2**: multi-tool instructions within the same category, and **I3**: multi-tool instructions spanning across different collections. Due to specific details, there are only six combinations of levels and scenarios: I1-Inst., I1-Tool., I1-Cat., I2-Inst., I2-Cat., and I3-Inst. Each combination comprises 200 test queries, with the exception of I3-Institution, which includes 100 queries. For a more detailed introduction, readers are recommended to [4, §3.2].

Models and Evaluation Based on the training dataset in ToolBench, [4] fine-tuned a model named ToolLlama, building upon Llama-2-7B [10]. The authors compare it with advanced close-source LLMs, including ChatGPT [2], Claude-2, Text-Davinci-003, and GPT-4 [3]. We implement *Attention Buckets* to enhance the performance of ToolLlama. The $\mathcal{B}_c$ employed for this benchmark is the same as detailed in §3.3 and depicted in Figure 4(b). Following [4], we adopt multiple reasoning methods for each model, including ReACT [25], DFSDT [4, 26], and an API retriever [4] augmentation for reducing noise in tool selection (DFSDT-Retriever). We adopt the greedy decoding strategy.

Evaluation of these models is conducted using two metrics: pass rate and win rate. The pass rate accesses how many user queries are fulfilled. The win rate, determined by ChatGPT evaluates the superiority of the model's solutions compared to those provided by ChatGPT-ReACT.

Results and Analysis We present our experimental findings in Table 2. Our *Attention Buckets* enhances the scores of ToolLlama in almost every task level and scenario. Notably, when paired with the DFSDT-Retriever setup (in the table's final row), our approach not only matches but often surpasses GPT-4's performance levels. On average, *Attention Buckets* stand out, boasting the highest pass rate of 71.3% and win rate of 71.5%. To our knowledge, *Attention Buckets* set a new state-of-the-art (SOTA) result in this benchmark.

We also implement *Attention Buckets* with various reasoning methods, as detailed in the table's bottom three rows. Each method showed marked

Table 2: The tool use performance on ToolBench [4]. We highlight the leading results for each task with **bold fonts**, and denote the second-best performance with underlines. *Attention Buckets* augment the ToolLlama with only 7B parameters to outperform GPT-4 in both overall pass rate and win rate.

Model	Method	I1-Inst.		I1-Tool.		I1-Cat.		I2-Inst.		I2-Cat.		I3-Inst.		Avg	
		pass	win	pass	win	pass	win	pass	win	pass	win	pass	win	pass	win
ChatGPT	ReACT	41.5	-	44.0	-	44.5	-	42.5	-	46.5	-	22.0	-	40.2	-
	DFSDT	54.5	60.5	65.0	62.0	60.5	57.3	75.0	72.0	71.5	64.8	62.0	69.0	64.8	64.3
Claude-2	ReACT	5.5	31.0	3.5	27.8	5.5	33.8	6.0	35.0	6.0	31.5	14.0	47.5	6.8	34.4
	DFSDT	20.5	38.0	31.0	44.3	18.5	43.3	17.0	36.8	20.5	33.5	28.0	65.0	22.6	43.5
Text-Davinci-003	ReACT	12.0	28.5	20.0	35.3	20.0	31.0	8.5	29.8	14.5	29.8	24.0	45.0	16.5	33.2
	DFSDT	43.5	40.3	44.0	43.8	46.0	46.8	37.0	40.5	42.0	43.3	46.0	63.0	43.1	46.3
GPT4	ReACT	53.5	60.0	50.0	58.8	53.5	63.5	67.0	65.8	72.0	60.3	47.0	78.0	57.2	64.4
	DFSDT	60.0	**67.5**	**71.5**	**67.8**	**67.0**	66.5	79.5	73.3	**77.5**	63.3	**71.0**	84.0	71.1	70.4
ToolLlama	ReACT	25.0	45.0	29.0	42.0	33.0	47.5	30.5	50.8	31.5	41.8	25.0	55.0	29.0	47.0
	DFSDT	57.0	55.0	61.0	55.3	62.0	54.5	77.0	68.5	77.0	58.0	66.0	69.0	66.7	60.0
	DFSDT-Retriever	64.0	62.3	64.0	59.0	60.5	55.5	81.5	68.5	68.5	60.8	65.0	73.0	67.3	63.1
ToolLlama + *Attention Buckets*	ReACT	31.5	45.0	32.0	42.5	33.5	49.0	31.5	65.0	32.0	42.0	28.0	58.0	31.3	50.3
	DFSDT	66.5	**67.5**	61.5	62.0	62.0	65.5	78.0	71.5	73.0	**66.5**	67.0	82.0	68.0	69.2
	DFSDT-Retriever	**68.5**	65.0	70.0	65.5	65.0	**67.0**	**84.0**	**78.0**	71.0	64.5	69.0	**89.0**	**71.3**	**71.5**

improvements over their respective baselines, illustrating the versatility and compatibility of our approach. These results collectively indicate that *Attention Buckets* boosts ToolLlama's tool-use proficiency, a success we attribute to its enhanced context awareness.

These accomplishments lead us to argue that language models harbor many untapped potentials. By effectively leveraging these capabilities, LLMs could prove to be far more powerful than we thought. We hope our findings inspire further research into unlocking more fundamental abilities of LLMs.

5 Exploring Applications for Retrieval-Augmented Generation

Considering our proposed *Attention Buckets* enhances the model's context awareness, it should be effective in other tasks demanding high contextual information utilization. This section explores the effectiveness and generality of *Attention Buckets* through an open-domain question answering (ODQA) task. We choose this task for two reasons:

(1) Open-domain generation [27, 28, 29] represents a long-standing challenge in natural language processing. Many current open-domain QA methods employ a Retrieval-Augmented Generation (RAG) paradigm [30, 31, 32, 33], where a retriever [34, 35] gathers relevant documents, followed by a generative reader [36, 37] to find answers, demanding high context awareness.

(2) Given that Toolbench requires GPT-4 and ChatGPT for API calls in benchmark tests and evaluations, conducting analysis about our method on it is prohibitively costly for us. In contrast, ODQA does not require API calls, and its evaluation process is both objective and convenient. We aim to investigate whether our identified $\mathcal{B}_c$ is close to optimal in this task.

Table 3: Accuracy on NQ and WebQ (10 documents).

Method	NQ	WebQA
FiD-XL (3B)	50.1	50.8
Llama-2 (7B)	48.5	51.7
+ *Attention Buckets*	**50.3** (1.8↑)	**53.1** (1.4↑)

5.1 Experiments on ODQA

We conduct experiments on two popular ODQA benchmarks NaturalQuestion (NQ [38]) and WebQA [39], with 3,610 and 2,032 test samples, respectively. We assess the models based on their accuracy in providing answers. An answer is considered correct if it contains one of the acceptable answers. In our experiments, we employed 10 documents as context and utilized the DPR [40] as the retriever, which is a supervised dense retrieval model trained on above datasets.

We use Llama-2-7B [10] as our backbone model and compare it with FiD-XL [36], the exclusive ODQA model trained on multiple ODQA benchmarks, including NQ and WebQ. When augmented with *Attention Buckets*, Llama-2-7B demonstrated superior performance over the dedicated QA model FiD-XL. The results are shown in Table 3.

Table 4: Searched bases by variant search algorithms.

	(N, T)	Searched Results ($\times 1e^4$)
$\mathcal{B}_{A.S.1}$	(7, -)	{1.00, 1.30, 1.60, 1.90, 2.20, 2.50, 2.80}
$\mathcal{B}_{A.S.2}$	(6, -)	{1.00, 1.40, 1.80, 2.20, 2.60, 3.00}
$\mathcal{B}_{c1}$	(7, 1,000)	{1.00, 1.7, 1.80, 1.90, 2.00, 2.30, 2.50}
$\mathcal{B}_{c2}$	(6, 500)	{1.00, 1.75, 1.80, 1.90, 2.00, 2.50}
$\mathcal{B}_{c3}$	(7, 500)	{1.00, 1.75, 1.80, 1.90, 2.00, 2.25, 2.50}

5.2 The ablation of components in $\mathcal{B}_c$

Considering the cost issues mentioned above, we conduct an ablation study on the components of $\mathcal{B}_c$ using the NQ dataset, instead of Toolbench, to assess the effectiveness of our search algorithm. We investigated the impact of varying N, the size of $\mathcal{B}_c$, and T, the search stride (granularity). We created three variations of $\mathcal{B}_c$ with different N and T, with each variation acting as a counterpart to the others. The specific values of N and T, along with their respective search outcomes, are presented in Table 4. Furthermore, we compare these searched $\mathcal{B}_c$ with (1) $\mathcal{B}_{A.S.}$ whose base values form arithmetic sequences with common differences of 3,000 and 4,000, respectively; (2) each individual element within $\mathcal{B}_c$.

We compare various variations against one another and illustrate the differences in accuracy in Figure 5. A higher accuracy discrepancy indicates that a particular variation is performing better. Our findings reveal that the combined bases in searched $\mathcal{B}_c$ yield better accuracy than any single base working independently and demonstrates superior performance compared to $\mathcal{B}_{A.S.}$. And a proper choice of N and T may contribute to increased accuracy. For more experiments and detailed results, please refer to the appendix E.

6 Related Work: LLM-Based Tool Use

Taking advantage of large language models to use external tools is an emerging research topic [41, 42]. Researchers have explored a variety of tools, including calculators for mathematical computations [5, 43], specialized expert models [8], and web API calls [4, 7]. In these studies, LLMs interact with users by analyzing their intents and needs. A tool-retriever is utilized to source relevant tools, typically in the format of documents containing details like tool names, examples of use, descriptions of functions, and arguments for those functions. The LLM processes these documents to choose the suitable tool, inputs the necessary arguments, and relays the tool's output back to the users. Many

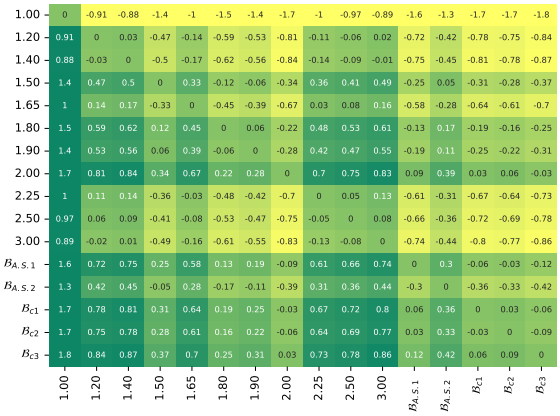

Figure 5: The accuracy discrepancy between (1) single-base, (2) $\mathcal{B}_{A.S.}$ and (3) $\mathcal{B}_c$ searched by Algorithm 1. For each variant (each row), the higher the accuracy discrepancy it attains with respect to a column, the more effective it is.

studies [44, 7, 4] have found that large language models, including GPT-4 [3], often exhibit hallucinations, such as inventing non-existent functions and arguments, or failing to adapt to changes in interactive environments. These highlight the critical need for enhancing these models' context awareness.

Current research heavily focuses on integrating multiple reasoning pathways to address errors caused by insufficient contextual comprehension. These techniques include ReAct [25], DFS tree search [4], self-consistency [45], etc. In contrast, our approach focuses on a fundamental solution: enhancing contextual awareness. It is both orthogonal to and stackable with those reasoning methods.

7 Conclusion

In this paper, we delve into the waveform patterns observed in attention scores and propose a hypothesis which suggests that fluctuations in the upper bound of attention score could potentially affect the model's context awareness, particularly in relation to the position of crucial information within the context. We verified this hypothesis through a preliminary study and propose *Attention Buckets*, an inference augmentation method designed to enhance the model's context awareness. This method combines various attention patterns, which are controlled by different RoPE bases. Our approach has achieved state-of-the-art (SOTA) performance on the current largest tool-use benchmark while showing the potential applicability to a wider range of RAG tasks.

Acknowledgement

This work was supported by Alibaba Group through Alibaba Innovative Research Program.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] OpenAI. OpenAI: Introducing ChatGPT, 2022.

[3] OpenAI. Gpt-4 technical report, 2023.

[4] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023.

[5] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023.

[6] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning, 2023.

[7] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis, 2023.

[8] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023.

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

[10] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[12] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2022.

[13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien

Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[14] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023.

[15] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. Baichuan 2: Open large-scale language models, 2023.

[16] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023.

[17] Yifei Gao, Lei Wang, Jun Fang, Longhua Hu, and Jun Cheng. Empower your model with longer and better context comprehension, 2023.

[18] Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation, 2023.

[19] Ang Lv, Xu Tan, Peiling Lu, Wei Ye, Shikun Zhang, Jiang Bian, and Rui Yan. Getmusic: Generating any music tracks with a unified representation and diffusion framework, 2023.

[20] Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. Pose: Efficient context window extension of llms via positional skip-wise training, 2023.

[21] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models, 2023.

[22] Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. Are we falling in a middle-intelligence trap? an analysis and mitigation of the reversal curse, 2023.

[23] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can open-source llms truly promise on context length?, June 2023.

[24] Paul J. Leach, Rich Salz, and Michael H. Mealling. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, jul 2005.

[25] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[26] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, june 2023. *arXiv preprint arXiv:2303.11366*, 2023.

[27] Ang Lv, Jinpeng Li, Shufang Xie, and Rui Yan. Envisioning future from the past: Hierarchical duality learning for multi-turn dialogue generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors,

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7382–7394, Toronto, Canada, July 2023. Association for Computational Linguistics.

[28] Ang Lv, Jinpeng Li, Yuhan Chen, Gao Xing, Ji Zhang, and Rui Yan. DialoGPS: Dialogue path sampling in continuous semantic space for data augmentation in multi-turn conversations. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1267–1280, Toronto, Canada, July 2023. Association for Computational Linguistics.

[29] Shufang Xie, Ang Lv, Yingce Xia, Lijun Wu, Tao Qin, Tie-Yan Liu, and Rui Yan. Target-side input augmentation for sequence to sequence generation. In *International Conference on Learning Representations*, 2022.

[30] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[31] Ang Lv, Xu Tan, Tao Qin, Tie-Yan Liu, and Rui Yan. Re-creation of creations: A new paradigm for lyric-to-melody generation. *arXiv e-prints*, pages arXiv–2208, 2022.

[32] Omar Khattab, Christopher Potts, and Matei Zaharia. Relevance-guided supervision for openqa with colbert. *Transactions of the association for computational linguistics*, 9:929–944, 2021.

[33] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[34] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*, 2020.

[35] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[36] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering, 2021.

[37] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*, 2022.

[38] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[39] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[40] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[41] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey, 2023.

[42] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng,

Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models, 2023.

[43] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings, 2023.

[44] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2023.

[45] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[46] Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Label words are anchors: An information flow perspective for understanding in-context learning, 2023.

A The Upper Bound of Attention Score Before Softmax

Formally, the inner-product between query vectors at position m and key vectors at position n within Transformer models utilizing RoPE can be formulated as:

$$
\begin{aligned}
&q_m = R_{\theta,m} W_q x_m, \quad k_n = R_{\theta,n} W_k x_n, \\
&q_m \cdot k_n = (R_{\theta,m} q)^T (R_{\theta,n} k) = \\
&\mathrm{Re}\left[\sum_{j=0}^{d/2-1} q[2j:2j+1] k^*[2j:2j+1] e^{i(m-n)\theta_j}\right] \\
&= \sum_{j=0}^{d/2-1} (q_{2j} \cdot k_{2j} + q_{2j+1} \cdot k_{2j+1}) \cos((m-n)\theta_j) \\
&+ (q_{2j} \cdot k_{2j+1} - q_{2j+1} \cdot k_{2j}) \sin((m-n)\theta_j),
\end{aligned} \tag{6}
$$

where x is the d-dimensional input of the current Transformer layer, and $\theta_j = B^{-\frac{2j}{d}}$.

When q_m and k_n are identical, the inner-product reaches its maximum value. For computational simplicity, we assume them as all-one vectors to derive the upper bound (UB) of the inner-product (attention score before softmax):

$$
UB = \sum_{j=0}^{d/2-1} 2\cos((m-n)\theta_j) \geq q_m \cdot k_n. \tag{7}
$$

Figure 1(a) illustrates the visualization of UB with base = 10,000. Figure 4(b) results from varying base with values from our searched set $\mathcal{B}_c$. These figures demonstrate the horizontal axis as the relative position between k_n and q_m.

B Locating Peaks and Troughs in an Attention Waveform

```python
d = 128 # dimension of Q or K vectors in Llama.
MAX_CONTEXT_LENGTH = 4096 # the maximum
    pre-trained context length.

# Calculate the upper bound of attention score
    before softmax.
def qmkn(base, pos_mn):
    # base: RoPE base.
    # pos_mn: relative token position for qm and
        kn.
    # return: the upper bound of attention score
        between qm and kn.
    score = 0.0
    for i in range(0, d/2):
        score += 2 * np.cos((pos_mn) *
            np.power(base, (-2*i/d))
    return score


# Find n peak positions
# within MAX_CONTEXT_LENGTH.
def fp(base, n, period):
    # base: RoPE base.
    # n: expected number of searched peaks.
    # period: init approximate period.
    # return: P, a list contains peak positions.
    scores = [qmkn(pos_mn, base) for pos_mn in \
        range(MAX_CONTEXT_LENGTH)]
    P = []
    start = 0
    while len(P) < n:
        p_max = np.argmax(scores[start: start \
                + period]).index + start
        P.append(p_max)
        start = p_max

        # The attention waveform exhibts
        # irregualr period throughout the
        # context, with each successive
        # period being approximately 1.5
        # times longer than the previous one.
        period *= 1.5
    return P


# Find n trough positions
# within MAX_CONTEXT_LENGTH.
def ft(base, n, period):
    # base: RoPE base.
    # n: expected number of searched troughs.
    # period: init approximate period.
    # return: T, a list contains trough
        positions.
    scores = [qmkn(pos_mn, base) for pos_mn in \
        range(MAX_CONTEXT_LENGTH)]
    T = []
    start = 0
    while len(T) < n:
        t_min = np.argmin(scores[start: start +\
            period]).index + start
        T.append(t_min)
        start = t_min
        period = 1.5 * period
    return T
```

C Supplement to The In-Context Retrieval Experiment

There are several critical settings to make our experiments fair and convincing:

- **Position Anchoring Based on Last Token**: We anchor the position of each target key-value pair at its last token. This approach is based on the findings of [46] that the sentence semantic is gathered to the last token.
- **Precise Positioning of Key-Value Pairs**: To accurately place key-value pairs at specific positions within the context, we insert padding tokens after the key-value JSON data and prior to the query.
- **Consistent Prompt Lengths Across Rounds**: Since our experiments involve comparing the accuracy between two rounds, it is essential to mitigate any potential biases arising from varying context lengths. To achieve this, we maintain the consistency in the context length across two rounds by inserting padding tokens at the beginning of the input.

D The Impact of Base Value Smaller Than That Used In Training

Considering Eq. 6, where the position information is integrated using $d/2$ sinusoidal functions with the frequency $\theta_j = B^{-\frac{2j}{d}}$, for $j \in [0, \frac{d}{2}]$. If a smaller value of B' is employed, compared to the pre-trained B, the frequency of these sinusoidal functions will be higher, resulting in a reduced period. In this case, given the maximum pre-trained context length, the final few tokens could correspond to positions within a period of the sinusoidal functions that are not encountered during training. These positions would be considered out-of-distribution for the model. We recommend that readers interested in a more in-depth analysis of this context "scaling law" refer to [18].

E More Results for Ablation on $\mathcal{B}$

In Table 5, we provide additional results on the accuracy of the $\mathcal{B}$ variations, complementing the information presented in Figure 5. The bottom 4 rows indicate the results of our searched $\mathcal{B}_c$ with varied T and N.

Table 5: Accuracy of the $\mathcal{B}$ variations on NQ (10 documents).

$\mathcal{B}$ ($\times 1e^4$)	Acc.
{1.00}	48.56
{1.20}	49.47
{1.30}	49.67
{1.40}	49.45
{1.50}	48.94
{1.65}	49.61
{1.70}	50.11
{1.80}	50.01
{1.90}	50.00
{2.00}	50.28
{2.10}	48.56
{2.25}	49.58
{2.30}	49.97
{2.45}	49.89
{2.50}	49.53
{2.60}	49.67
{2.70}	49.52
{2.80}	49.47
{2.90}	49.31
{3.00}	49.45
{1.00, 1.30, 1.60, 1.90, 2.20, 2.50, 2.80}	50.19
{1.00, 1.40, 1.80, 2.20, 2.60, 3.00}	49.89
{1.00, 1.30, 1.60, 1.90, 2.20, 2.50}	50.14
{1.00, 1.80, 2.00, 2.20, 2.40, 2.60, 3.00}	49.72
{1.00, 1.30, 1.60, 1.90, 2.20, 2.50, 2.80}	48.56
{1.00, 1.75, 1.80, 1.90, 2.00, 2.50}	50.22
{1.00, 1.77, 1.78, 1.90, 2.02, 2.47, 2.48}	50.25
{1.00, 1.70, 1.80, 1.90, 2.00, 2.30, 2.50}	50.25
{1.00, 1.75, 1.80, 1.90, 2.00, 2.25, 2.50}	**50.31**