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†] **Rui Yan**[1,3†]
[1]Gaoling School of Artificial Intelligence, Renmin University of China [2]Alibaba Group
[3]Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education
{yuhanchen, anglv, chen.changyu, ruiyan}@ruc.edu.cn
{ting-en.lte, shengxiu.wyc, f.huang, shuide.lyb}@alibaba-inc.com

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11160–11174
August 11-16, 2024

Motivation

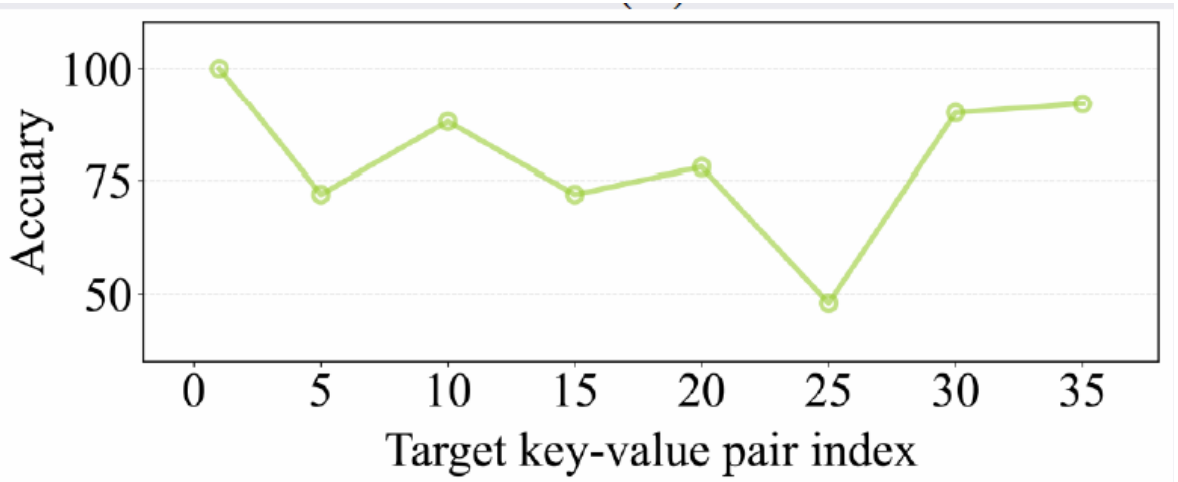
- In our practical experience, **we observed that LLMs exhibit varying levels of awareness concerning different positions within the context.** For instance, LLMs may overlook certain tools within the context, resulting in a failed call; however, by altering the position of these tools, the task can be successfully executed. Such variations significantly affect the performance of LLMs in tool-use.
- When LLMs are presented with **multiple key-value pairs** and instructed to **retrieve the value associated with a specific key**, the index of the queried target key results in significant fluctuations in accuracy.

Input Context

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

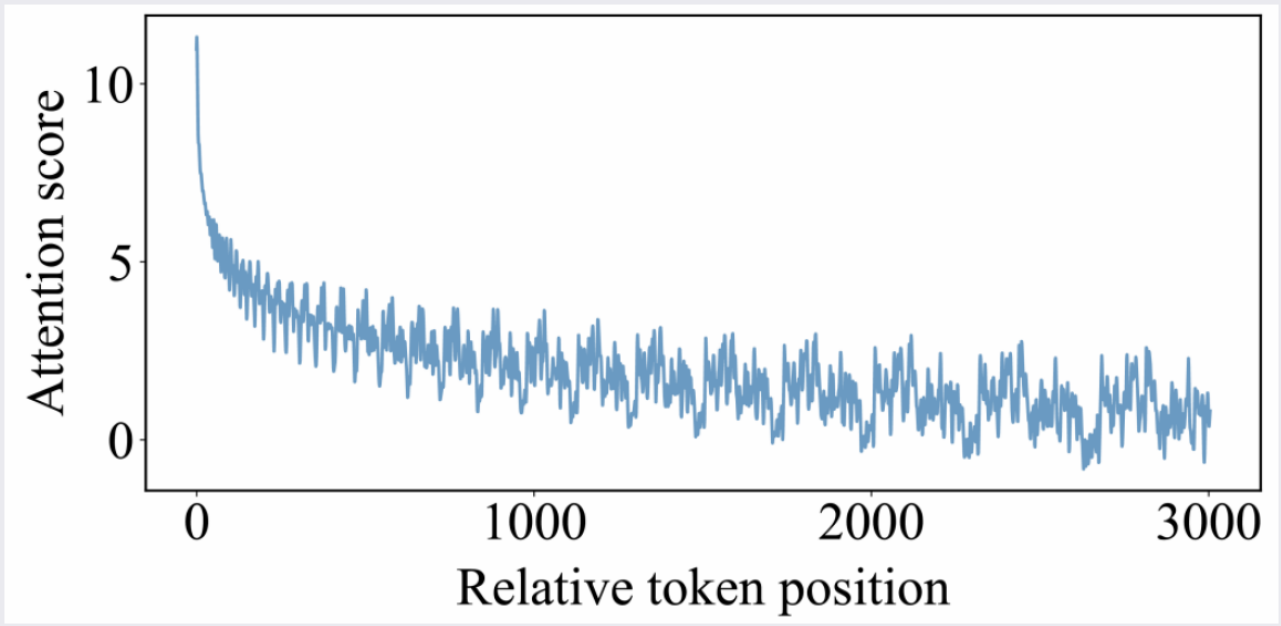
JSON data:
{"1e0029ce- … -f1ed9642d893": "3d678cff- … -2950de83f31c",
"da448545- … -bcb1d03a2254": "89e4a63e- … -c7c96e4c8b9d",
"**a5d4c9ee- … -54ea28f8bf21**": "dc9708f9- … -c6aeb3606222",
…
"0194ec2b- … -728bab87b4f7": "5daf5bed- … -9c134fb7745a",
"2ac4aebd- … -76a333d48489": "3945b582- … -3dbb44d3162b"}

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



Analysis

- We go beyond the superficial fluctuations previously observed and identify that **these position-related performance differences are closely associated with the model's fluctuating attention allocation**.
- We demonstrate that if the position of the crucial information coincides with a trough in the attention waveform, the model may overlook it, leading to decreased accuracy.



- **The attention score exhibits fluctuations when retrieving the same token from the context at specific relative positions**, a phenomenon we refer to as the "attention waveform."

Analysis

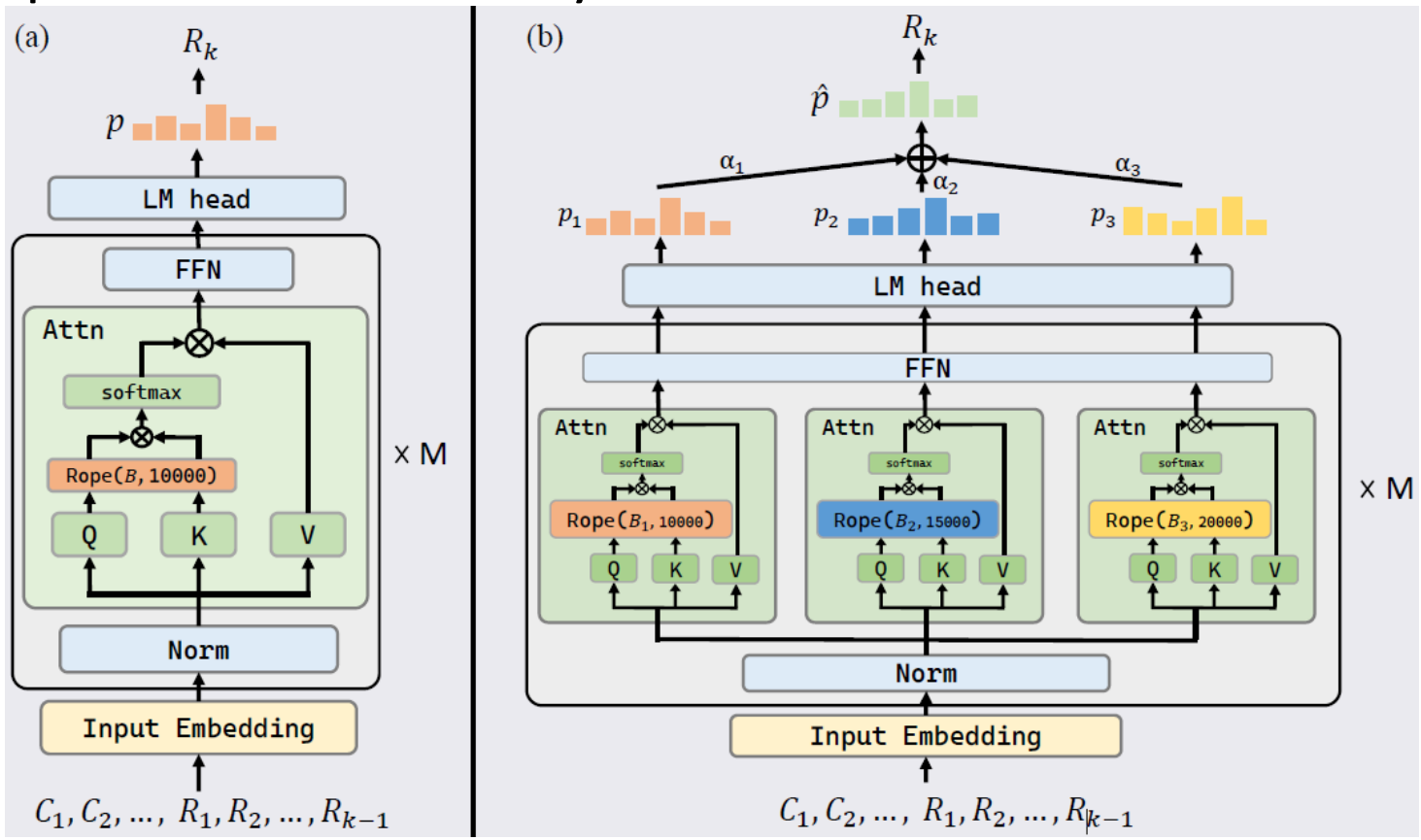
- For each base value, **we calculate the corresponding waveform of attention score and identify the positions of the peaks and troughs**. Each test sample undergoes two evaluation rounds: In the first round, we position the target key-value pair at the attention peak nearest to the exact middle of the context. In the second round, we move the target pair to the nearest attention trough.
- Our experimental findings reveal a performance trend associated with varying RoPE base values: **we observe an initial rise followed by a subsequent fall**.

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	96.6	96.2	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

- Notably, **placing a key-value pair at the peak of the attention waveform consistently yields better outcomes than positioning it at the trough**.

Method

- Given an input context 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 B_c**, resulting in N corresponding predicted distribution p over the vocabulary V.



Method

- **Our selection of Bc 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 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 R. We quantify the model's confidence on prediction a_j as:

$$\alpha_j^{'} = \max_{v \in V} p(\mathcal{R}_k = v | \mathcal{C}, B_j, \mathcal{R}_{1:k-1}),$$

$$\alpha_j = \frac{e^{\alpha_j^{'}}}{\sum_{i=1}^{N} e^{\alpha_i^{'}}}.$$

- Next, we compute a weighted sum of each run's output distribution p_j to derive the final predicted distribution $\hat{p}$:

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

Method

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

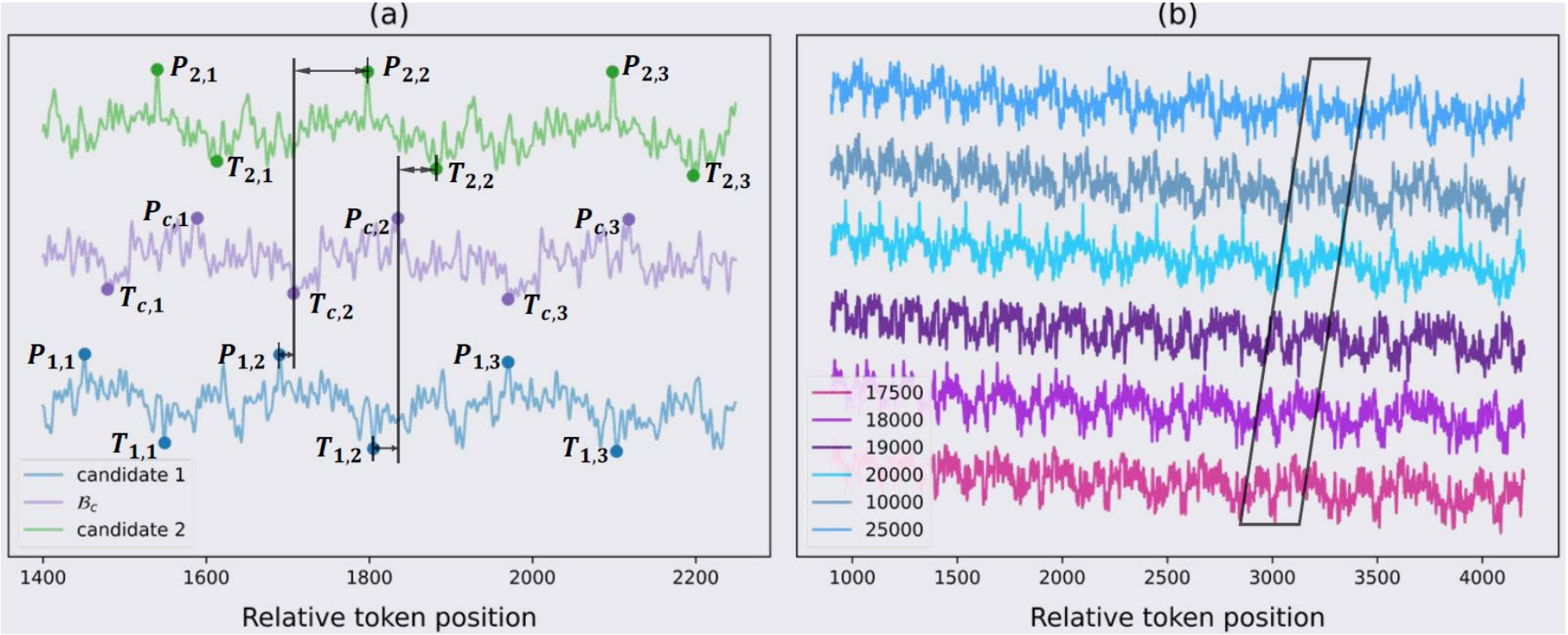
- Our goal is to **develop strategies ensuring that the attention waveform troughs of any given base overlap with peaks from different bases**, and vice versa. Firstly, we define a discrete base search space, denoted as:

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

- We set B_{min} equal to B_{train}, the base used during model pre-training.
- In every round, we first identify the peaks and troughs within the waveform associated with each base in B_s and B_c. The selection of a candidate is determined by **measuring the distance between the position of the i-th peak (and trough) for a candidate base and that of the i-th trough (and peak) for bases within set B_c**.

Method

- we measure the distance from the candidate bases' attention peaks (or troughs) to the attention troughs (or peaks) corresponding to items in B_c.
- These peak points effectively divide this edge into several equal segments. This suggests that **our searched Bc possesses waveforms that are evenly and densely distributed**, minimizing the likelihood of a position being overlooked.



Experiments

- **Benchmark: ToolBench.**

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.**

- **Model: ToolLLaMA.**
- **Reasoning Methods: ReAct, DFSDT.**
- **Metrics:**
    - **Passrate**: access how many user queries are fulfilled.
    - **Winrate:** evaluate the superiority of the model's solutions compared to those provided by ChatGPT-ReACT.

Experiments

	Original		+AB$_{once}$		+ASort		+USC		+AB	
	pass	win	pass	win	pass	win	pass	win	pass	win
I1-Inst.	64.0	62.3	52.0	37.0	66.0	63.0	66.0	63.0	**68.5**	**65.0**
I1-Cat.	64.0	59.0	40.5	31.0	67.0	62.0	65.5	61.5	**70.0**	**65.5**
I1-Tool.	60.5	55.5	47.0	34.5	59.5	58.5	61.0	58.5	**65.0**	**67.0**
I2-Inst.	81.5	68.5	70.5	65.0	80.0	68.5	80.0	73.0	**84.0**	**78.0**
I2-Cat.	68.5	60.8	65.0	58.0	68.5	60.3	70.0	61.5	**71.0**	**64.5**
I3-Inst.	65.0	73.0	61.0	52.0	66.0	75.0	66.0	78.0	**69.0**	**89.0**
Avg	67.3	63.1	56.0	45.3	67.8	64.6	68.1	65.9	**71.3**	**71.5**

- **Attention Bucketsonce: computes the average of N attention waveforms from individual bases and then encodes the positional information using this averaged waveform.** This technique utilizes only a single inference process, thereby avoiding any additional memory cost.

Experiments

- **Attention Sorting**. The authors first segmented the context and calculated the average attention of each segment. **By rearranging segments in context based on sorted attention scores(with the highest attention segment placed last)**, they generate the answer using the newly sorted context. This approach does not require extra memory;
- **Universal Self-Consistency.** USC is a universal self-consistency algorithm that supports free-format outputs. The LLM first generates N responses. Subsequently, the LLM is tasked with selecting the response that **exhibits the highest degree of consistency, employing a specific prompt.** The memory cost of this method is roughly equivalent to that of our Attention Buckets.

Experiments

- **RAG task:** open-domain question answering (ODQA).
- **BenchMarks:** NaturalQuestion, WebQA.

  3,610 and 2,032 test samples.
- **Retriever:** DPR.
- **Model:** LlaMA-2 7B.
- **Metrics:** Accuracy.

Method	NQ	WebQA
FiD-XL (3B)	50.1	50.8
Llama-2 (7B)	48.5	51.7
+ ASort	48.9	52.1
+ USC	47.6	51.7
+ *Attention Buckets*	**50.3** (1.8↑)	**53.1** (1.4↑)

Appendix-attention waveform

- 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:

$$R_{\theta_i,m} = \begin{bmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{bmatrix}$$

- 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 positionmand 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):

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