Language Models Speak in Sentences: Sentence Structure Improves Language Model's Reasoning Capability

Anonymous ACL submission

Abstract

We propose a new method to improve large language models' (LLMs) performance by incorporating the sentence structure knowledge into the model. Based on the intuitive assumption that a complete sentence is the basic unit of thinking and reasoning for human beings, we test it for LLMs by explicitly inserting special segment tokens to the positions within the input sequences where sentence boundaries are detected, which achieves better performance in complex reasoning tasks by significant margins. Two approaches for incorporating sentence structure knowledge are experimented: In-context learning (ICL) on instruction-tuned models (Llama3-8B-Instruct and Qwen2-7B-Instruct) and supervised fine-tuning (SFT) on a base model (Llama3-8B fine-tuned with TULU3), and evaluated in highly reasoning-intensive tasks (e.g., math), both show positive results. Our findings indicate that similar to human reasoning, structured sentences can effectively facilitate LLM reasoning performance; integrating linguistically motivated priors, such as sentence boundaries, is a promising future direction for developing simple-yet-effective prompting techniques.

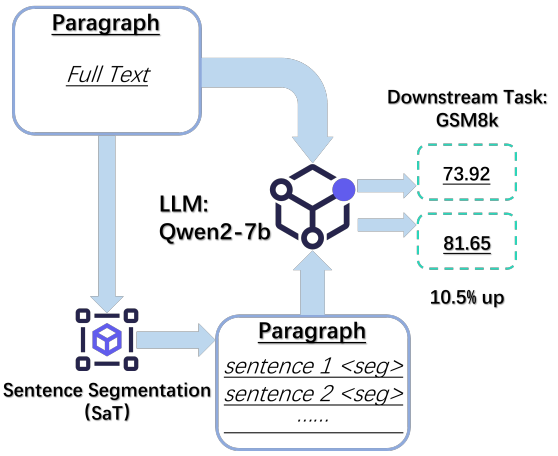


Figure 1: **Summary of this study**: **Downstream performance can be improved through sentence segmentation.** Given a prompt, we use a sentence segmentation model, SaT, to segment it into sentences, and then reconstruct it back with segmentation token inserted (at the end of each sentence). We observe significant performance gains after applying sentence segmentation to LLMs, e.g., the downstream performance of segmented paragraph outperforms the original paragraph by approximately 10% on the GSM8k when tested with the Qwen2-7B-Instruct model.

1 Introduction

As the foundation Large Language Model (LLM) training workflow was proposed by GPT series (Brown et al., 2020; Ouyang et al., 2022), there are many works trying to enhance the performance of language model through different approaches. Since the proposal of different scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; DeepSeek-AI et al., 2024), the most common approaches to improve language models' performance at training time are to scale up the model size (Chowdhery et al., 2022), or alternatively, to scale up the training data (Touvron et al., 2023). However, scaling up is heavily dependent on computational resources. Most researchers have to explore other ways to enhance the model performance at inference time, under the constraint of limited computational budget.

Several works like Wei et al. (2022) and Yao et al. (2023) are trying to improve performance by employing thought-prompting methods to scale the thinking time during inference. Building on the concept of test-time-scaling, researchers have explored various techniques to enhance performance, including self-verification methods like Renze and Guven (2024), and reinforcement learning (RL) methods such as Monte-Carlo Tree Search (MCTS) (Qi et al., 2024; Zhang et al., 2024). Similarly, there are also some other RL-based approaches have been proposed, such as reinforcement finetuning (RFT), including DeepSeek R1 and Kimi K1.5

(DeepSeek-AI et al., 2025; Team et al., 2025).

These methods are either time-consuming or data-consuming. Therefore, it leads to the research question: *Can we get a free lunch to improve LLM performance by leveraging some sentence-level priors in human language?*

We make the following assumptions:

Assumptions

1. The minimum unit of human thought is the sentence.
2. Sentence structure can facilitate the process of thinking and reasoning.

Based on these assumptions, we propose a method that uses sentence segmentation to improve LLM performance. Figure 1 summarizes our workflow on how sentence segmentation is incorporated into the overall LLM inference procedure. We summarize our contributions as follows:

1. We propose a method that explicitly incorporates the prior knowledge of sentence structure into LLMs by inserting segmentation tokens at sentence boundaries, enabling the model to effectively recognize and utilize sentence structure information.
2. Experimental results in Section 4 show consistent performance improvements on both In-Context Learning (ICL) and Supervised Fine-Tuning (SFT) approaches, confirming that the sentence structure representation is beneficial for LLMs' performance.
3. Ablations on ICL in Section 4.2 demonstrate that, a more structured segment token is better for downstream performance, and segmentation by sentence structure is optimal for performance compared to n-token segmentation.
4. An explanation analysis in Section 4.4 shows why the sentence segmentation approach could work.
5. A potential contribution of this work would be a by-product that brings a useful tool to any work on **LLM+sentence**, such as sentence-level reward model mentioned in Section 2 and others like MCTS in reasoning (treating sentence as state & action).

2 Related Work

Training Language Model with Pause Token Goyal et al. (2024) suggests that a pause before an LLM answers the question is beneficial for the performance: it allows LLM to think more about the question through the "**<pause>**" tokens. They tested the idea during both pretraining stage and finetuning stage, and conducted experiments using SQuAD (Rajpurkar et al., 2016), CommonSense QA (Talmor et al., 2019), and GSM8k (Cobbe et al., 2021) datasets for a 1B model. They observed 18% improvements in SQuAD, 8% in CommonSense QA, and 1% in GSM8k, demonstrating that pause tokens can help enhance the model's reasoning capabilities.

Sentence Segmentation Sentence segmentation is a fundamental natural language processing (NLP) task that aims to segment text into sentences. This process plays a crucial role in many NLP systems. Frohmann et al. (2024) proposed a model called Segment Any Text (SaT) for universal text segmentation, which achieves state-of-the-art performance on sentence segmentation tasks.

Sentence Level Prediction Ippolito et al. (2020) proposed a new approach to solve the fluency problem in story generation and other creative writing tasks. They suggest using sentence prediction rather than token prediction. By using a pretrained BERT model to generate sentence embedding, they trained an MLP model to select the most suitable sentence from a limit yet large pool of candidate sentences. However, this approach is not suitable for open-ended text generation, which makes it far from real-world applications.

Sentence Level Reward Model Qiu et al. (2025) suggests that sentence segmentation can be used for reward modeling, thereby introducing a new framework named sentence-level RM. The results show that sentence-level reward can outperform token-level RM and response-level RM (e.g.: outperforms response-level RM 2.7% on RewardBench (Lambert et al., 2024)). It shows that sentence structure is benefit for reward modeling.

3 Encoding Sentence Structure into Language Model

3.1 Sentence Segmentation

The mainstream approaches to sentence segmentation fall into these categories: rule-based methods (Sadvilkar and Neumann, 2020), supervised methods (Frohmann et al., 2024; Wicks and Post, 2021), and unsupervised methods (Loper and Bird,

2002). Among these methods, supervised methods are currently the most accurate approach, widely integrated into various natural language processing toolkits. While LLMs are also capable for the sentence segmentation task using prompting method (Ouyang et al., 2022), the purpose of testing LLMs on this task is to demonstrate their general capabilities, rather than to design a tool for sentence segmentation task. Supervised methods usually use BERT-based model (Devlin et al., 2019) for supervise sentence segmentation. Minixhofer et al. (2023) defined the objective of sentence segmentation as identifying characters that can be followed by a delimiter, which can be formula as follows:

$$L_\theta = \sum_i \log P(y_i|x_1, x_2, \ldots, x_n, \theta)$$
$$\text{where } y_i = \begin{cases} 1 & x_{i+1} \text{ is sentence boundary} \\ 0 & \text{else} \end{cases} \quad (1)$$

It shows that the primary objective of traditional sentence segmentation is to predict sentence boundaries (e.g.: "**n**").

3.2 Inference with Sentence Segmentation

Figure 2 illustrates how sentence segmentation is applied in the language modeling task. We reformulate the tradition sentence segmentation task by introducing a segment token (represented as "**<seg>**"), and insert it at the end of each sentence.

Figure 2: As an example, the paragraph start with "**<bos>**", end with "**<eos>**". We insert segmentation tokens "**<seg>**" after every sentences.

This explicitly incorporates the sentence structure into the model. By treating "**<seg>**" as an input token, the original language modeling task could be extended, which requires the LLM to predict the segment token at sentence boundaries, allowing the model to learn the sentence structure. According to the previous assumption, downstream performance should be improved when segment tokens are added.

3.3 In-Context Learning

We propose an In-Context Learning (ICL) approach for learning sentence structure using segment tokens. The ICL approach could learn the structure from analogy according to Dong et al. (2024). It is formulated as:

$$x_i = argmax\ P(x_i|S, x_1, x_2 ... x_{i-1}, \theta) \quad (2)$$

where S represents a sequence of all previous texts, and x denotes the tokens in the generating sentence. S is defined as:

$$S = [s_1, x_{seg}, s_2, x_{seg}, ..., s_n, x_{seg}] \quad (3)$$

where s_i is a sentence, and x_{seg} is the segment token.

By repeatedly inserting segmentation tokens after each sentence, LLMs can infer sentence structure from in-context information, and learn to output segmentation tokens at sentence boundaries. Our observations show that most of the modern published LLMs are able to successfully identify sentence structures and insert segmentation tokens appropriately. The subsequent experiment in Section 4 demonstrates that the downstream performance of LLMs improves with this ICL-based structural learning.

3.4 Supervised Finetuning

In contrast to ICL, another approach uses supervised fine-tuning (SFT) to learn the sentence structure. It integrates sentence segmentation as structural knowledge into the LLM's parameters instead of relying on context only. We formulate the problem in SFT approach the same as Equation (2). For finetuning, we use the normal form of SFT:

$$L_\theta = \sum_{i=k}^{N} \log P(x_i|x_1, x_2, ..., x_{i-1}, \theta) \quad (4)$$

where x denotes all tokens in the input text (including segmentation tokens), N is the length of the whole text, and k represents the length of the input prompt. Compared to eq. (1), which predicts the segmentation token, the target of sentence segmentation here is included in the next token prediction task by predicting the segment token.

While ICL relies on sufficient contextual knowledge (i.e., requiring long context), SFT does not require a context. It can automatically incorporate segment tokens into model outputs. This approach is more suitable for zero-shot scenarios, which are better aligned with LLM's real-world applications.

In our SFT experiments, we treat the segment token directly as a special token inside the tokenizer. The token is added to the vocabulary; thus it introduces new **embedding** and **lm_head** weights.

4 Experiments

4.1 Settings

Our experiments aim to highlight differences before and after applying sentence segmentation in reasoning-related tasks. The hypothesis is that sentence segmentation can enhance the downstream task performance, especially on tasks that require strong reasoning abilities.

Dataset For the ICL experiments, we selected datasets that satisfy both criteria: having a long enough context for learning, and evaluating reasoning abilities. We choose two math datasets, GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b), and a reading comprehension dataset, DROP (Dua et al., 2019). We also choose MMLU (Hendrycks et al., 2021a) to test our method's performance in language understandingby output the answer with CoT prompt and calculate the exact match (EM) score (Clémentine Fourrier, 2023). We apply 8-shot CoT for GSM8k, 4-shot CoT for MMLU and MATH, and 3-shot CoT for DROP, reporting the EM score.

For SFT experiments, we additionally introduced three datasets, MMLU-Pro (Wang et al., 2024), GPQA (Rein et al., 2024) and HumanEval (Chen et al., 2021). MMLU-Pro is a harder version of MMLU dataset. GPQA is an extremely difficult QA dataset designed by domain experts, which requires knowledge, understanding, and reasoning. HumanEval is a coding dataset for Python coding question. It is similar to the math problem that also evaluates reasoning ability. GPQA and HumanEval do not have a long enough context (0-shot by default), so they are not included in ICL experiments. The settings of GSM8k, MATH and DROP are the same as ICL, while we apply 0-shot CoT for MMLU and GPQA here. HumanEval is also in 0-shot, and MMLU-Pro is tested in 5-shot CoT. We report the EM score for each task except HumanEval, in which we report the pass@1 score.

For SFT dataset, we use a subset from TULU3 SFT dataset (Lambert et al., 2025) that removes safety & non-compliance subset, multilingual subset, and TableGPT subset.

Model We conduct ICL experiments on the LLaMA3-8b-Instruct model (Grattafiori et al., 2024) and Qwen2-7b-Instruct model (Yang et al., 2024), and SFT experiments on the LLaMA3-8b-Base model (Grattafiori et al., 2024). We performed a full-parameter SFT on 8×L40 GPUs.

	MMLU	GSM8k	MATH	DROP
Llama3-8B	62.89	75.51	**32.60**	46.39
Llama3-8B-seg	**67.28**	**78.01**	32.26	**53.16**
	+4.39%↑	+2.50% ↑	-0.34% ↓	+6.77% ↑
Qwen2-7B	64.43	73.92	53.33	38.14
Qwen2-7B-seg	**69.96**	**81.65**	**54.30**	**50.64**
	+5.53% ↑	+7.73% ↑	+0.97% ↑	+12.50%↑

Table 1: Main results of performance comparison between LLM w./w.o. sentence segmentation applied in ICL approach. {Model Name}-seg represents model evaluating with sentence segmentation. The segmentation token here we used is "**<seg>**".

Preprocessing All input texts are preprocessed by a model released by Frohmann et al. (2024), named SaT-12L-sm (which we refer to as SaT directly), a SOTA model in sentence segmentation. It is a BERT-like model with 12 layers. For each input, the model returns a list of segmented sentences. Then we insert a segment token (e.g.: "**<seg>**") after each sentence, and concatenate them back into the original text. In ICL experiments, we directly take the segmented text as input. For SFT experiments, we treat the segment token as a new special token and introduce it in finetuning, so that it could learn from the sentence segmentation.

4.2 In-context Learning Experiments

4.2.1 Main Results of In-Context Learning

The main results of ICL are shown in Table 1. It demonstrates that both the two models with sentence segmentation applied outperform the two models without sentence segmentation applied, except the result of MATH in Llama3-8B-Instruct. An obvious enhancement appears in MMLU and GSM8k in both models, also in MATH in Qwen-7B-Instruct. A greater enhancement appears in DROP, which improves in nearly 6.8% in Llama3-8B-Instruct, and 12.5% in Qwen2-7B-Instruct. The table shows that segmenting the input into sentences can improve reasoning performance through ICL. The sentence structure indeed helps LLM in reasoning tasks.

We find that the only opposite effect in the result is MATH in Llama3-8B-Instruct. The same situation also appears in Section 4.2.2's experiments. We conjecture the reason is that Llama3-8B-instruct has a weaker reasoning ability compared to Qwen2-7B-Instruct. Therefore, for a harder math problem, the model's reasoning ability failed to meet the passing standard. Even if segmentation is

	Llama3-8B			Qwen2-7B		
	GSM8k	MATH	DROP	GSM8k	MATH	DROP
orig.	75.51	32.60	46.39	73.92	53.33	38.14
<seg>	78.01	**32.26**	53.16	81.65	54.30	50.64
seg	77.17	30.58	28.30	80.36	54.30	44.56
<and>	77.17	31.48	53.14	**82.41**	58.70	**51.98**
and	74.34	31.10	53.14	74.68	58.84	47.24
####	77.86	31.18	53.30	80.81	**60.18**	49.58
.$?	78.01	31.90	**53.85**	82.17	58.40	16.17
114	**78.46**	31.56	**53.85**	81.88	57.12	36.81

	<seg>	seg	<and>	and	####	.$?	114
Avg. Improve	**+5.02**↑	-0.77↓	**+5.83**↑	+3.24↑	**+5.50**↑	+0.10↑	+3.30↑

Table 2: Results of the performance of different segmentation tokens in ICL. The upper table is the comparison between different tokens. The the left first column is the segmentation tokens we tested, "orig." represents that no segmentation token is applied. The lower table is the average improvement of segmentation tokens.The three segmentation tokens that have an overall better performance are bold.

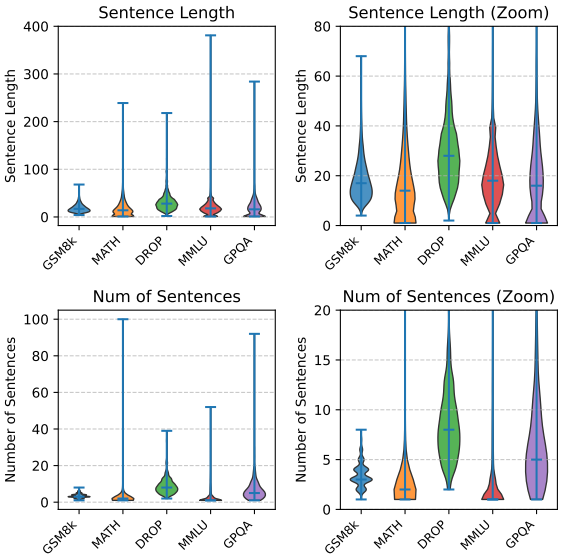


Figure 3: The distribution of sentence length and number of sentences of each datasets. The left column figures are the origin distribution, the right column figures are zoomed up based on the left figures. We mark the medians and extrema on these figures. The length is the length of tokenized sentence, tokenized by Llama3 tokenizer.

applied, the abilities to solve such problems cannot be enhanced.

4.2.2 Different Segmentation Tokens Affect the Results

In Section 4.2.1, the main experiments of ICL used "**<seg>**" as segmentation token. However, the semantic information of the segmentation token may affect the LLM in recognizing sentence structure. To see how the difference of segmentation tokens affects ICL performance, we conducted more experiments on it.

Table 2 shows the ICL results with different segmentation tokens. The result shows that LLM can recognize the sentence structure and thereby performs better if the token is wrapped by "**<>**", e.g., "**<seg>**" performs better than "**seg**", so as "**<and>**". This indicates that structured segmentation tokens are easier for LLM to recognize. Also, we found that using tokens that contain semantic information like "**seg**" could be harmful in some tasks, while using "**and**" does not; we suggest that the reason is word "**and**" performs some degrees of semantic segmentation in a paragraph, while "**seg**" is actually not.

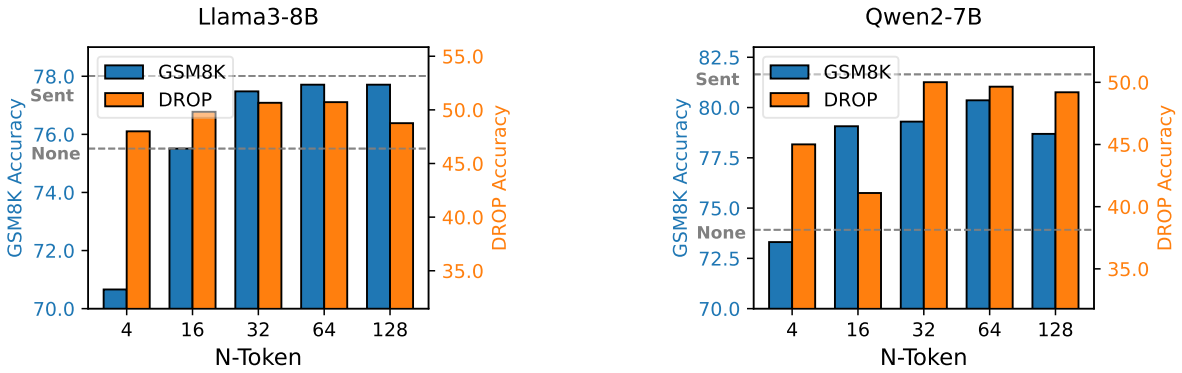
To further study the effects of different semantic information, we tested three tokens, "**####**", "**.$?**", "**114**", which correspond to structured information (marker between CoT and final answer in GSM8k), punctuations, and numbers. We are surprised to find that, although "**.$?**" and "**114**" do not perform as well as others, they are able to enhance the performance on many tasks. It suggests that LLMs are sensitive to regularly repeated tokens, and are able to exclude the influences of harmful semantic information. "**####**" performs generally as well as "**<seg>**" and "**<and>**", averagely better than "**.$?**" and "**114**".

In conclusion, the average improvements of "**<seg>**", "**<and>**" and "**####**" are larger than other segmentation tokens. It supports our assumption: a more structured segment token can have better overall performance. The result shows that for ICL, it is better to choose a segment token that contains more structured information and less semantic information.

4.2.3 Sentence Segmentation vs. N-Token Segmentation

Next, we explore whether the sentence structure under human prior knowledge is optimal for LLMs. To find the answer, a comparison of downstream performance is made between segmentation based on sentence and segmentation based on n-token.Segmentation of n-token is to insert the segment token in the tokenized text every n tokens. We use "**<seg>**" as the segmentation token in this section. The results are shown in Figure 4.

From the figure, we can see that the performance



(a) Results of Llama3-8b-Instruct

(b) Results of Qwen2-7b-Instruct

Figure 4: The results of Llama3-8b-Instruct and Qwen2-7b-Instruct over GSM8k and DROP datasets on in-context learning experiments. We compare different segmentation methods. The x-axis is the n of n-token segmentation (e.g.: 32 represents 32-token segmentation). "None" represents no segmentation applied, and "Sent" represents segment via sentence structure. For Llama3-8B, the upper gray line of "Sent" represents GSM8k acc. 78.01, and DROP acc. 53.16, the lower gray line of "None" represents GSM8k acc. 75.51, and DROP acc. 46.39. For Qwen2-7B, the upper gray line of "Sent" represents GSM8k acc. 81.65, and DROP acc. 50.64, the lower gray line of "None" represents GSM8k acc. 73.92, and DROP acc. 38.14.

of n-token segmentation with $n \geq 4$ outperforms that of no segmentation, except 4-token segmentation, where both LLMs perform poorly in GSM8k. The performance increases as n increases, until it reaches $n = 128$, where both LLMs are poorer in both tasks compared to $n = 64$. We make a new assumption that the segment tokens are to help LLMs to summarize the previous information, or to help LLMs to stop and think. Therefore, n-token segmentation with $n = 4$ makes the information too sparse, making the semantic information fragmented; while n becomes much larger ($n = 128$) making the situation similar to no segmentation, so that the performance decreases. For $n = 16, 32, 64$, there are less fragments as n increases, so more semantic information is kept inside the segments.

We plot a violin plot to demonstrate the distribution of the sentence length and the number of sentences of these datasets in Figure 3. It shows that a sentence contains mostly 5-40 tokens. Therefore, in the aspect of sentence structure, the case of $n = 4$ results in too many fragments. With $n = 16$, some short sentences can be included. $n = 32$ can contain most sentences, while $n = 64$ can cover almost all sentences and sometimes include multiple sentences. $n = 128$ is essentially equivalent to a text that has not been segmented in GSM8k, since GSM8k contains less than 5 sentences, while DROP contains more (less than 15), it fits our observation that the decrease in DROP from $n = 64$ to $n = 128$ is less than GSM8k.

However, segmenting by sentence is always better than n-token. This means that keeping the semantic information of a sentence not to be fragmented would be the best choice. The sentence structure is the optimal solution to segmentation.

4.3 Supervised Finetuning Experiments

The main results of SFT experiments are shown in Figure 5. We finetuned two Llama3-8B-Base models w./w.o. sentence segmentation on the TULU3 subset. The two finetuned model were tested on the seven datasets. The results show that Llama3-8b-seg outperforms all tasks compared to Llama3-8B. We observed a steady performance improvement on the GSM8k, MATH, and DROP datasets tested in the ICL experiments, which was in line with our expectations. The newly included datasets, MMLU-Pro, GPQA, and HumanEval, also improve after segmentation.

We aim to show that: sentence segmentation can be learned by training, in other words, LLMs are able to learn to output segment tokens automatically, therefore, a long context is not a must. Putting aside those datasets that need to be tested with n-shot settings, we found that for the three 0-shot datasets, Llama3-8B-seg all outperforms Llama3-8B, although not as significant as those n-shot datasets. It suggests that LLMs can learn sentence segmentation (by outputting the segment tokens) and gain an enhancement through segmentation, while a long context could make such en-

	MMLU	GSM8k	MATH	DROP	MMLU-Pro	GPQA	HumanEval
Llama3-8B	59.02	72.48	30.86	48.50	34.25	26.93	56.71
Llama3-8B-seg	**60.13**	**74.91**	**31.58**	**53.26**	**40.71**	**27.43**	**62.80**
	+1.11%↑	+2.43%↑	+0.72%↑	+4.76%↑	6.46%↑	0.50%↑	6.09%↑

Figure 5: Main results of performance comparison between LLM w./w.o. sentence segmentation applied in SFT approach. Here the Llama3-8B model are finetuned with TULU3 subset. We treat segment token as a special token inside tokenizer.

hancement higher.

We are surprised to see that Llama3-8B-seg greatly outperforms Llama3-8B on HumanEval. Since SaT are only trained in human languages, we did not expect it to be effective for coding, as their structures are different. However, the results make us believe that there exist some commonalities between human language and code. Such commonalities enable the segmentation patterns learned from human language to also be effective for code tasks.

Since we treat the segmentation token as a special token as described in Section 3.4, we do not need to be concerned about the semantic information that the segmentation token may introduce to the experiments.

4.4 Analysis

Information Flow of Language Modeling The stack of same-token representations is referred as a "residual stream" (Nelson Elhage, 2021), and the overall computation can be viewed as a sequence of residual streams connected through layer blocks, Ferrando and Voita (2024) define the residual streams along with attention edges, FFN edges as information flow. Therefore, (Ferrando and Voita, 2024) suggests to use graph visualization to track the information flow. We use the visualization tools proposed by (Tufanov et al., 2024) for information flow visualization. The visualization results are in Appendix C. Through the figure, the last token absorbs the information of the entire paragraph. In particular, the information of segmentation tokens flow into the last token at a relatively high layer (usually higher than the other tokens within the same sentence). Moreover, we have found that the information within a sentence tends to converge on the segmentation token first.

Attention Map We also use heat map for attention visualization. The results of attention map are shown in Appendix C. Through the attention map, we find that the segmentation token (along with the tokens surrounding the segmentation token) tend to capture the majority of the attention. Especially, we observe that the query tokens in the latter half of the sentence tend to focus their attention particularly on the segmentation token.

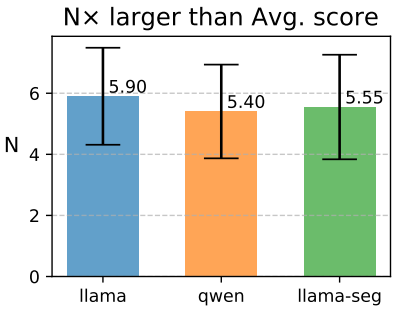


Figure 6: The average attention score of segmentation tokens obtained from the last token in GSM8k dataset is always larger than the average attention score. We illustrate how much larger are the segmentation tokens' attention score is compared to the average attention score.

Average Attention Score of Segmentation Tokens For a more comprehensive understanding of the phenomenon inside attention, we calculate the average attention score of the segmentation tokens and compare it with the overall average attention score of the sentence. The calculation is applied on the attention scores of the last token with respect to the entire sentence. Figure 6 shows that the attention score of the segmentation tokens is always significantly higher than the average attention score of the sentence.

Assumptions Based on Visualization Based on the visualization results in information flow and the attention map, and also the numeric results in the average attention score of the segment tokens, we assume that the segmentation token has the role of "summarizing the information within the sentence". Specifically, the segment token aggregates

the information within the sentence onto itself, and increases its own weight in the attention of the subsequent tokens. We think the results suggest that the segment tokens could help LLMs to stop at the segment token and think about the information that the previous sentence include; and this behavior is benefit for the final performance.

Assumptions on Why Segmentation Tokens better than Punctuations There might be a question: why does the segmentation token work better than punctuation, since punctuation is enough to delineate sentences? We think the reason might be that punctuation contains both structural information and semantic information, so that the structural information is implicit rather than explicit. Therefore, an LLM cannot directly judge the sentence structure from punctuation tokens, except it is required by prompts. In contrast, the only usage of segment tokens is to demonstrate the sentence structure, so that the structural information is explicitly included. We plot the N$\times$ larger figure in Appendix B, which shows that punctuations generally obtain a lower attention score than segment tokens. We conclude that it is easier for an LLM to assign its attention to segment tokens than punctuations.

5 Conclusion

In conclusion, this study proposed a new method to improve LLM performance by sentence segmentation. Introducing sentence segmentation into LLM inference can be helpful for language modeling. We studied two approaches to apply sentence segmentation, ICL and SFT, both of which are able to work. ICL is a cheap and efficient approach that can directly improve downstream performance. However, ICL requires a long context for LLM to learn from analogy, which is probably not capable for most application scenarios. On the other hand, SFT approach does not require a long context to learn from. By adding the segment tokens inside the training target, it requires the model to explicitly learn to segment sentences (by outputting segment tokens). It can work directly on 0-shot task, which is closer to most of the real-world applications.

Further ablations show that semantic information could affect the performance in ICL, therefore a more structured segment token is a better choice. Moreover, we discuss the level of segmentation, and find out that although token-level segmentation is somehow enough to enhance the performance, sentence-level segmentation always works better than token-level segmentation. Our visualization indicates that such enhancements of sentence segmentation may be due to the summarization and aggregation effects of segmentation tokens during language modeling.

Our work shows that introducing sentence structure into language modeling can actually improve LLM performance. It is a cheap and easy approach that only needs to add a segment token after each sentence during the data processing procedure, while the improvement is significant. It encourages us to look more on language structure and use such human prior knowledge to help with language modeling task.

Finally, we want to discuss more about the potential by-product of our work that could be useful in any work about **LLM+sentence**. Work such as MCTS, decoding strategy, and sentence-level reward model, is required to segment the sentence during either preprocessing stage or postprocessing stage if any of them want to study the effects of treating sentence as a unit. However, achievements by prompting methods are somehow harmful to performance, and achievements by segmentation models are both time-consuming and resource-consuming. It would be much more convenient if an LLM could segment the sentence itself. Our research shows that the use of ICL is not harmful, even beneficial to performance; while SFT is better for applications that do not contain a long context. Therefore, we are here calling for the use of this methodology within the next generation of LLMs.

6 Limitation

The work has the following limitations: First, the robustness of the proposed method is not tested on other segmentation methods (such as rule-based methods), since most of the applications care more about the SOTA method. Secondly, although we conducted the experiment and proved the effectiveness of sentence segmentation on 7B-level LLMs, a further scaling experiment is needed. Lastly, due to the limitation of GPU resources, the effectiveness of sentence segmentation is only examined on the finetuning stage; although we believe that the SFT results are enough to demonstrate usability in pretraining, it still needs to be confirmed through further experiments.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, et al. 2021. Evaluating large language models trained on code.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, et al. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Julien Launay Thomas Wolf Clémentine Fourrier, Nathan Habib. 2023. What's going on with the open llm leaderboard?

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *Preprint*, arXiv:2401.02954.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *Preprint*, arXiv:1903.00161.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale. *Arxiv*.

Markus Frohmann, Igor Sterner, Ivan Vulić, Benjamin Minixhofer, and Markus Schedl. 2024. Segment any text: A universal approach for robust, efficient and adaptable sentence segmentation. *Preprint*, arXiv:2406.16678.

Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, Ramaneswaran S, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, and Dinesh Manocha. 2024. A closer look at the limitations of instruction tuning. *Preprint*, arXiv:2402.05119.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. *Preprint*, arXiv:2310.02226.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, et al. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Daphne Ippolito, David Grangier, Douglas Eck, and Chris Callison-Burch. 2020. Toward better storylines with sentence-level language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7472–7478, Online. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, et al. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi,

Noah A. Smith, and Hannaneh Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling. *Preprint*, arXiv:2403.13787.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *Preprint*, arXiv:cs/0205028.

Benjamin Minixhofer, Jonas Pfeiffer, and Ivan Vulić. 2023. Where's the point? self-supervised multilingual punctuation-agnostic sentence segmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7215–7235, Toronto, Canada. Association for Computational Linguistics.

Catherine Olsson Tom Henighan Nicholas Joseph Ben Mann Amanda Askell Yuntao Bai Anna Chen Tom Conerly Nova DasSarma Dawn Drain Deep Ganguli Zac Hatfield-Dodds Danny Hernandez Andy Jones Jackson Kernion Liane Lovitt Kamal Ndousse Dario Amodei Tom Brown Jack Clark Jared Kaplan Sam McCandlish Chris Olah Nelson Elhage, Neel Nanda. 2021. A mathematical framework for transformer circuits.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, et al. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *Preprint*, arXiv:2408.06195.

Wenjie Qiu, Yi-Chen Li, Xuqin Zhang, Tianyi Zhang, Yihang Zhang, Zongzhang Zhang, and Yang Yu. 2025. Sentence-level reward model can generalize better for aligning llm from human preference. *Preprint*, arXiv:2503.04793.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, page 476–483. IEEE.

Nipun Sadvilkar and Mark Neumann. 2020. Pysbd: Pragmatic sentence boundary disambiguation. *Preprint*, arXiv:2010.09657.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, et al. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Igor Tufanov, Karen Hambardzumyan, Javier Ferrando, and Elena Voita. 2024. Lm transparency tool: Interactive tool for analyzing transformer language models. *Arxiv*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Rachel Wicks and Matt Post. 2021. A unified approach to sentence segmentation of punctuated text in many languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3995–4007, Online. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, et al. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Preprint*, arXiv:2406.03816.

A Tries of Finetuning Instructed Model

We made several tries to finetune an instructed model Llama3-8b-Instruct. Both full parameter finetuning and LoRA finetuning (Hu et al., 2021) are tested. Amount all our tries, the model's performances are always ruined by the finetuning procedure, making the performances decrease compared to it's original performances. However, we observed an interesting phenomenon that, no matter how we reduce the training size of the dataset (from the full dataset to the 10% dataset), the full-parameter finetune always outperforms the LoRA finetune. According to Ghosh et al. (2024), the LoRA finetune is the initiation of response and style tokens learning. Therefore, we conclude that sentence segmentation is not a kind of pattern or format for style learning, it is a kind of knowledge that requires a full parameter finetune to inject itself into the model.

B Average Attention Score of Punctuations

For comparison with segmentation tokens, we compute the average attention score of punctuation (e.g.: ".", "!", ","...) and the newline character ("\n"). The results are shown in Figure 7. Compared to Figure 6, we can see that the average attention score of punctuation is less than the segmentation tokens.

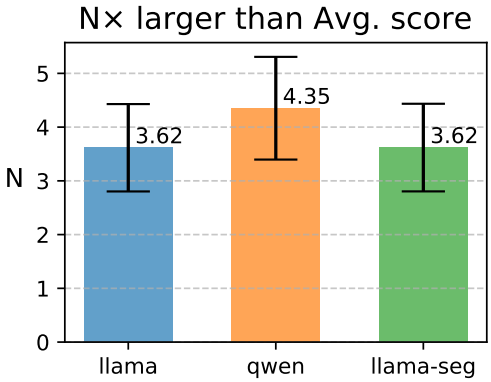


Figure 7: The average attention score of punctuations and newline character obtained from the last token in GSM8k dataset.

C Visualizations of Information Flow and Attention Map

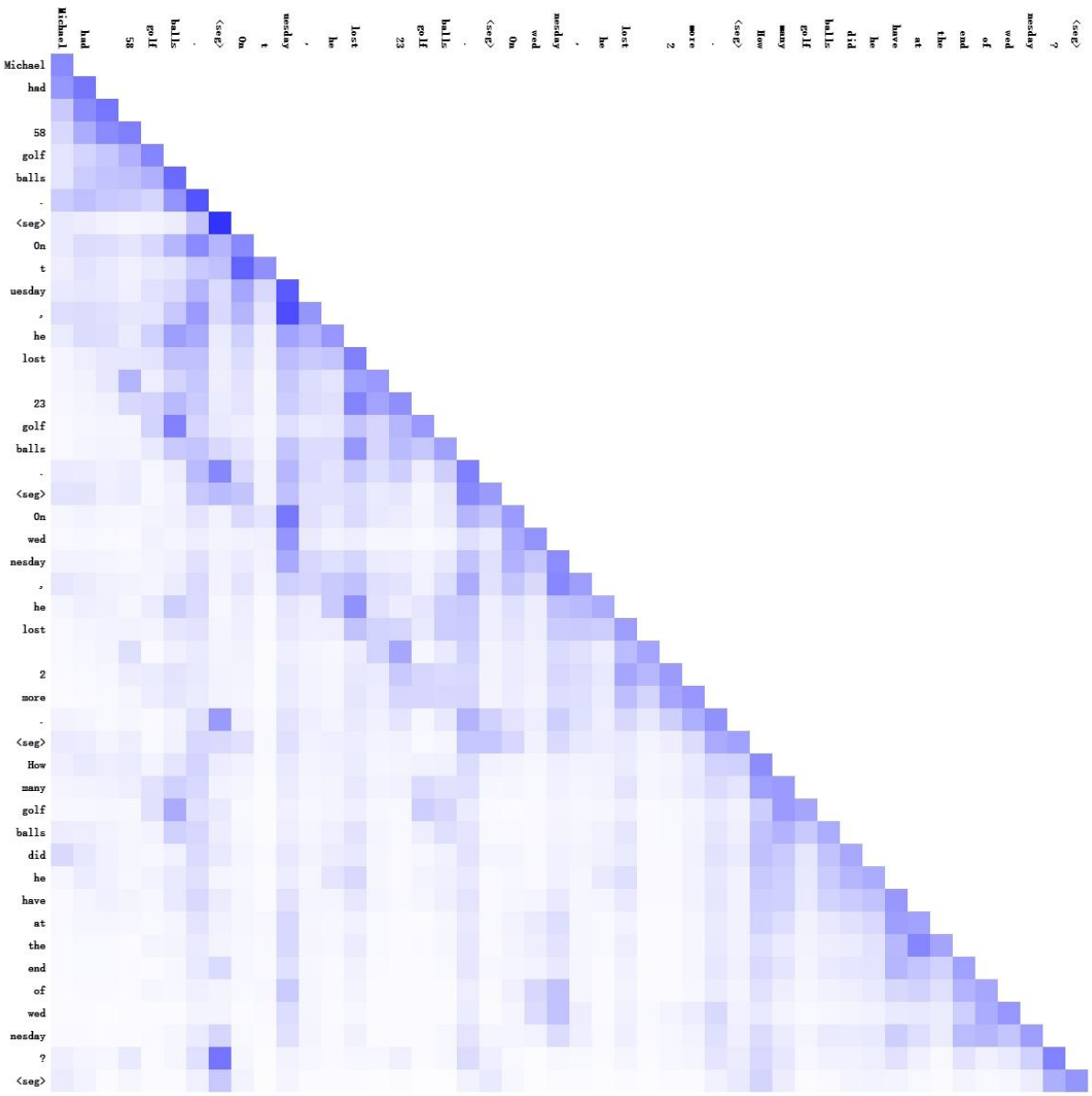


Figure 8: Attention map of Llama3-8b-SFT $_{\text{seg}}$

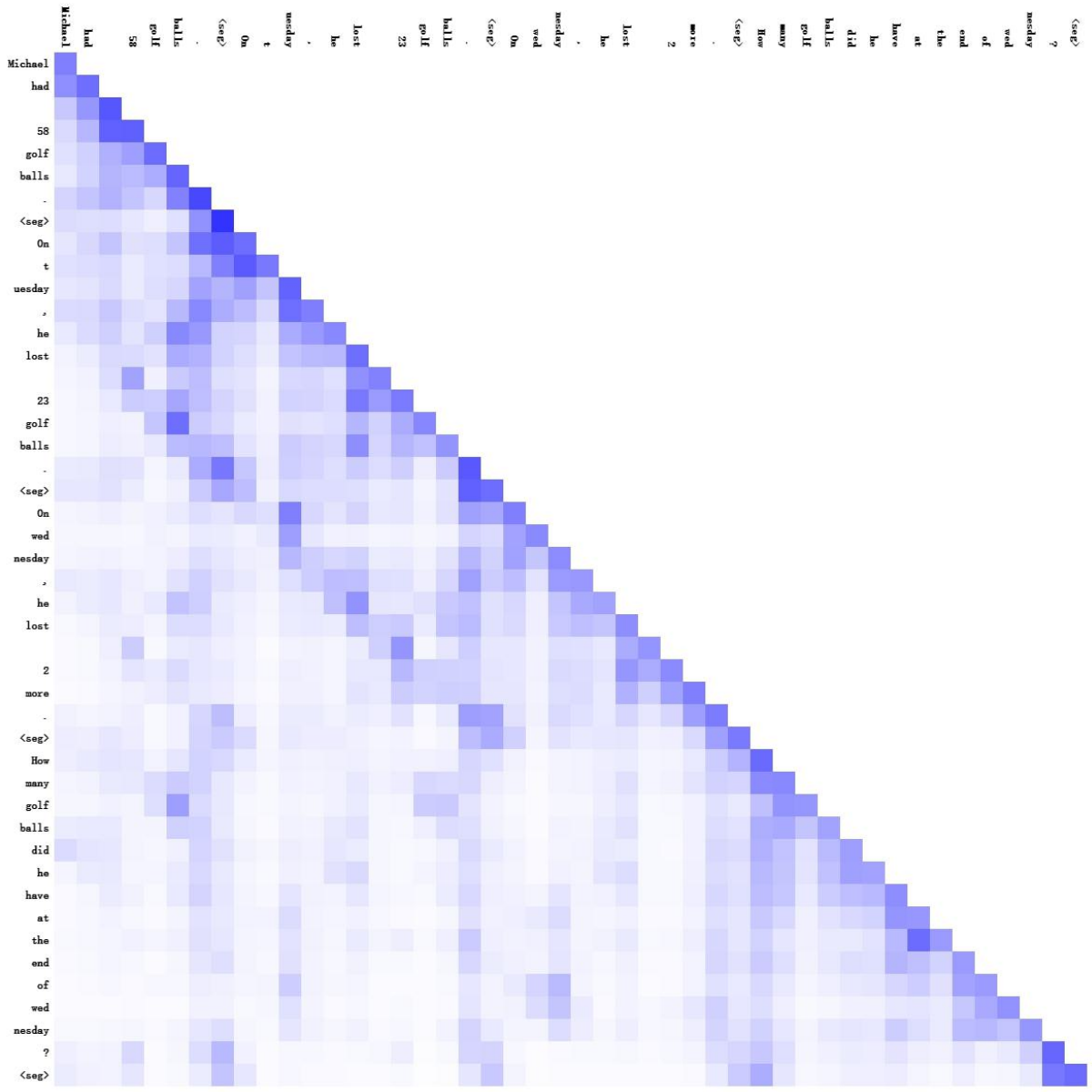


Figure 9: Attention map of Llama3-8b-Instruct. The segmentation token we used is "**####**". We replaced it to "**<seg>**" only when visualization.

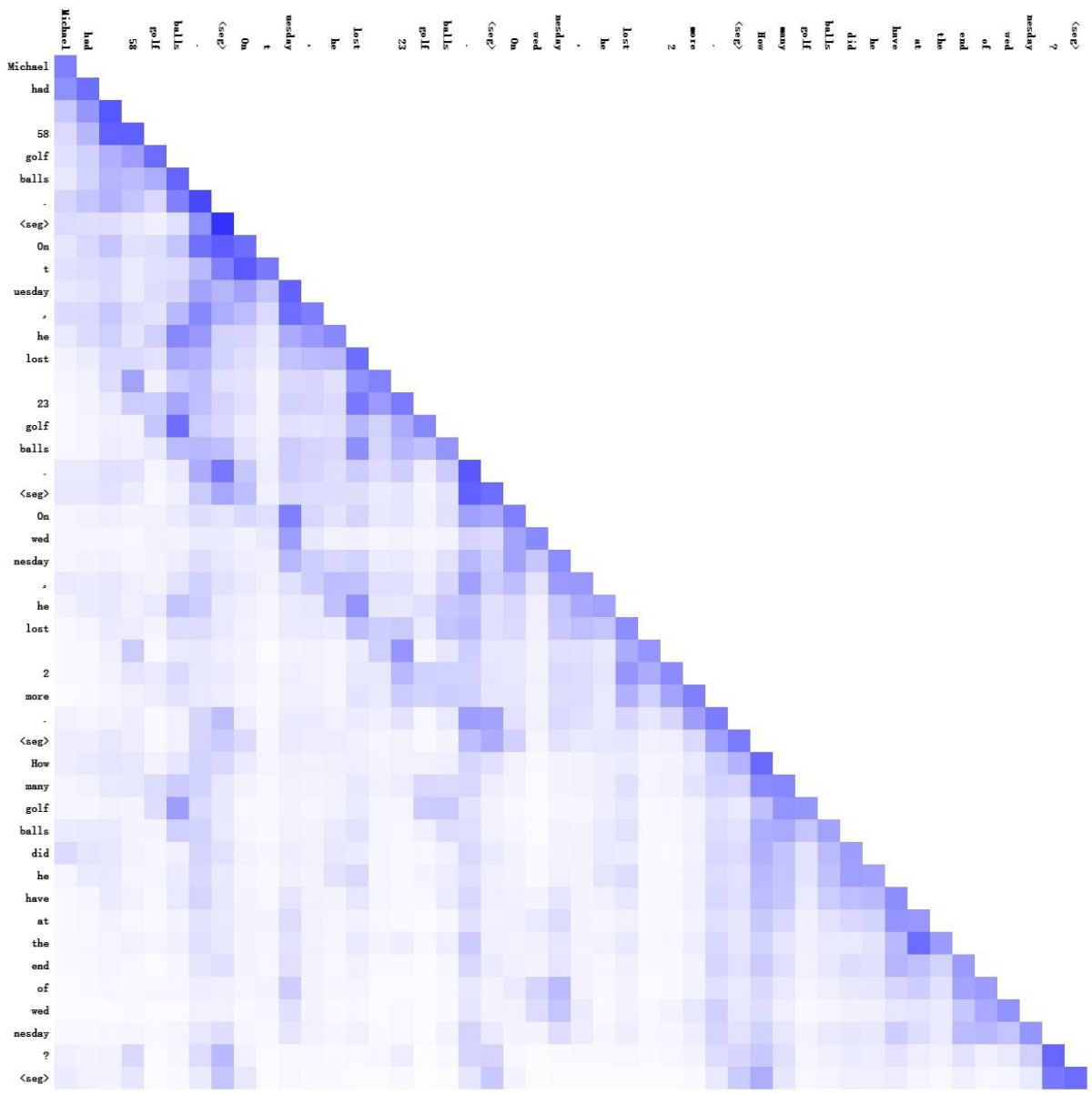


Figure 10: Attention map of Qwen2-7b-Instruct. The segmentation token we used is "**####**". We replaced it to "**<seg>**" only when visualization.

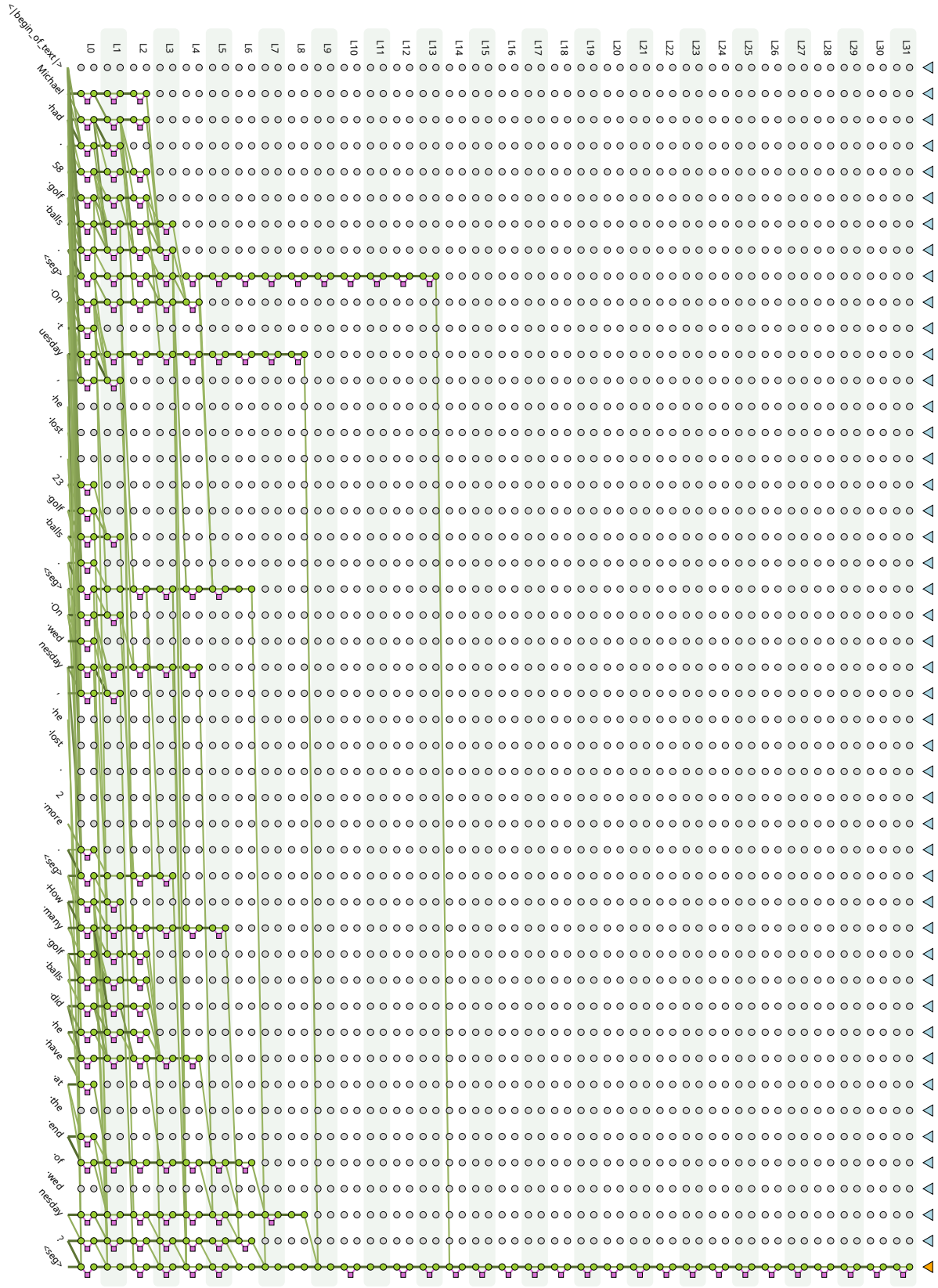


Figure 11: Information flow of Llama3-8b-SFT $_{\text{seg}}$

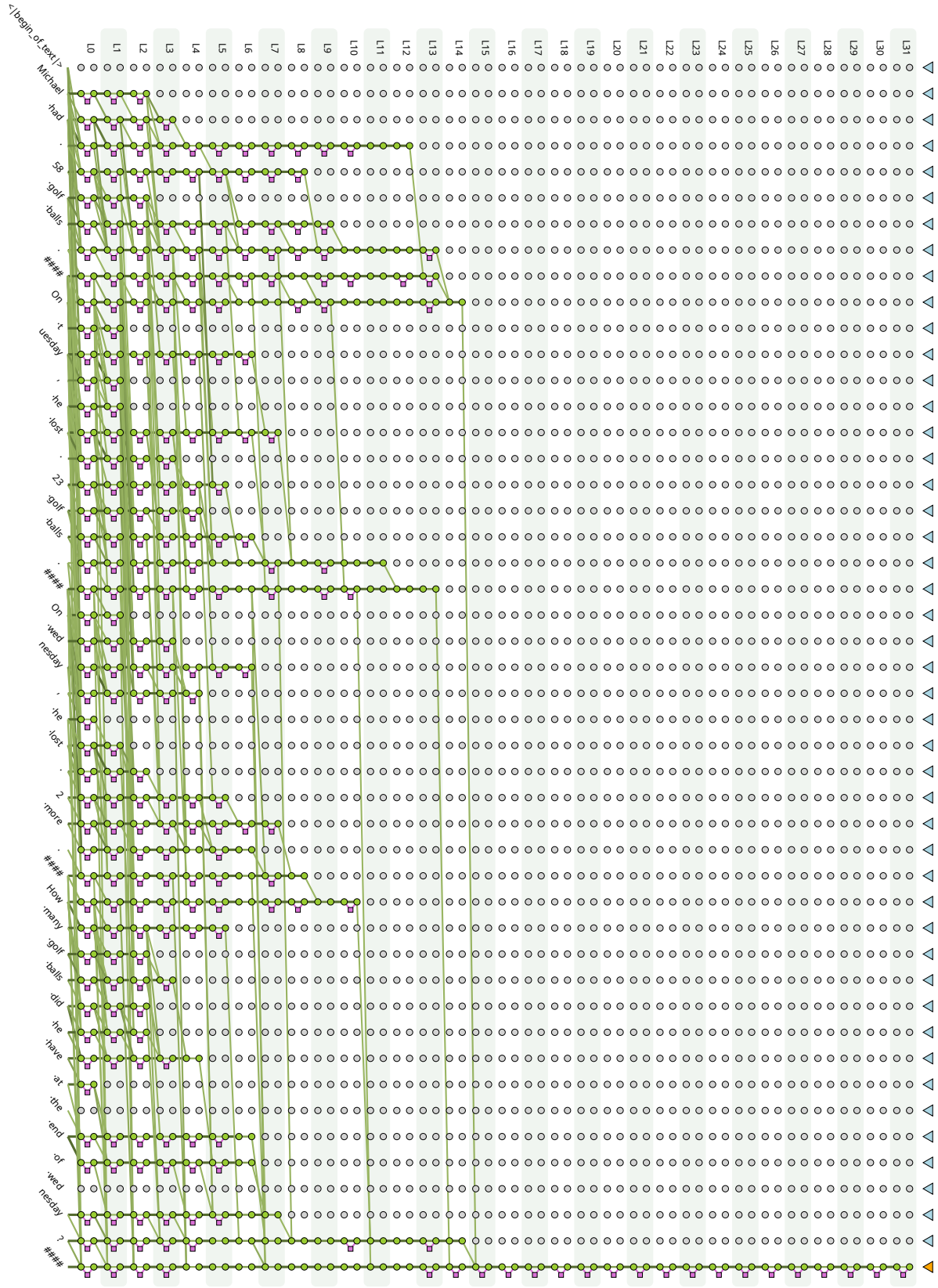


Figure 12: Information flow of Llama3-8b-Instruct on segmentation token "####"

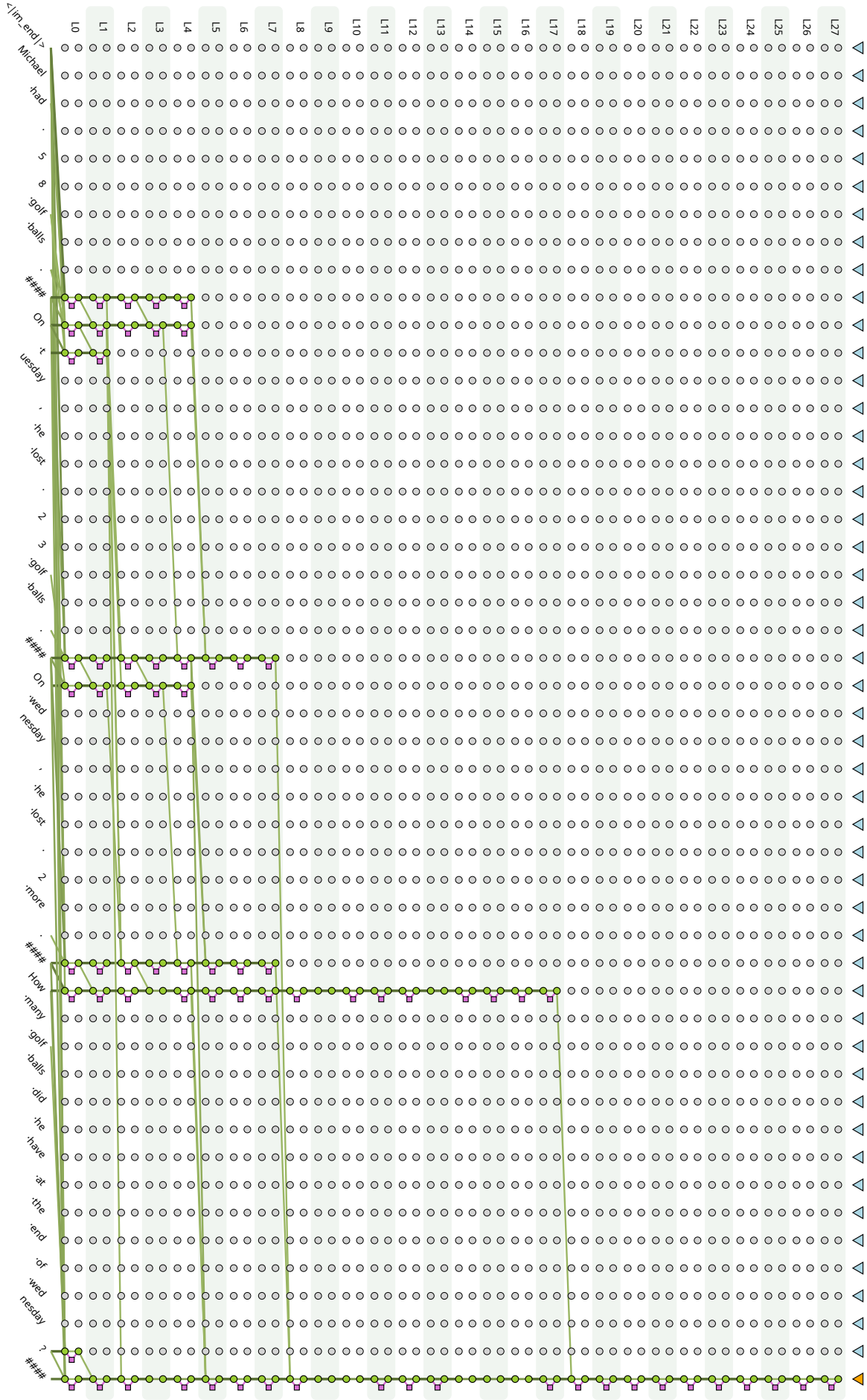


Figure 13: Information flow of Qwen2-7b-Instruct on segmentation token "####"