END: Early Noise Dropping for Efficient and Effective Context Denoising

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated remarkable performance across a wide range of natural language processing tasks. However, they are often distracted by irrelevant or noisy context in input sequences that degrades output quality. This problem affects both long- and short-context scenarios, such as retrieval-augmented generation, table question-answering, and in-context learning. We reveal that LLMs can implicitly identify whether input sequences contain useful information at early layers, prior to token generation. Leveraging this insight, we introduce Early Noise Dropping (END), a novel approach to mitigate this issue without requiring fine-tuning the LLMs. END segments input sequences into chunks and employs a linear prober on the early layers of LLMs to differentiate between informative and noisy chunks. By discarding noisy chunks early in the process, END preserves critical information, reduces distraction, and lowers computational overhead. Extensive experiments demonstrate that END significantly improves both performance and efficiency across different LLMs on multiple evaluation datasets. Furthermore, by investigating LLMs' implicit understanding to the input with the prober, this work also deepens understanding of how LLMs do reasoning with contexts internally.

1 Introduction

Large language models (LLMs) have exhibited impressive performance across a wide range of natural language processing tasks. As their application scope expands, the input lengths of LLMs are also increasing rapidly (Dubey et al., 2024; Yang et al., 2024; Liu et al., 2024). However, when processing long sequences, LLMs often face a significant challenge from the noise in the contexts which may distract LLMs. This issue arises when models fail to effectively utilize relevant contextual information, becoming distracted by irrelevant or noisy data instead. Consequently, the quality of their output deteriorates, resulting in inaccuracies, hallucinations, incomplete responses, and occasional failures to follow instructions (Anil et al., 2024; Shi et al., 2023).

Such distraction is **common and not limited to** long-sequence tasks. For instance, while retrieval-augmented generation (RAG) can filter out most noise, the prompt delivered to LLMs after retrieval and reranking still contains a significant amount of irrelevant information. Similar challenges arise in table-based question answering (QA), multi-turn dialogue QA, and in-context learning (ICL). In these scenarios, only a small fraction of the provided context is directly relevant to the query, whereas the remaining information—despite often being contextually similar—acts as noise, potentially hindering the model's ability to focus on the most relevant content and also affecting efficiency.

Existing approaches to mitigate this issue primarily fall into two categories. The first involves multi-agent collaboration frameworks (Team, 2024; Lee et al., 2024), where the context is split into segments processed by different agents, followed by inter-agent interaction to produce a final output. While effective, these methods often **require complex agent interaction designs and multiple inference steps**, resulting in increased latency. The second approach, known as "Parallel Context Encoding" (Yen et al., 2024; Merth et al., 2024), either trains an additional encoder to process context segments before feeding them to the LLM or trains a separate module to aggregate outputs from multiple LLM runs on different segments. However, these solutions **necessitate non-trivial training of new components of the LLM itself** and also incorporates significant changes to LLMs' original forward mechanism.

This motivates us to explore whether it's possible to leverage a LLM's inherent ability to handle noisy cases without fine-tuning. Inspired by previ-

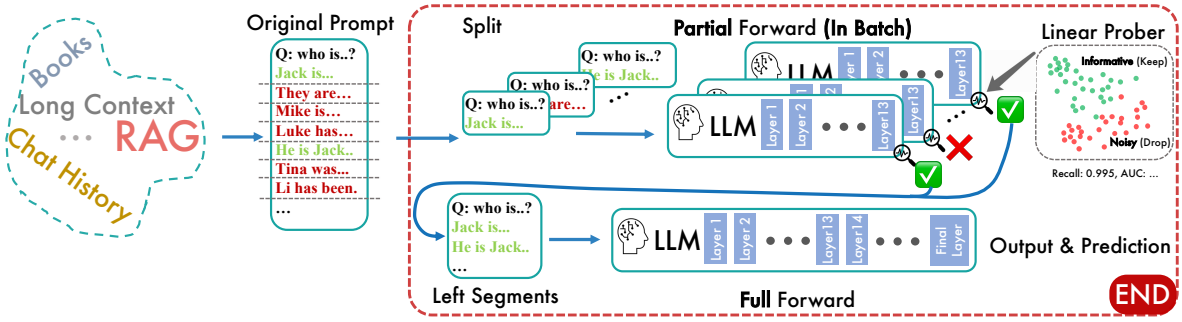

Figure 1: The framework of the proposed END. The long input, which might come from various sources such as RAG, will be split into chunks for *parallel* processing first. It's a partial forward. A linear prober is attached to the designated layer (layer 13 in the diagram) of a LLM, and the prober is used to determine whether a chunk is noisy (red) or not (green). After that, those selected informative chunks will be combined together as the new input for the LLM, which will generate the final prediction.

ous works suggesting that LLMs implicitly know when they are generating incorrect responses (Kadavath et al., 2022; Yin et al., 2023), we hypothesize that LLMs can identify whether the input contains useful information related to the questions before generating the first response token. Our experiments, using a simple linear prober, strongly support this assumption and reveal that such distinguishing abilities emerge in very early layers.

Based on these findings, we propose "Early Noise Dropping" (END), a novel approach to mitigate noise distraction in LLMs. END segments the input into multiple chunks and processes them in parallel. A linear prober, operating on hidden states from lower layers, distinguishes between informative and noisy chunks, discarding irrelevant ones early while retaining essential context for the final prediction. The remaining chunks are then combined and processed in a full forward pass. This method significantly reduces computational overhead while preserving key information. Compared to strong baselines, including direct LLM noise discrimination approaches, END achieves over 10% performance improvement and reduces computation by approximately 50%.

Crucially, the proposed END requires no fine-tuning, instead leveraging LLMs' innate ability to discern task-relevant information. By dropping redundant chunks early, it enhances both efficiency and accuracy while shedding light on LLMs' internal noise discrimination mechanisms. Our main contributions include:

- We identify LLMs' noise sensitivity: Even trivial noise distracts LLMs, while noise resembling target information severely degrades performance.
- We find that LLMs can internally differentiate relevant and irrelevant context at lower layers, which can be effectively exploited via a linear prober.
- We propose END: By selectively processing input chunks, END mitigates noise distraction effectively and efficiently. Comprehensive experiments confirm its superiority over baselines.

2 Methodology

In this section, we first show the harmfulness caused by noise in the context and the impact of longer contexts. Then via constructing a linear prober, we elicit the LLMs' inherent capabilities of being aware of whether the input is informative or noisy. Finally, based all previous results, we introduce END, which processes segmented long input with the help of a linear prober, and only retains informative segments for predictions.

Difficulty Level	Standard	Extended (Long)
Level_0	1.00	1.00
Level_1	1.00	0.94
Level_2	0.91	0.71
Level_3	0.54	0.40
Level_4	0.08	0.04

Table 1: C omparison of performance (accuracy) on NoisyRetrieval between "Standard" ($\approx$ 4k) and "Extended" ($\approx$ 8k) for `Llama3-Instruct-8B` with different Difficulty levels. Level_0 is least confusable and Level_4 is most confusable. By adding random texts to the contexts, each instance from "Standard Context" is transformed into "Extended Context". Such trivial noise still significantly harms the performance.

2.1 The Impact of Noisy Contexts from Longer Contexts.

In Table 1, we show how the performance of `Llama3-8B-instruct` on our synthetic task **NoisyRetrieval** shifts under two key factors: (1) the difficulty of added noise and (2) overall context length. Briefly, for each question, we add a single true segment (containing the answer) plus 12 distraction segments that range in noise difficulty from Level_0 (least confusable) to Level_4 (most confusable). These distraction segments have information similar to the true answer but do not contain the correct answer (Further details on how we generate these segments and control the noise difficulty for this task appear in Section 3.1.1). In this table, we refer to it as the standard context setting for each instance has approximately 4k tokens. Additionally, we test an extended context setting with trivial noise (2× length, ≈ 8k tokens) by adding random texts. Even this seemingly harmless extra material leads to worse performance, particularly when combined with harder noise (e.g., Level_4). We draw two conclusions: (1). The sharp decrease in performance as the noise level increases highlights the critical need for removing noise from the input contexts. (2). The performance gap between the "Standard" and "Extended" settings demonstrates that even trivial noise in longer context can cause severe performance degradation. This finding underscores the importance of reducing input length and mitigating noise to optimize LLM performance.

2.2 A Linear Prober can Effectively Elicit LLMs' Inherent Discrimination Capabilities.

Previous works on LLMs' implicit abilities suggest that while generating predictions for a task, LLMs inherently perform many functions beyond the task itself. For instance, Slobodkin et al. (2023) found that the last layer's hidden states can be used to reveal whether the generated prediction is hallucination or not.

This insight prompts us to ask: **Do LLMs inherently check whether the input contains information related to the current queries?** We hypothesize that the answer is 'Yes', considering that LLMs perform well when directly asked whether a given input can answer a specific question. To validate this hypothesis, we constructed linear probers for four models: `Llama3-8B-Instruct`, `Mistal-Inst-v0.3`, `Qwen2-7B-Instruct`, and `Gemma-1.1-7b-it`, across three datasets: *NoisyRetrieval*, *NaturalQA*, and *TriviaQA*. Details of these datasets can be found in Section 3.1.1. Following common practice, the linear prober takes the hidden representation of the last input tokens and predicts binary labels for each input: '0' for input irrelevant to the query and '1' for input informative to the query. We also conducted a layer-wise analysis to reveal how these inherent abilities emerge in LLMs. The results are shown in Figure 2. For each query, we set 10 negative inputs and 1 positive input.We can conclude the following from our results:

- LLMs can successfully determine whether the input is noisy via a simple linear prober. Although we don't explicitly require the model to discriminate the input in the prompts, the linear prober performs remarkably well.
- Such discrimination abilities arise at very early layers. Using the hidden states from layers around layer 13, the linear prober can already achieve a high recall (>0.95) for the positive input (i.e., the non-noise input).
- Different models exhibit similar behavior, suggesting that these inherent abilities are universal across LLMs.

These findings underscore the potential for leveraging LLMs' innate discrimination capabilities in handling noisy or irrelevant input data.

2.3 END: Early Noise Dropping.

The core idea of the proposed END is to segment the input sequence into multiple chunks, process these chunks in parallel through the lower layers of the LLM, identify and retain only the essential chunks, and finally forward the retained chunks for the final prediction. This approach leverages the LLM's inherent ability to discriminate between informative and noisy input early in its processing stages. The framework of the proposed END is shown in Figure 1. Its details are as follows:

Step 1: Input Segmentation. We begin by dividing the input sequence into multiple chunks. The optimal chunk size may vary depending on the specific task and model architecture. Most chunks will only contain noise rather than the answer to the task. These chunks are then processed in parallel through the lower layers of the LLM, allowing for efficient computation. During this parallel processing, each chunk is attached to a task-related query.

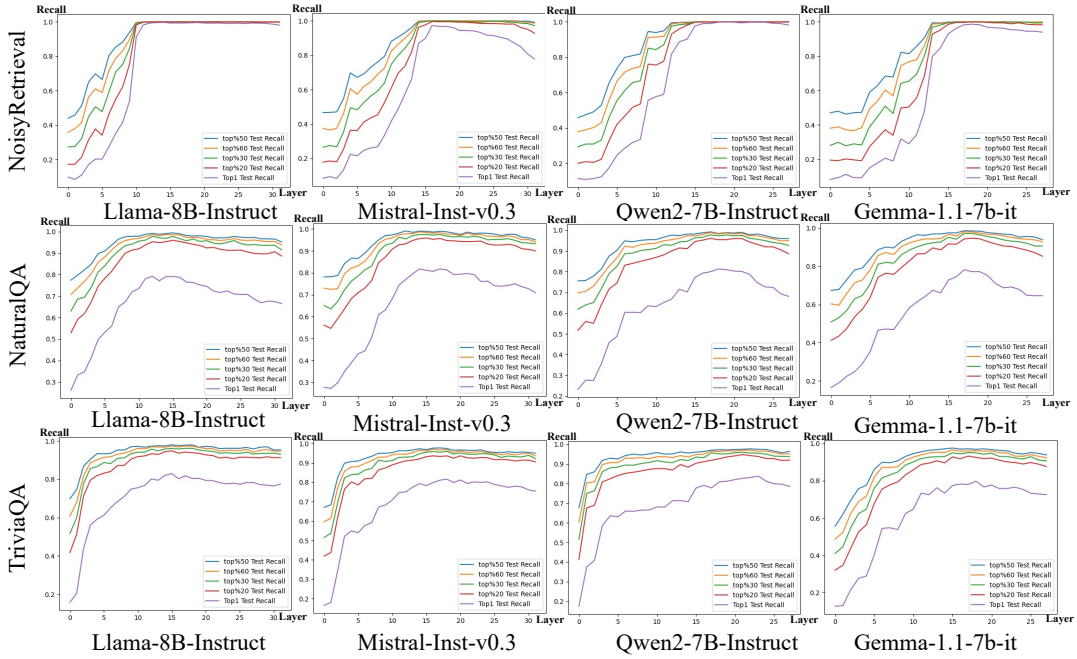

Figure 2: This figure shows the recall for positive input of the linear prober when attached to different layers of LLMs. The y-axis represents the recall value, while the x-axis indicates the index of LLM layers. Layer 0 corresponds to the first layer of the model. The top four experiments were conducted on NoisyRetrieval, the middle section on NaturalQA, and the bottom four on TriviaQA. "Top-1 recall" indicates that only the input with the highest score is predicted as positive. "Top 20% recall" means that inputs with scores in the top 20% from the linear prober are predicted as positive. Similar interpretations apply for "top 30% recall", "top 50% recall", and "top 60% recall"

Step 2: Noise Discrimination and Chunk Dropping. In this step, we leverage LLMs' inherent noise discrimination capability shown earlier to discriminate noisy chunks and drop them. We employ the previously described linear prober as the discriminator, attached to the lower layers of the LLM. Specifically, we use a logistic regression model as the prober and, based on our previous study, we attach it to layer 13. The prober predicts a score for each chunk, and we drop all chunks with a prediction score lower than a specific threshold. It's not necessary for the linear prober to predict with 100% accuracy whether a segment is noisy or not. As long as it can filter out most noisy segments, we can expect good downstream performance. Hence, in our experiments, we keep chunks with the top 30% predicted scores.

Step 3: Continue the Forward Pass with Retained Chunks. To obtain the final prediction after noise reduction with the linear prober, we perform a complete forward pass with the remaining segments. We combine all retained chunks and feed them into the LLM in a new forward pass to get the final output. Our approach requires slightly more than one forward pass through the LLM but involves less manipulation of the LLMs. Considering the parallel processing of the initial inputs, the prober applis at early layers, and the largely reduced input length for the final prediction, our approach does not enforce efficiency burden at the inference time.

3 Experiments

In this section, we introduce experimental settings first. Then we show the effectiveness of the proposed END and provide further analysis.

3.1 Experimental Settings

3.1.1 Dataset

We primarily tested the proposed method on question-answering tasks.

NoisyRetrieval. This synthetic task requires retrieving the passkey of an item characterized by five attributes: *NAME*, *MATERIAL*, *COLOR*, *BRAND*, and *ITEM*. Each instance contains a positive segment (which includes the correct answer) and 12 noisy negative segments that serve as distractors. To generate these negative segments, we control the number of shared attributes between the distractor and the positive segment, creating 5 diffi-

Model	Task	Subtask	RAG	LLM-Discrim	**END**	Prober Recall
Llama3-8b-Instrcut	NoisyRetrieval(Acc)	level_0	1.0	1.0	**1.0**	1.0
		Level_1	1.0	1.0	**1.0**	1.0
		Level_2	0.95	1.0	**1.0**	1.0
		Level_3	0.51	0.98	**0.98**	1.0
		Level_4	0.07	0.20	**0.68**	0.99
	NatualQA (F1)	Weak	69.4	**70.69**	69.8	1.0
		Hard	58.18	58.72	**61.47**	0.97
	TriviaQA (F1)	Hard	**37.83**	37.6	37.34	0.96
Qwen2-7b-inst	NoisyRetrieval(Acc)	level_0	1.0	0.96	**1.0**	1.0
		Level_1	1.0	0.96	**1.0**	1.0
		Level_2	1.0	0.96	**1.0**	1.0
		Level_3	0.81	0.96	**0.98**	1.0
		Level_4	0.52	**0.89**	0.79	0.99
	NaturalQA (F1)	Weak	**70.18**	60.09	67.8	0.96
		Hard	55.37	54.70	**59.07**	0.95
	TriviaQA (F1)	Hard	35.71	36.40	**36.93**	0.92
Mistral-Inst-v0.3	NoisyRetrieval(Acc)	level_0	1.0	0.93	**1.0**	1.0
		Level_1	1.0	0.93	**1.0**	1.0
		Level_2	0.96	0.93	**1.0**	1.0
		Level_3	0.64	0.93	**0.99**	1.0
		Level_4	0.25	**0.90**	0.78	0.99
	NaturalQA (F1)	Weak	**61.76**	48.39	57.53	1.0
		Hard	**48.58**	45.66	47.88	0.94
	TriviaQA (F1)	Hard	35.63	**37.32**	37.23	0.95

Table 2: The performance of the proposed END. For NoisyRetrieval, we use exact match accuracy as the metric. For NaturalQA and TriviaQA, we use F1 as the metric. We also listed the performance for the linear prober in the Chunk Dropping stage (Prober Recall).

culty levels of noise (from `Level_0` to `Level_4`). Here, `Level_4` is the most challenging, as the distractor shares 4 attributes with the positive segment, whereas `Level_0` is the easiest, with no shared attributes. For example, consider the question, "What is the passkey of Jack's Green Wooden Samsung Phone?" The evidence is: "the password of **Jack's Green Wooden Samsung Phone** is 12345." A `Level_0` distractor might be: "the password of Luke's Red Metal LG Laptop is 54321," and a `Level_4` distractor might be: "the password of Jack's Green Metal Samsung Phone is 41415." More details about this task and additional examples can be found in Appendix E.

Natural QA. This question answering dataset is from DPR (Karpukhin et al., 2020), where each instance has positive and negative segments retrieved in advance. Each instance has negative segments of two noise level 'Weak' and 'Hard' according to the similarity score. Hard negative segments can distract LLMs more easily because they have larger similarity scores.

Trivia QA. This dataset is also from DPR. Similar to Natural QA, each instance in trivia QA has positive and negative segments retrieved in advance. Following the settings from previous work, each instance has the negative segments with one noise level as 'Hard'.

3.1.2 Baselines

To ensure a fair comparison across scenarios with varying noise levels and context lengths, we establish two baselines:

RAG serves as a baseline, simulating the full retrieval-augmented generation (RAG) pipeline. A typical RAG pipeline consists of three main stages: retrieval, reranking, and prediction. During the reranking stage, retrieved chunks or segments are embedded, and similarity scores are computed to reorder them. The top-ranked chunks are retained and passed to the model for prediction. We didn't do real RAG for the proposed END can be attached the prompt after RAG and further reduce noise. To simulate a real-world RAG scenario, we combine

positive segments (containing the answer) with negative segments (distractors) as the post-reranking inputs for prediction. For example, the datasets from DPR (e.g., NatureQA, TriviaQA), inherently include negative chunks that have already been filtered and ranked based on similarity scores. This makes our simulation particularly reasonable, as it accurately reflects the reranking and filtering process in the RAG pipeline, ensuring alignment with real-world behavior.

LLM-Discrim acts as an **"formidable" baseline** and is supposed to surpass all methods by design (although Table 2 shows this is not the case). It has **two forward** passes. In the first forward pass, it splits the input into chunks and asks the LLMs to determine whether a segment contains the answer. Then, in the second forward pass, it requires the LLMs to provide the prediction using the remaining chunks as input. **LLM-Discrim** is **particularly effective** (Zheng et al., 2024; Fu et al., 2023), especially for hard tasks, as it explicitly leverages the ability of LLMs to discriminate input segments, setting a performance benchmark that END method does not aim to exceed.

3.2 Main Results & Analysis

The results for the proposed END are listed in Table 2. In our experiments, we retained the segments with the top 30% of linear prober prediction scores.

Effectiveness Analysis. As a result, our proposed END significantly outperforms the RAG baseline, it also achieves the best overall performance even when compared to the strong **LLM-Discrim baseline**. Besides, the superiority of END becomes more pronounced on more challenging tasks. As expected, the recall of this linear prober is sufficiently high. Although there are some instances of inferior performance, we believe these may be attributed to the F1 metric used for evaluation. For example, in the case of *NaturalQA (Weak)* with `Mistral-Inst-v0.3`, the recall is nearly perfect (0.998), yet the F1 score is still lower than anticipated. This discrepancy suggests that while our method is highly effective at identifying relevant information, the overall performance metric may not fully capture this advantage in certain scenarios.

Efficiency Analysis. While END appears to require an additional forward pass due to the extra probing and Chunk Dropping, it actually still processes a significantly reduced input during its second forward pass compared to the **RAG** baseline, owing to the substantial chunk elimination. Moreover, in comparison with the strong **LLM-Discrim** baseline, the Chunk Dropping stage consumes less than half of a forward pass. Assuming quadratic complexity for LLM operations, and considering an input segmented into 10 chunks, each of length L, using `Llama-3-8B-Instruct` (32 layers) as the backbone LLM, and dropping 70% (7) of chunks at layer-13, we can approximate the computational cost as follows: **RAG**: $\Theta((10L)^2) = \Theta(100L^2)$ **LLM-Discrim** (assuming 2 chunks retained): $\Theta((10L)^2 + (2L)^2) = \Theta(104L^2)$ **END** (30% chunks retained): $\Theta(\frac{13}{32}(10L)^2 + (3L)^2) = \Theta(49\frac{5}{8}L^2)$ This demonstrates that END achieves superior efficiency compared to both baselines. Real-world wall-clock time analysis also support its efficiency (Appendix D).

Backbone	NoisyRetrieval	NaturalQA	TriviaQA
Llama3-rand	0.60	0.71	0.66
Llama3	0.99	0.91	0.90
Qwen2-rand	0.64	0.71	0.67
Qwen2	0.95	0.88	0.86
Mistral-rand	0.60	0.70	0.66
Mistral	0.93	0.89	0.89

Table 3: Linear prober F1 scores on test sets comparing pre-trained and randomly initialized models. Llama3 refers to `Llama-3-8B-Instruct`, Qwen2 to `Qwen2-7B-Inst`, and Mistral to `Mistral-Inst-v0.3`. The -rand suffix indicates randomly initialized versions of these models with preserved embedding layers.

3.3 Analysis of the Linear Prober

The efficacy of END is heavily influenced by the performance of the linear prober. A more accurate and precise linear prober results in shorter final inputs, enabling LLMs to generate predictions more efficiently and to better understand the contexts. This section presents a comprehensive analysis of the linear prober, demonstrating that, as hypothesized in the introduction, it unveils LLMs' inherent capabilities and implicit reasoning processes when addressing factual queries based on the input.

Linear Prober's Effectiveness Is not from of fitting. To verify that the linear prober's discriminative capability does not stem from the fitting process of training the prober itself, we conducted experiments using randomly initialized LLMs as the backbone while maintaining the original embedding layer to preserve semantic meaning. The performance comparison between these randomly

Task	[S|C|Q]	[S|Q|C]	[Q|S|C]	[S|C]
NoisyRetrieval	1.0	0.98	0.96	0.25
NaturalQA	0.97	0.83	0.84	0.36
TriviaQA	0.96	0.70	0.68	0.36

Table 4: The top 30% recall of the linear prober (layer-15) across different prompt formats for Llama-3-8B-Instruct.

initialized LLMs and their pre-trained counterparts is presented in Table 3. The substantial performance gap of the linear prober demonstrates that this implicit discriminative ability originates from the pre-trained LLMs rather than training the linear prober with a LLM as a general feature extractor.

The linear prober is robust to prompt variations. Previous research has shown that LLMs can be sensitive to prompt formats (Sclar et al., 2023). To explore the robustness of the linear prober, we analyze its performance across four different prompt formats: [S|C|Q], [S|Q|C],[Q|S|C], and [S|C], where 'S' represents the system prompt, 'Q' the question, and 'C' the context. The linear prober is trained using the [S|C|Q] format with `Llama-3-8B-Instruct` as the backbone model. As shown in Table 4, although the linear prober is only trained on [S|C|Q], it generalizes well across [S|C|Q], [S|Q|C] and [Q|S|C] formats. Such kind of generalization strongly supporting the idea that LLMs are inherently prepared to answer a question once the context is fully presented.

In contrast, the performance degradation on the [S|C] format suggests that the linear prober relies on more than just distinguishing between noise and informative input—it extracts the model's implicit understanding of the input in relation to the question. The specific prompts used for each dataset and format are provided in Appendix F.

The distinguishing capabilities of large and small models arise at similar layers. We conduct probing experiments on both Llama3-70B-Instruct and `Llama3-8B-Instruct` to investigate how their distinguishing abilities evolve across layers. Figure 3 shows the performance of linear probers applied to both models. Interestingly, despite the significant difference in size—Llama3-70B-Instruct has 80 layers, while Llama3-8B-Instruct has only 32 layers—the performance of the linear prober saturates around similar layers for both models, specifically between layers 10 and 15. This finding suggests that the key distinguishing capabilities in language models is not solely dependent on model size but is instead strongly tied to the depth. Even in the larger `Llama3-70B-Instruct`, these capabilities arise early in the network, around the same layer range as in the much smaller `Llama3-8B-Instruct`. This consistency across models of varying sizes implies that these distinguishing features are fundamental properties of the model's internal representations and layer-wise progression.

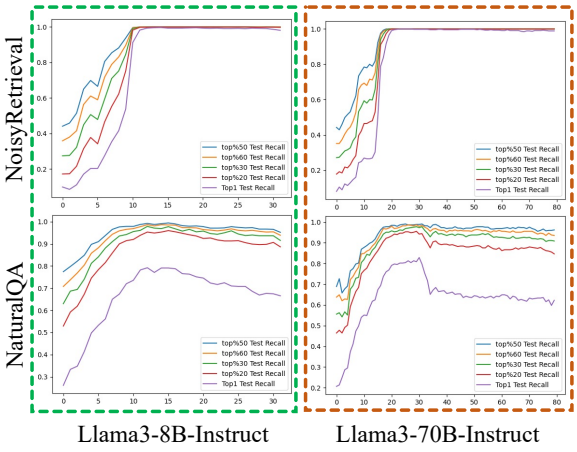

Figure 3: The performance for the linear prober with `Llama-3-8B-Instruct` and `Llama-3-70B-Instruct` as the backbone.

The implicit discrimination ability is as strong as explicitly requiring LLMs to discriminate the input. We conducted an experiment where we replaced the task query in the prompt with a distinguishing query (Appendix F.4) that directly asks the LLM to determine whether the input is relevant to the question, similar to the first forward pass of the baseline method **LLM-Discrim**. We then attached the linear prober to the representations generated from this input to evaluate whether this explicit requirement would lead to improved performance. The results, shown in Figure 4, are surprising. While the performance with the distinguishing query appears slightly better than with the task query, the overall performance of the linear prober remains nearly the same. This outcome supports our earlier hypothesis: regardless of whether the model is explicitly instructed to distinguish relevant input, LLMs seem to perform this discrimination step implicitly before answering questions related to context.

4 Related Work

Enhancing LLMs' Long Context Ability. Zhang et al. (2024) found that positional encod-

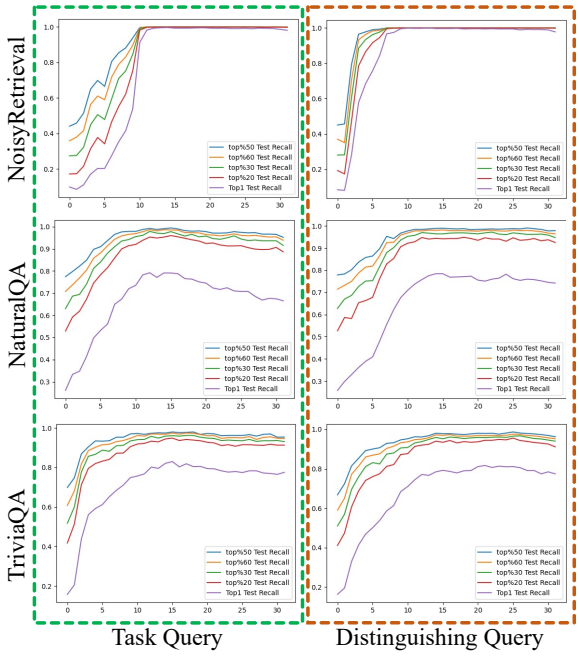

Figure 4: The performance for the linear prober with `Llama-3-8B-Instruct` as the backbone. The left three figures use the task query in the prompts. The right three figures use a distinguishing query in prompts.

ing decay weakens self-attention on relevant parts and proposed modifying key attention head dimensions to improve LLMs' performance on noisy input. Hsieh et al. (2024) addressed positional bias by introducing a calibration mechanism that subtracts baseline attention from a dummy document, preventing overemphasis on input boundaries. He et al. (2024) tackled this via instruction tuning to help LLMs focus on target information. Beyond data-centric approaches, Wu et al. (2024) applied contrastive loss in post-training, enhancing robustness by retrieving similar document pairs.

Retrieval Augmented Generation (RAG). Different from directly feeding all collected texts to query the LLMs, RAG first performs chunking and then retrieves the most related chunks across all candidate texts (Lewis et al., 2020; Asai et al., 2023). Although RAG can significantly boost the performance of LLMs, it still suffers from the distraction problem as we will show below. This is because there are still only a few useful chunks among all the retrieved chunks, even though they all have scores considering their relation to the query.

Prompt/Context Compression The original prompts always contain many useless tokens, and this line of works automatically prunes redundant tokens in the prompts so as to reduce the input length. They leverage tailored metrics such as self-information to keep important information (Li et al., 2023). Together, these studies demonstrate that strategic compression of input contexts can lead to computational savings without sacrificing accuracy (Jiang et al., 2023a,b; Xu et al., 2023).

Separate Context Processing. This approach decomposes long inputs into separate parts for individual processing, including agent collaboration and parallel context processing. Multi-agent systems iteratively exchange results to refine predictions (Zhao et al., 2024; Team, 2024; Lee et al., 2024). Parallel processing extracts and merges useful information (Yen et al., 2024) or employs trainable selection mechanisms to determine predictions (Merth et al., 2024). Some methods apply KV-cache compression post-segmentation to reduce noise and context length (Kim et al., 2024).

Linear Probing in Natural Language Processing. Linear probing has been used to extract knowledge from models, including LLMs (Gurnee and Tegmark, 2023). Recent NLP studies show that applying it to the first generated token can aid trust-related tasks like hallucination detection (Slobodkin et al., 2023). Unlike prior work, we perform layer-wise probing and find that LLMs assess input informativeness at an early stage. Moreover, while existing methods probe task-related concepts, our approach explores unrelated concepts.

5 Conclusion

In this paper, we introduced Early Noise Dropping (END), a novel method to improve Large Language Models' (LLMs) performance when processing noisy or irrelevant context. END effectively identifies and removes noisy input chunks early in the LLM pipeline, improving performance across various tasks. The method demonstrates versatility by working well across different LLM architectures without requiring fine-tuning. By reducing unnecessary computation on irrelevant information, END offers both performance and efficiency gains.

While END shows significant promise, future work could explore more advanced chunking strategies and dynamic thresholding techniques. The prober could be enhanced with additional trainable parameters or greater complexity.

6 Limitations

We acknowledge that the linear prober's generalization across tasks remains challenging (generalization analysis can be found in Appendix C.1). While fitting a simple linear regression requires minimal data, this limitation could potentially be addressed through the implementation of a more sophisticated probing mechanism. Additionally, the mechanism of LLMs' internal behavior is still not entirely clear, requiring further investigation. Besides, due to our limited computational resources, we did not conduct extensive experiments on large LLMs such as Llama-3-70B. However, the prober still performs well (Appendix C.2) at this scale and appears to save more computation, considering that huge models have more layers.

References

Cem Anil, Esin Durmus, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Nina Rimsky, Meg Tong, Jesse Mu, Daniel Ford, et al. 2024. Many-shot jailbreaking. *Anthropic, April*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

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

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

gkamradt. Llmtest$_n eedleinahaystack$.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, LiuYiBo LiuYiBo, Qianguosun Qianguosun, Yuxin Liang, Hao Wang, Enming Zhang, and Jiaxing Zhang. 2024. Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13628–13642, Bangkok, Thailand. Association for Computational Linguistics.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long T Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, et al. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. *arXiv preprint arXiv:2406.16008*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. 2024. Infinipot: Infinite context processing on memory-constrained llms. *arXiv preprint arXiv:2410.01518*.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*.

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

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Thomas Merth, Qichen Fu, Mohammad Rastegari, and Mahyar Najibi. 2024. Superposition prompting: Improving and accelerating retrieval-augmented generation. In *International Conference on Machine Learning*.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. 2023. The curious case of hallucinatory (un) answerability: Finding truths in the hidden states of over-confident large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3607–3625.

Qwen Team. 2024. Generalizing an llm from 8k to 1m context using qwen-agent.

Zijun Wu, Bingyuan Liu, Ran Yan, Lei Chen, and Thomas Delteil. 2024. Reducing distraction in long-context language models by focused learning. *arXiv preprint arXiv:2411.05928*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

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

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2588–2610, Bangkok, Thailand. Association for Computational Linguistics.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. 2024. Found in the middle: How language models use long contexts better via plug-and-play positional encoding. *arXiv preprint arXiv:2403.04797*.

Jun Zhao, Can Zu, Hao Xu, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longagent: Scaling language models to 128k context through multi-agent collaboration. *arXiv preprint arXiv:2402.11550*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

A Dataset Statistics

The detailed statistics of the dataset are as follows. After filtering out instances without a sufficient number of negative chunks, the dataset used in our study has the following distribution:

Dataset	Train Set (Prober)	Test Set
NQ	1000	2653
Trivia	1000	2619
NoisyRetrieval (each difficulty level)	1000	2000

Table 5: Dataset statistics

B Prober Training Details

The prober was trained using the following steps:

1. For each question (instance), collect negative-positive segment pairs.
2. Label negative segments as 0 and positive segments as 1.
3. Feed these segments, along with their corresponding questions, into the model to extract intermediate representations.
4. Split all instances, along with their corresponding negative and positive segments, into training and test sets.
5. Train a simple sigmoid-based linear prober (logistic regression).

The training set for each prober on each dataset consists of 1,000 instances. Further details about these data can be found in the previous section.

C More results of the Prober's performance

C.1 Generalization of The Linear Prober

We tested cross-task generalization with two settings: The prober is trained with NQ, but tested on Trivia; The prober is trained with Trivia, but tested on NQ. As shown in Figure 5, we found that: Cross-task performance is decent (about 0.9 recall), but not perfect (1.0 recall). It still shows the existing ability of implicit discrimination. Meanwhile, the performance shows some instability.

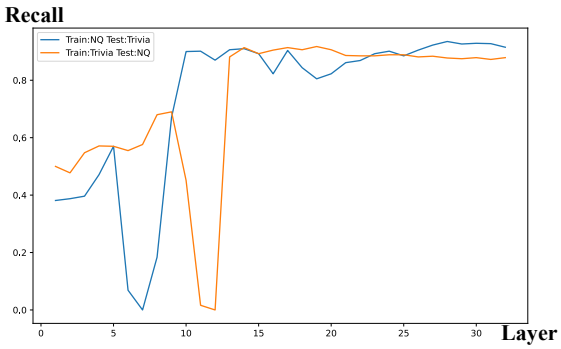

Figure 5: The cross-task performance for the linear prober with `Llama-3-8B-Instruct` as the backbone.

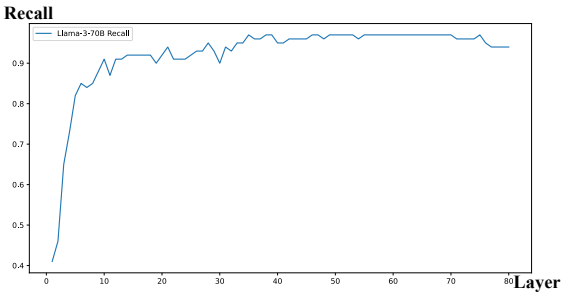

Figure 6: The performance for the linear prober with `Llama-3-70B-Instruct` as the backbone.

C.2 Prober on Larger LLMs

We conducted probing experiments with Llama-3-70B, which has 80 layers (Figure 6). The results show that the optimal probing layer is similar to that of much smaller models (e.g., the 8B model), typically around layers 10–15. This finding suggests that END remains beneficial for larger models, as the layer-wise performance does not shift significantly and we just need to run the a few layers. The portion of the first partial forward in the whole cost becomes lower (13/32 –> 13/80). However, if LLMDISCRIM is used instead, the cost of full forward passes would increase considerably due to the greater number of layers and parameters per layer.

D Real-World Efficiency Analysis

In real-world LLM deployment, flash-attention (Dao, 2024), which significantly reduces the quadratic time complexity of self-attention, is widely used. In this section, we demonstrate that our proposed **END** method continues to yield efficiency benefits with the flash-attention kernel. Table 6 presents the wall-clock inference times under the same experimental settings as our other

Model/Method	Chunk Number	4k	8k	16k	32k
Llama3-8B-END	**10**	**1.4810s**	**1.315s**	**1.887s**	**3.109s**
Llama3-8B-END	**20**	**1.508s**	**1.314s**	**1.871s**	**3.052s**
Llama3-8B	NA	1.226s	1.846s	2.996s	6.173s
Llama3-70B-END	**10**	**5.049s**	**6.069s**	**OOM**	**OOM**
Llama3-70B	NA	6.416s	9.998s	OOM	OOM

Table 6: Wall-clock inference times (in seconds) with the flash-attention kernel for different models/methods and sequence lengths. **xxx-END** indicates models equipped with the proposed **END** method; "Chunk Number" denotes the number of segments for each input.

experiments: 30% of the chunks are retained for the second forward pass, and the first forward pass is executed in batch across layers 1 to 13.

Our results indicate that when the input is long (i.e., when the computation cost is dominated by the input length) or when the model is large, the proposed **END** method clearly outperforms the standard approach of processing the entire sequence in a single forward pass. Although the observed efficiency improvement does not precisely match the predictions from our complexity analysis, these findings nonetheless highlight the practical benefits of our method in real-world scenarios.

E NosiyNeedle

NoisyRetrieval (Synthetic)

The original "Needle-in-a-Haystack" style task (gkamradt) such as passkey retrieval (Mohtashami and Jaggi, 2023) is too easy. The passkey retrieval task is to retrieve a simple passkey like "The passkey is 12345". Its contexts are some meaningless texts such as repeated "The grass is green, the sky is blue". An example looks like:

```
The grass is green, the sky is blue. The grass
    is green, the sky is blue. The passkey is
    12345. The grass is green, the sky is blue.
    The passkey is 12345. The grass is green,
    the sky is blue. The grass is green, the sky
     is blue. The grass is green, the sky is
    blue. What is the passkey?
```

This is far from real cases, and any retriever can perfectly find the answer segments. Hence, we design the **NoisyRetrieval** task as a noisier version. We set five noise levels (level_0 - level_4) for segments, by controlling five attributes in the target sentence: `[Name]`, `[Material]`, `[Color]`, `[Brand]`, `[Item]`. For example:

> [Jack's] Password to his [Green] [Wooden] [Benz] [phone] is 34512:

With this sentence as the answerm, some examples are:

- **level_0**: Random but meaningful texts from some papers.
- **level_1**: Random but meaningful texts from some papers + a sentence with one attribute the same as the target:

  > [Paul's] Password to his [red] [golden] [Apple] [**phone**] is 51233

- **level_4**: Random but meaningful texts from some papers + a sentence with four attributes the same as the target:

  > [**Jack's**] Password to his [**Green**] [**Wooden**] [**Benz**] [laptop] is 12345

An example input for **NoisyRetrieval**:

```
[other contexts] [Jack's Password to his Green
    Wooden Apple phone is 12345] [other contexts
    ] [Paul's Password to his Green Wooden Apple
     phone] [other contexts] [Jack's Password to
     his red glass Benz phone is 11145] [other
    contexts] [Jack's Password to his yellow
    glass Apple phone is 32525] [Noise] ...
Question: What is Jack's Password to his Green
    Wooden Benz phone?
Answer:
```

For this task, each positive chunks contains the answer inserted randomly into some random texts from random essays. Each negative chunk include distractor content inserted randomly into random texts from random essays. For input context construction, to ensure sufficient difficulty and realism, the chunk containing the answer is placed in the middle of the context. For example, with 10 negative chunks and 1 positive chunk, the positive chunk is positioned as the 6th chunk within the context.

E.1 Attribute Value

The possible values of all attributes are listed below:

```
Name = ["John", "Emma", "Alex", "Sophia", "
    Michael", "Olivia", "Liam", "Ava", "Noah", "
    Isabella", "Ethan", "Mia", "Mason", "
    Charlotte", "William", "Amelia", "James", "
    Harper", "Benjamin", "Evelyn"]
Color = ["red", "blue", "green", "yellow", "
    black", "white", "purple", "orange", "pink",
     "brown", "gray", "navy", "teal", "maroon",
    "olive", "silver", "gold", "turquoise", "
    lavender", "coral"]
Item = ["bag", "watch", "phone", "laptop", "
    headphones", "sunglasses", "shoes", "jacket
    ", "camera", "tablet", "wallet", "backpack",
     "earbuds", "smartwatch", "keyboard", "mouse
    ", "speaker", "charger", "fitness tracker",
    "power bank"]
Brand = ["Apple", "Samsung", "Nike", "Adidas", "
    Sony", "Gucci", "Microsoft", "Dell", "LG", "
    Bose", "Lenovo", "Asus", "Logitech", "Prada
    ", "Canon", "Nikon", "Fitbit", "Fossil", "
    JBL", "Anker"]
Material = ["leather", "aluminum", "plastic", "
    glass", "titanium", "silicone", "ceramic", "
    fabric","wood", "rubber", "nylon", "
    polyester", "cotton", "wool", "denim", "
    suede", "velvet", "cork"]
```

F Used Prompts

The prompts used in the probing stage are listed below. [S|C|Q] is the default prompt format without specifying.

F.1 NaturalQA

[S|C|Q] Setting:

```
System Prompt: "You are given some pieces of a
    story, which can be either a novel or a
    movie script, and a question. Answer the
    question as concisely as you can, using a
    single phrase if possible. Do not provide
    any explanation."

Prompt Template:
"{SYSTEM}
Pieces of the story:
{CONTEXT}
Now, if you can find the required information,
    answer the question based on the story as
    concisely as you can, using a single phrase
    if possible. Do not provide any explanation.

Question: {QUERY}

Answer:"
```

[S|Q|C] Setting:

```
System Prompt: "You are given some pieces of a
    story, which can be either a novel or a
    movie script, and a question. Answer the
    question as concisely as you can, using a
    single phrase if possible. Do not provide
    any explanation."

Prompt Template:
"{SYSTEM}

Now, if you can find the required information,
    answer the question based on the story as
    concisely as you can, using a single phrase
    if possible. Do not provide any explanation.
     Question: {QUERY}

Pieces of the story:
{CTX}

Answer:"
```

[Q|S|C] Setting:

```
System Prompt: "You are given some pieces of a
    story, which can be either a novel or a
    movie script, and a question. Answer the
    question as concisely as you can, using a
    single phrase if possible. Do not provide
    any explanation."

Prompt Template:
"Now, if you can find the required information,
    answer the question based on the story as
    concisely as you can, using a single phrase
    if possible. Do not provide any explanation.

Question: {QUERY}

{SYSTEM}

Pieces of the story:
{CTX}

Answer:"
```

[S|C] Setting:

```
System Prompt: "You are given some pieces of a
    story, which can be either a novel or a
    movie script, and a question. Answer the
    question as concisely as you can, using a
    single phrase if possible. Do not provide
    any explanation."

Prompt Template:
"
{SYSTEM}

Pieces of the story:
{CTX}

Answer:"
```

F.2 NoisyNeedle

[S|C|Q] Setting:

```
System Prompt: "There are information about a
    passkey hidden in input. Please remember it
    ."

Prompt Template:
"{SYSTEM}
Part of the input:
{CTX}
Remeber your task: according to all previous
    input, if there is the answer to: {QUERY},
    answer the question
"
```

[S|Q|C] Setting:

```
System Prompt: "There are information about a
    passkey hidden in input. Please remember it
    ."

Prompt Template:
"{SYSTEM}
Remeber your task: according to all previous
    input, if there is the answer to: {QUERY},
    answer the question
Part of the input:
{CTX}
"
```

[Q|S|C] Setting:

```
System Prompt: "There are information about a
    passkey hidden in input. Please remember it
    ."

Prompt Template:
"Remeber your task: according to all previous
    input, if there is the answer to: {QUERY},
    answer the question
{SYSTEM}
Part of the input:
{CTX}
"
```

[S|C] Setting:

```
System Prompt: "There are information about a
    passkey hidden in input. Please remember it
    ."

Prompt Template:
"{SYSTEM}
Part of the input:
{CTX}

"
```

F.3 Trivia

[S|C|Q] Setting:

```
System Prompt: "Answer the question based on the
     given passage. Only give me the answer and
    do not output any other words."

Prompt Template:
"{SYSTEM}

The following are some passages:
{CTX}
Now, if you can find the required information,
    answer the question based on those passages.
     Do not provide any explanation.

Question: {QUERY}

Answer:"
```

[S|Q|C] Setting:

```
System Prompt: "Answer the question based on the
     given passage. Only give me the answer and
    do not output any other words."

Prompt Template:
"{SYSTEM}
Now, if you can find the required information,
    answer the question based on those passages.
     Do not provide any explanation.

Question: {QUERY}

The following are some passages:
{CTX}

Answer:"
```

[Q|S|C] Setting:

```
System Prompt: "Answer the question based on the
     given passage. Only give me the answer and
    do not output any other words."

Prompt Template:
"Now, if you can find the required information,
    answer the question based on those passages.
     Do not provide any explanation.

Question: {QUERY}

{SYSTEM}

The following are some passages:
{CTX}

Answer:"
```

[S|C] Setting:

```
System Prompt: "Answer the question based on the
     given passage. Only give me the answer and
    do not output any other words."

Prompt Template:
"{SYSTEM}

The following are some passages:
{CTX}

Answer:"
```

F.4 Distinguishing Prompt

The distinguishing prompt directly requires the LLM to discriminate whether the context contains answers to a given question.

```
Prompt Template:
"
You are a helpful assistant. I will give you a
    query and some contexts. Please help me with
     my question about the given query and
    contexts.

Question: {QUERY}

The following are some passages:
{CTX}

Do the given contexts contain the answers to the
     given question? Use 'Yes' or 'No' to answer
     it. Do not provide any explanation.
```

```
Answer:"
```