Can Few-shot Work in Long-Context? Recycling the Context to Generate Demonstrations

Anonymous ACL submission

Abstract

Despite recent advancements in Large Language Models (LLMs), their performance on tasks involving long contexts remains sub-optimal. In-Context Learning (ICL) with few-shot examples may be an appealing solution to enhance LLM performance in this scenario; However, naïvely adding ICL examples with long context introduces challenges, including substantial token overhead added for each few-shot example and context mismatch between the demonstrations and the target query. In this work, we propose to automatically generate few-shot examples for long context QA tasks by *recycling* contexts. Specifically, given a long input context (1-3k tokens) and a query, we generate additional query-output pairs from the given context as few-shot examples, while introducing the context only once. This ensures that the demonstrations are leveraging the same context the target query while only adding a small number of tokens to the prompt. We further enhance each demonstration by instructing the model to *explicitly* identify the relevant paragraphs before the answer, which improves performance while providing fine-grained attribution to the answer source. We apply our method on multiple LLMs and obtain substantial improvements on various QA datasets with long context, especially when the answer lies within the middle of the context. Surprisingly, despite introducing only single-hop ICL examples, LLMs also successfully generalize to multi-hop long-context QA using our approach.

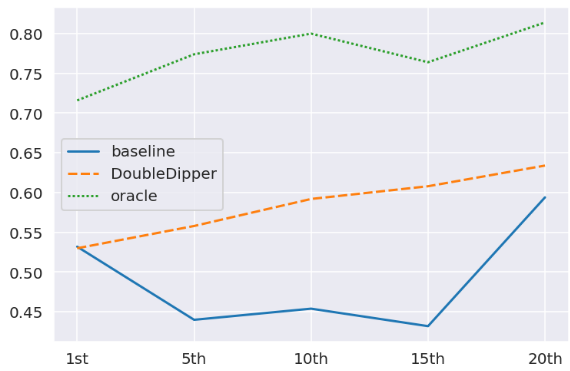

Figure 1: Performance of Gemini Pro (v1.0) on a sample of the Lost-in-the-middle dataset (Liu et al., 2023). The X-axis is the position of the relevant passage in the context. The baseline (blue line) displays a U-shaped curve, performing well only when the relevant passage is at the beginning or end of the input. The oracle (green line) shows significant performance gain when the relevant passage ID is provided in the prompt, showing that the main bottleneck is the identification of supporting evidence(s). DOUBLEDIPPER (our method, orange line) flattens this U-shaped trend.

1 Introduction

Long contexts are prevalent in various domains, ranging from legal documents and scientific articles to lengthy reports and novels. These may consist of a single extensive document or multiple passages, typically retrieved through specific retrieval mechanisms (e.g., RAG; Lewis et al., 2020).

Yet, while Large Language Models (LLMs) have demonstrated impressive capabilities in a variety of tasks including answering questions requiring one or multiple reasoning steps, they often struggle to answer simple questions when faced with long contexts. Despite substantial engineering efforts (Chen et al., 2023) to extend the context window of LLMs to extremely long inputs (32k and even 1M tokens), these models continue to struggle with much shorter inputs, comprising only a few thousand tokens.

In order to answer questions from long inputs, models should *implicitly* identify relevant information segments and then reason over these segments to formulate an answer. It has been shown that LLMs struggle when the relevant information is buried in the middle of the context (Liu et al., 2023) or obscured by numerous irrelevant details (Levy et al., 2024). Our analysis (illustrated in Figure 1) identifies the identification of relevant information

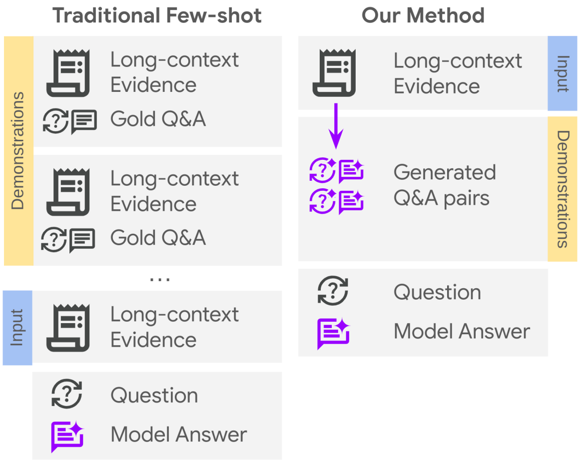

Figure 2: Comparison of traditional In-Context-Learning (ICL) and our new method. In traditional ICL (left), each example comprises a possibly lengthy context, accompanied by a query and an answer, typically derived from the training dataset. Conversely, our approach (right) simplifies each example to just a question and an answer, both of which are generated directly from the provided input context.

as a primary performance bottleneck of current models in long contexts.

In this work, we introduce a novel method to enhance the QA performance of LLMs in long input setups (to allow direct comparisons across a wide swath of models, we limit "long context" here to 1-3k tokens). Our approach, termed DoubleDipper, leverages LLMs' In-Context Learning (ICL) capability and is based on two principles. First, instead of typical ICL, where each few-shot example is standalone with a separate length context and a question-answer (QA) pair, we propose to *recycle* the given input context and automatically generate few-shot examples from this context. Specifically, we randomly select a few paragraphs from the given input context and generate QA pairs for each passage. These generated QAs serve as demonstration examples and are placed between the input context and the target input question. Figure 2 illustrates the differences between the traditional ICL with few-shot examples and DoubleDipper. Second, we further enrich each ICL example to instruct the model to *explicitly* identify the paragraph(s) containing the relevant information before generating the answer. This can be regarded as a structured Chain of Thought that incentivizes the model to pinpoint relevant information before reasoning, an essential capability for long-context processing.

By generating few-shot demonstrations from various sections of the input context while instructing the model to identify relevant passages, DoubleDipper encourages the model to develop deeper reading comprehension skills specific to the given input evidence. This, in turn, allows the model to answer subsequent queries with higher accuracy. DoubleDipper presents several advantages. In terms of efficiency, since each example does not include its own input context, our method adds to the original prompt a minimal number of tokens, resulting in a substantially cheaper inference than traditional ICL. Additionally, recycling the same context for ICL demonstrations ensures that the few-shot examples refer to the same domain as the input question, thus obviating the need for external retrieval processes. Finally, DoubleDipper generates answers with attribution to relevant paragraphs, improving the model's lookup ability and offering transparency, which substantially simplifies human evaluation (Slobodkin et al., 2024).

We applied DoubleDipper to a variety of LLMs, both commercial (Gemini Pro and Gemini Ultra; Team et al., 2023) and open-source (Llama (Touvron et al., 2023), Mistral (Jiang et al., 2023) and Gemma (Team et al., 2024)), and evaluate it on various QA datasets with long inputs, including common multi-hop QA datasets. Our experiments demonstrate that with only 3 self-generated few-shot examples, DoubleDipper consistently outperforms the baseline on our evaluation set. In addition, for some models, DoubleDipper enhances the robustness to the position of the relevant information within the text. Interestingly, while our few-shot examples focus on single-paragraph answers, DoubleDipper generalizes well to multi-hop QAs where the answer requires information from multiple passages.

2 Background

Challenges in Long Context for Language Modeling. LLMs have been well-documented to struggle when input length grows (An et al., 2023), and especially so when it exceeds input lengths seen during training (Anil et al., 2022). Various methods have been proposed to advance long-context capabilities: Architectural, e.g., to augment the embedding layer to cleverly extrapolate to unseen lengths (Vaswani et al., 2017; Press et al., 2021; Caciularu et al., 2022); and via data, e.g., to incorporate longer inputs and more challenging long-context scenarios into training (He et al., 2023;

Chen et al., 2023). However, this challenging problem stubbornly remains in competitive models today (Liu et al., 2023; Bishop et al., 2023; Levy et al., 2024). In contrast to the above methods, DOUBLEDIPPER does not involve training or architectural changes.

In documenting and exploring LLM performance in long-context settings, many different benchmarks targeting it have been proposed, such as Scrolls and Zero-Scrolls (Shaham et al., 2022, 2023), Loogle (Li et al., 2023), LongBench (Bai et al., 2023), inter alia. We describe the most relevant evaluation benchmarks used in this work in Section 4.

In-Context Learning The area of in-context learning (ICL) is a class of prompting techniques where demonstrations are added to the prompt in order to steer or improve model behavior (Min et al., 2022a). Typically, in-context learning involves hand-crafted demonstrations (Song et al., 2022), automatic retrieval of demonstrations from a larger set (Paranjape et al., 2023), or instructing the model to perform various tasks one after another in a pipeline (Gao et al., 2022). Recent improvements in long-context capabilities of LLMs have also had effect on improving the yield from in-context learning by simply using more short-length demonstrations (Agarwal et al., 2024).

While such methods are widely used for their effectiveness (Brown et al., 2020b; an Luo et al., 2024), they remain under-explored in settings of long-context. The reason is simple: If the context is already extremely long, adding additional demonstrations comparable in length to the input context will likely amplify the existing limitations of long context handling (Li et al., 2024). In this work, we tackle these challenges and present a novel ICL method for long contexts.

3 DOUBLEDIPPER

Recall that our work focuses on the task of question answering (QA) with long input context comprising multiple paragraphs. In addition to the answer, we aim to identify the supporting paragraphs in order to provide attribution. Formally, given a long input text D composed of n paragraphs $D = \{p_1, p_2, ..., p_n\}$ and a question q, the goal is to generate the answer a and identify the set(s) of paragraphs that support the answer $S = \{s_1, ..., s_k\}$. The number of the supporting paragraphs is not known in advance and can be one or more.

We describe DOUBLEDIPPER, an efficient method for improving the performance of large language models (LLMs) when dealing with long contexts. The core principles of DOUBLEDIPPER involve: (1) recycling the input context to automatically generate few-shot examples, and (2) "teaching" the model via in-context learning (ICL) to *explicitly* pinpoint the supporting paragraphs before generating the answer.

Figure 3 illustrates DOUBLEDIPPER. Starting with the input paragraphs D, we initially select k paragraphs at random (e.g., paragraphs 15, 5, and 17, for $k := 3$). For each chosen paragraph, we prompt the model to formulate a question that pertains to the specific paragraph, accompanied by an appropriate answer (for further details on prompt specifications, refer to Appendix A). Each generated QA pair is directly associated with its origin paragraph, enabling us to assemble the following structured in-context demonstration, shown as the DOUBLEDIPPER block in Figure 3:

$$\text{Question} : q_i$$
$$\text{Evidence} : p_i$$
$$\text{Answer} : a_i$$

Here, p_i indicates the index of the paragraph associated with the QA pair (q_i, a_i). Given a test question q, we then form a QA prompt by concatenating the original input context D, the compiled demonstrations and q. The model first generates the one or more indices of the supporting paragraph(s), followed by the answer.

Unlike traditional few shot examples that instruct the model about a specific *task*, DOUBLEDIPPER aims to coach the model on how to "handle" the specific input context. This is achieved by guiding the model to explicitly localize relevant information before generating the answer. By randomly sampling multiple paragraphs from the input, DOUBLEDIPPER guarantees that the ICL demonstrations involve reading different parts of the context, allowing the model to better comprehend the input text. Beyond improving the performance of the QA task, instructing the model to provide the supporting paragraphs is valuable on its own as it offers transparency and substantially eases human evaluation (Slobodkin et al., 2024).

DOUBLEDIPPER offers several advantages. First, as each example in the demonstration consists only of a question, an answer and the ID of relevant passage, the number of added tokens due

Input

Instructions: [...]

[0]: The Parc botanique de Neuvic (6 hectares) is a botanical garden located in Neuvic-Sur-L'Isle [...]
⋮
[5] Santa Cruz de las Flores is the name of a town located south of Tlajomulco de Zuñiga, in the state of Jalisco, Mexico. It has been called Xochitlan, meaning "Place of Flowers"
[6]: Graft-De Rijp is a former municipality in the Netherlands, in the province of North Holland.
⋮
[15]: The Jardin Botanique de l'Université de Strasbourg (3.5 hectares) is a botanical garden at 28 rue Goethe, Strasbourg, Bas-Rhin, Alsace, France. It is open daily without charge.
⋮
[17]: Marquette is an unincorporated community in [...], located on Illinois Route 29, east of De Pue.
[18]: The capital and seat of the provincial government is Haarlem, and the province's largest city is the Netherlands' capital Amsterdam. The King's Commissioner of North Holland is Johan Remkes, serving since 2010.

DoubleDipper

See below a few examples:
Question: Is there an admission fee for the Jardin botanique de l'Université de Strasbourg?
Evidence: [15]
Answer: No, it is open daily without charge.
Question: What is the name of the town located south of Tlajomulco de Zuñiga?
Evidence: [5]
Answer: Santa Cruz de las Flores
Question: What is the name of the community that is west of Marquette?
Evidence: [17]
Answer: De Pue

Task

Who was in charge of the state where Graft-De Rijp is located?
Evidence: [6, 18]
Answer: Johan Remkes

Figure 3: Example of DOUBLEDIPPER applied to the MuSique dataset. Given 20 passages as input, DOUBLEDIPPER randomly selects 3 passages (specifically passages 15, 5, 17) and automatically generates a question-pair for each one. As each QA is associated with its respective paragraph, we form the demonstrations to instruct the model to identify the relevant passage(s) and the correct answer.

to the extra demonstrations is minimal, leading to a low additional cost and computation. Furthermore, by reusing the same context to generate demonstrations, our approach guarantees that all few shot examples are derived from the exact same domain as the input query (Rubin et al., 2022).

Finally, we observe that, although the QA pairs in the demonstration are confined to individual paragraphs, the actual query q may require reasoning over multiple paragraphs (i.e., multi-hop QA). Surprisingly, LLMs can generalize from DOUBLEDIPPER *local* examples to these complex, *global* questions and successfully generate indices to multiple paragraphs (see Section 5).

4 Experiments

Datasets We apply our method to various datasets, each presenting its own domain-specific challenges. We selected these datasets because the supporting paragraphs are also annotated. Overall our evaluation set include 5.5K instances, with statistics of each dataset given in Table 1.

The Lost-in-the-middle dataset (Liu et al., 2023) includes examples from NaturalQuestions-Open (Kwiatkowski et al., 2019; Lee et al., 2019). Each instance consists of twenty Wikipedia passages, with only one passage containing the answer to the query. The remaining passages are distractors that are lexically similar but do not contain the answer. To assess the robustness of large language models (LLMs) to the position of relevant information, Liu et al. (2023) evaluated cases where the relevant passage appeared in positions 1, 5, 10, 15, and 20. Following their methodology, we sampled 500 instances for each position, resulting in a total of 2,500 instances.

Dataset	# Instances	Avg. # tokens
Lost-in-the-middle	2,500	2,815
FLenQA	1,500	3,225
HotpotQA	500	1,646
2Wiki	500	1,222
MuSiQue	500	2,549

Table 1: Evaluation datasets in our experiments. The average number of tokens is computed according to Gemma's tokenization of the simple prompt.

FLenQA (Levy et al., 2024) is a benchmark that includes simple questions with answers of either "True" or "False" based on two key sentences.

FLenQA includes three subtasks. The first subtask is Monotone Relations (MonoRel), where each instance asks whether a transitive relation between two entities holds based on the context (e.g., "Is X younger than Y?" based on the sentences "X is younger than Z" and "Z is younger than Y"). The second subtask, People In Rooms (PIR), involves one key sentence indicating that a person is in a specific room and another key sentence describing a property of this room. The question asks whether the person is in a room with the described property. The final subtask is Simplified Rule Taker (SRT), based on RuleTaker (Clark et al., 2020). Each instance consists of a logical rule, two sentences each introducing a fact, and a question over the rule and facts. For each subtask, FLenQA includes contexts with varying lengths, from 50 to 3,000 tokens, by simply adding irrelevant text, demonstrating consistent performance degradation with increased input length. In our experiments, we sampled 250 instances for each subtask with input lengths of 2,000 and 3,000 tokens, leading to a total of 1,500 instances for FLenQA.

In addition, we evaluate our method on common multi-hop QA benchmarks. We sampled 500 instances from each of the following datasets: HotPotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). In all these datasets, the input text includes multiple passages, and models need to perform at least two steps of reasoning over different passages in order to answer the question.

Models We apply our method to a variety of models, both commercial and open-source. The commercial models include Gemini-1.0-Ultra and Gemini-1.0-Pro (Team et al., 2023). The open-source models we tested are Llama-2-7b-chat, Llama-2-13b-chat, Llama-2-70b-chat (Touvron et al., 2023), Gemma-2b-it (v1.1), Gemma-7b-it (v1.1)(Team et al., 2024) and Mistral-7b-instruct (v0.2)(Jiang et al., 2023). Details about models' size and context window are shown in Table 2.

For few-shot generation, we conducted two distinct experiments in order to analyze the effect of different question generation models. In the first experiment, dubbed DOUBLEDIPPER (Self), we employed the same LLM for both generating the demonstrations and answering to the query. In the second experiment, named DOUBLEDIPPER (PaLM 2), we generated a single set of few-shot examples using PaLM 2 (Anil et al., 2023) and then supplied these examples to various LLMs. This approach allowed us to assess the performance consistency across different models when provided with the same set of demonstrations.

	Size	Context Window
Gemini v1.0	Pro	32k
	Ultra	32k
Llama-2	7B	4k
	13B	4k
	70B	4k
Mistral	7B	8k
Gemma	2B	8k
	7B	8k

Table 2: Models used with DOUBLEDIPPER.

Baselines We compare DOUBLEDIPPER to the vanilla baseline, which takes as input the entire context D and the query q and generate only the answer a.

Evaluation We evaluate each dataset with the original evaluation metrics. Namely, we report Accuracy for Lost-in-the-middle (Liu et al., 2023) and FLenQA (Levy et al., 2024), and Token F1 for HotPotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020) and MuSique (Trivedi et al., 2022).

In addition to the task's accuracy, we also evaluate the performance of the identification of the supporting paragraph(s), by computing the F1 score on the predicted set of supporting passages compared to the ground truth (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022).

Implementation Details We randomly select three passages from the input (see Section 6 for an analysis of the number of self-generated demonstrations on the performance), each containing at least two sentences, and ask the model to generate five QA pairs per passage (see Appendix A for the exact prompt). We then randomly select a single QA pair for each passage to form the few shot demonstrations. For all experiments, including few-shot generation and question-answering, we use a temperature setting of 0.

5 Results

Table 3 presents the QA performance of our two variants of DOUBLEDIPPER, namely DOUBLEDIPPER (Self) and DOUBLEDIPPER (PaLM 2) on our evaluation set. We report the performance of the

	Avg.	Lost	PIR	MonoRel	SRT	HotPotQA	2Wiki	MuSique
Gemini Pro	39.4	49.0	73.0	30.8	**54.4**	28.1	18.6	22.1
+ DoubleDipper (Self)	50.0	**58.4**	85.6	28.6	52.2	56.6	36.5	32.0
+ DoubleDipper (PaLM 2)	**54.6**	**58.4**	**87.6**	**40.6**	52.8	**62.2**	**42.8**	**38.0**
Gemini Ultra	53.9	53.0	**81.2**	**30.8**	**66.0**	61.6	42.6	42.3
+ DoubleDipper (Self)	53.1	**57.2**	73.4	20.6	54.6	68.9	51.5	**45.6**
+ DoubleDipper (PaLM 2)	**54.7**	56.8	78.6	25.0	56.6	**69.1**	**52.6**	44.2
Gemma-2b-it (v1.1)	35.9	24.0	53.4	49.8	**49.6**	34.6	**28.1**	11.6
+ DoubleDipper (Self)	35.5	**25.5**	66.0	54.2	49.2	24.7	18.6	10.5
+ DoubleDipper (PaLM 2)	**41.2**	23.6	**75.8**	**57.8**	49.4	**38.2**	**28.1**	**15.4**
Gemma-7b-it (v1.1)	27.2	7.1	52.2	49.8	**48.6**	12.3	16.7	3.4
+ DoubleDipper (Self)	47.6	**45.8**	92.6	74.2	45.4	40.4	22.4	12.4
+ DoubleDipper (PaLM 2)	**51.3**	44.0	**94.8**	**74.8**	45.2	**49.6**	**32.1**	**18.4**
Mistral-7b-instruct (v0.2)	45.8	**61.3**	70.6	68.0	**49.0**	36.3	21.7	13.9
+ DoubleDipper (Self)	48.9	60.4	**96.2**	**84.6**	42.6	28.3	17.8	12.6
+ DoubleDipper (PaLM 2)	**52.8**	58.9	95.6	82.8	44.4	**43.1**	**26.4**	**18.4**
Llama-2-7b-chat	31.6	42.0	46.8	**39.0**	**37.4**	32.7	15.1	8.2
+ DoubleDipper (Self)	33.3	**45.6**	64.2	37.2	35.4	23.3	17.8	9.2
+ DoubleDipper (PaLM 2)	**38.5**	43.9	**67.6**	36.8	35.2	**39.2**	**30.1**	**16.4**
Llama-2-13b-chat	32.4	**51.2**	54.2	40.6	**43.8**	18.2	10.8	7.9
+ DoubleDipper (Self)	34.9	46.9	66.0	**44.6**	38.4	22.7	19.3	8.4
+ DoubleDipper (PaLM 2)	**38.3**	47.6	**71.0**	37.8	36.4	**32.8**	**32.2**	**10.4**
Llama-2-70b-chat	45.3	**61.1**	66.6	76.4	**50.2**	33.8	17.7	11.0
+ DoubleDipper (Self)	46.1	57.2	85.6	76.6	44.8	27.8	20.3	10.7
+ DoubleDipper (PaLM 2)	**51.6**	54.2	**88.2**	**78.0**	47.0	**44.5**	**34.2**	**15.0**

Table 3: Results of the baseline, DoubleDipper (Self) and DoubleDipper (PaLM 2) on various QA datasets.

supporting paragraphs prediction in Appendix B. On average, our straightforward method, which automatically generates few-shot examples from the provided input context, outperforms the baseline, often by a significant margin (e.g., +15.2 points for Gemini Pro, +7 points for Mistral Instruct, +6.3 points for Llama2 70B). Comparing the two variants, DoubleDipper (PaLM 2) generally yields better results than DoubleDipper (Self), indicating that higher-quality question-answer (QA) pairs in the few-shot examples enhance performance. Nonetheless, DoubleDipper (Self) already surpasses the baseline by a considerable margin on its own in most benchmarks. Notably, while DoubleDipper produces simple QAs answerable from a single paragraph, it demonstrates strong generalization across diverse QA formats (except SRT), including PIR and MonoRel, which involves True/False questions, and multi-hop QA datasets (HotPotQA, 2Wiki, and MuSique), where reasoning over multiple paragraphs is required.

Furthermore, following (Liu et al., 2023), Figure 4 shows the performance of the tested LLMs for both the baseline and DoubleDipper (Self) on our sample of the Lost-of-the-middle dataset, according to the position of the document that contains the answer. For Gemini Pro, Gemini Ultra, Gemma 2B, Gemma 7B, Llama 7B, the performance curve for DoubleDipper consistently surpasses the baseline across nearly all document positions. This reveals that beyond improving performance, DoubleDipper can make the model more robust to the position of the relevant document. For Mistral, Llama 2 13B and Llama 2 70B, Figure 4 shows that DoubleDipper performs on par or better than the baseline when the relevant document is in the middle of the input, but still underperforms the baseline at the beginning and the end. This variation can likely be attributed to the inherent biases of LLMs towards the beginning and end of inputs, while adding in context demonstrations mitigates this bias.

6 Analysis

Few-shot vs. Zero-shot In order to assess the effectiveness of the few shots in DoubleDipper, we prompt the model in a zero-shot setting to *explicitly* specify the ID of the relevant passage(s) before generating the answer. The results of the QA task are detailed in Table 4 and the performance in identifying supporting paragraphs is reported in Appendix B.

For all open-source models, this zero-shot prompting strategy leads to a significant decline in QA performance compared to the baseline (see Table 3), which merely prompts the model for the answer (e.g., -32.3 points for Gemma 2B, -9.2 points for Llama 7B, -4.2 points for Mistral). This drop is likely because retrieving relevant passages makes the task more complex compared to standard QA,

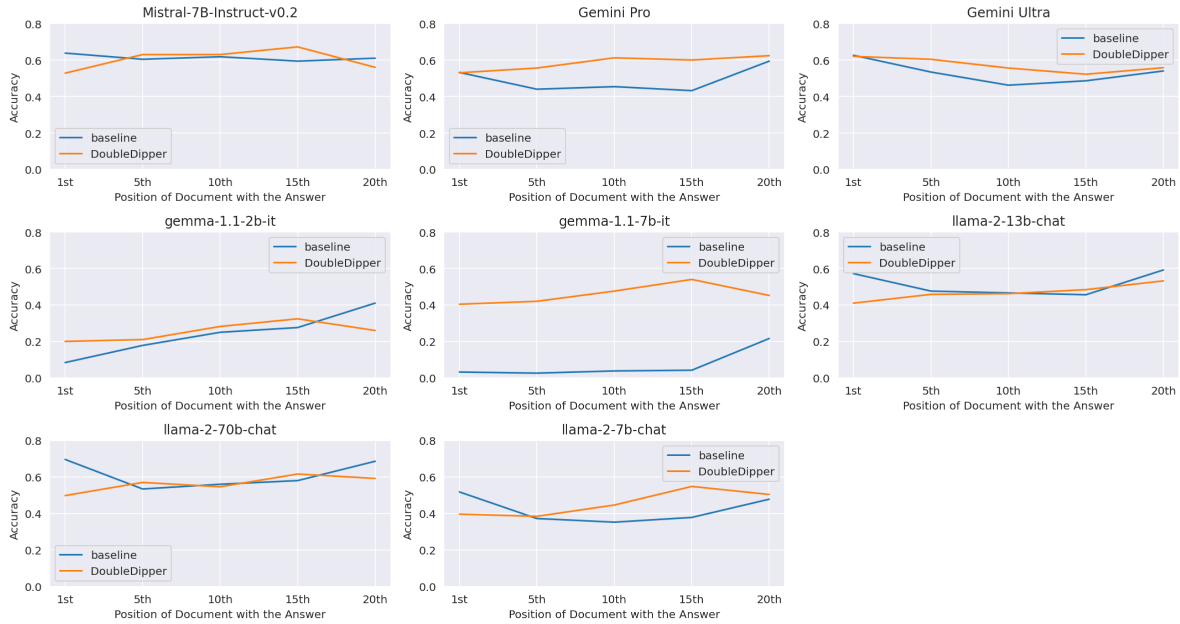

Figure 4: Performance (accuracy) of the models on our sample of the Lost-in-the-middle dataset (Liu et al., 2023) according to the position of the document that contains the answer.

		Avg.	Lost	PIR	MonoRel	SRT	HotPotQA	2Wiki	MuSique
Gemini Pro	Zero-shot	50.0	59.7	77.4	38.8	**54.2**	51.2	34.6	34.1
	DOUBLEDIPPER	**54.6**	**58.4**	**87.6**	**40.6**	52.8	**62.2**	**42.8**	**38.0**
Gemini Ultra	Zero-shot	**56.5**	56.2	76.8	**34.8**	**62.2**	68.3	52.4	**44.9**
	DOUBLEDIPPER	54.7	**56.8**	**78.6**	25.0	56.6	**69.1**	**52.6**	44.2
Gemma-2b-it (v1.1)	Zero-shot	3.6	6.4	4.2	5.4	0.0	3.3	4.9	0.7
	DOUBLEDIPPER	**41.2**	**23.6**	**75.8**	**57.8**	**49.4**	**38.2**	**28.1**	**15.4**
Gemma-7b-it (v1.1)	Zero-shot	7.5	2.6	36.0	5.4	0.0	3.9	3.3	1.0
	DOUBLEDIPPER	**51.3**	**44.0**	**94.8**	**74.8**	**45.2**	**49.6**	**32.1**	**18.4**
Mistral-7b-instruct (v0.2)	Zero-shot	41.9	**59.0**	77.4	67.4	**51.8**	13.7	12.3	11.8
	DOUBLEDIPPER	**52.8**	58.9	**95.6**	**82.8**	44.4	**43.1**	**26.4**	**18.4**
Llama-2-7b-chat	Zero-shot	22.4	29.2	45.0	29.2	33.4	9.8	7.9	2.4
	DOUBLEDIPPER	**38.5**	**43.9**	**67.6**	**36.8**	**35.2**	**39.2**	**30.1**	**16.4**
Llama-2-13b-chat	Zero-shot	14.0	8.0	43.0	3.0	37.0	5.2	0.3	1.2
	DOUBLEDIPPER	**38.0**	**36.7**	**71.4**	**36.6**	**50.0**	**27.4**	**27.4**	**16.8**

Table 4: Comparison of DOUBLEDIPPER to prompting models in a zero-shot setting to provide explicitly the supporting paragraphs, then generate the answer.

as models are not typically trained to perform such "retrieval" tasks. This is further supported by the poor performance in supporting passages identification shown in Appendix B, where F1 scores are often close to zero.

Despite these challenges, DOUBLEDIPPER's self-generated demonstrations systematically improve performance for all open-source models on both QA and supporting passages identification across all datasets. On average, Mistral's QA performance improves by 10.9 points, and its identification of relevant paragraphs increased by 30.5 F1 score. This aligns with previous research suggesting that models can learn new tasks through in-context learning (ICL).

Comparison to traditional ICL To analyze the performance gain of DOUBLEDIPPER compared to the traditional In-Context-Learning, we prompt a representative model from each family (Gemini Pro, Gemma 2B, Llama 7B, and Mistral) with few-shot examples from the same dataset, each example including its own context and question-answer pair.

We report the results in Table 5. For all models, DOUBLEDIPPER achieves a higher performance than the traditional ICL. Interestingly, for Gemma 2B and Llama 7B, the traditional ICL not only underperforms DOUBLEDIPPER but it also significantly hurts performance compared to the baseline (e.g., -14.3 for Gemma 2B and -14.9 for Llama 7b). This performance drop might be because each

		Avg.	Lost	PIR	MonoRel	SRT	HotPotQA	2Wiki	MuSique
Gemini Pro	ICL	48.6	**59.1**	69.8	30.0	**52.8**	55.6	41.0	32.0
	DoubleDipper	**54.6**	58.4	**87.6**	**40.6**	**52.8**	**62.2**	**42.8**	**38.0**
Gemma-2b-it (v1.1)	ICL	21.6	0.2	28.2	26.4	13.0	**39.7**	**39.1**	4.3
	DoubleDipper	**41.2**	**23.6**	**75.8**	**57.8**	**49.4**	38.2	28.1	**15.4**
Llama-2-7b-chat	ICL	16.7	0	13.2	13.2	14.2	37.0	**38.2**	0.7
	DoubleDipper	**38.5**	**43.9**	**67.6**	**36.8**	**35.2**	**39.2**	30.1	**16.4**
Mistral-7b-instruct	ICL	52.1	**68.5**	79	67.8	39.2	**49.4**	**36.7**	**24.1**
	DoubleDipper	**52.8**	58.9	**95.6**	**82.8**	**44.4**	43.1	26.4	18.4

Table 5: Comparison of traditional In-Context Learning (ICL) where each demonstration example comprises a full text, a question and an answer to DoubleDipper where the demonstrations contain only question-answer pairs, automatically generated on the same input text.

	$k = 3$	$k = 5$	$k = 10$
Gemini Pro	54.6	54.1	54.7
Gemini Ultra	54.7	54.1	53.8
Gemma-2b-it (v1.1)	41.2	40.8	41.0
Gemma-7b-it (v1.1)	51.3	50.2	49.5
Mistral-7b-instruct (v0.2)	52.8	52.8	52.4
Llama-2-7b-chat	33.3	32.5	31.5
Llama-2-13b-chat	34.9	34.4	33.0
Llama-2-70b-chat	51.6	51.1	51.5

Table 6: Average performance on our evaluation set with various numbers of self-generated few shot demonstrations (k) in DoubleDipper.

demonstration in the traditional ICL setup comprises a few thousand tokens, which exacerbates the models' existing challenges with processing long-range dependencies.

As mentioned in Section 3, DoubleDipper promotes also efficiency by adding to the original prompt only a few extra tokens, leading to a significantly cheaper inference than the traditional ICL.

How many examples are needed? In Table 6, we explore the impact of varying k, the number of self-generated few-shot examples in DoubleDipper to 3, 5, and 10. Our analysis reveals no significant differences in performance across these variations, while 3 self-generated examples are sufficient to improve performance. This finding is in line with previous work (Brown et al., 2020a; Min et al., 2022b).

DoubleDipper without identification of supporting paragraphs To ablate the second principle in DoubleDipper, namely the explicit identification of the supporting paragraphs before generating the answer, we prompt Gemini Pro and Gemini Ultra with self-generated few shot examples that comprise only question-answer pairs (without instructing the model to retrieve the relevant passage(s)). Gemini Pro achieves on average 36.2 points (-18.4) and Gemini Ultra achieves 54.0 (-0.7 points). This indicates that instructing small models (e.g., Gemini Pro) with explicit prediction of supporting paragraphs achieves similar results to large models (Gemini Ultra).

Do few-shot examples really "instruct" the model to comprehend the text or provide clues to the query? To answer this question, we prompt Gemini Pro and Gemini Ultra with only the self-generated demonstrations without the input context. As expected, removing the input context leads to a huge performance drop, -31.7 for Gemini Pro and -22.8 for Gemini Ultra compared to DoubleDipper. This confirms that DoubleDipper indeed "teaches" the model how to comprehend the input.

Qualitative Analysis We manually analyze 50 prompts, with few-shot demonstrations produced by Gemini Ultra. Our review confirms that 93.5% of these self-generated QAs are correct, meaning that the question is meaningful and the answer could be found in the corresponding paragraph.

7 Conclusion

We develop DoubleDipper, a straightforward method for enhancing the performance of Question Answering with long context and providing attribution to the relevant paragraph(s) in the input. By recycling the input context to generate the few shot examples, each demonstration includes solely a question, an answer and a pointer to the relevant paragraph, without a separate context, thus effectively addressing the challenging of In-Context-Learning with long context. Experimental results show that our approach ourperforms both the vanilla LLM and the traditional ICL in various QA settings, including distractor passages in the input, True/False questions and multi-hop QA.

8 Limitations

One notable limitation of our approach is the extended inference time required for generating question-answer pairs. Future research could mitigate this issue by developing smaller, specialized models specifically tailored for QA generation.

Additionally, our evaluation set is constrained to instances that are solely in English and range between 1,000 to 4,000 tokens. Expanding the diversity of languages and token ranges could enhance the robustness and applicability of our findings.

Lastly, although we employ a strategy of randomly sampling k paragraphs from the input to ensure the model engages with varied segments of the text, we did not optimize the selection of these paragraphs. Future work could explore more strategic methods for paragraph selection to potentially enhance the efficacy and relevance of the generated examples.

References

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning. *Preprint*, arXiv:2404.11018.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *Preprint*, arXiv:2307.11088.

an Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *ArXiv*, abs/2401.11624.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *ArXiv*, abs/2207.04901.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Tachard Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Z. Chen, Eric Chu, J. Clark, Laurent El Shafey, Yanping Huang, Kathleen S. Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Michael Brooks, Michele Catasta, Yongzhou Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, C Crépy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, M. C. D'iaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fan Feng, Vlad Fienber, Markus Freitag, Xavier García, Sebastian Gehrmann, Lucas González, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, An Ren Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wen Hao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Mu-Li Li, Wei Li, Yaguang Li, Jun Yu Li, Hyeontaek Lim, Han Lin, Zhong-Zhong Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Oleksandr Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alexandra Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Marie Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniela Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Ke Xu, Yunhan Xu, Lin Wu Xue, Pengcheng Yin, Jiahui Yu, Qiaoling Zhang, Steven Zheng, Ce Zheng, Wei Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *ArXiv*, abs/2305.10403.

Yushi Bai, Xin Lv, Jiajie Zhang, Hong Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *ArXiv*, abs/2308.14508.

Jennifer A Bishop, Qianqian Xie, and Sophia Ananiadou. 2023. Longdocfactscore: Evaluating the factuality of long document abstractive summarisation. *ArXiv*, abs/2309.12455.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack

Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Avi Caciularu, Ido Dagan, Jacob Goldberger, and Arman Cohan. 2022. Long context question answering via supervised contrastive learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2872–2879, Seattle, United States. Association for Computational Linguistics.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *ArXiv*, abs/2309.12307.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *International Joint Conference on Artificial Intelligence*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, N. Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2022. Rarr: Researching and revising what language models say, using language models. In *Annual Meeting of the Association for Computational Linguistics*.

Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, Yibo Liu, Yuxin Liang, Hao Wang, Qianguo Sun, Songxin Zhang, Zejian Xie, and Jiaxing Zhang. 2023. Never lost in the middle: Improving large language models via attention strengthening question answering. *Preprint*, arXiv:2311.09198.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *ArXiv*, abs/2402.14848.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2023. Loogle: Can long-context language models understand long contexts? *ArXiv*, abs/2311.04939.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *ArXiv*, abs/2404.02060.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Rethinking the role of demonstrations: What makes in-context learning work? *ArXiv*, abs/2202.12837.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *ArXiv*, abs/2303.09014.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *ArXiv*, abs/2108.12409.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989, Singapore. Association for Computational Linguistics.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aviv Slobodkin, Eran Hirsch, Arie Cattan, Tal Schuster, and Ido Dagan. 2024. Attribute first, then generate: Locally-attributable grounded text generation. *ArXiv*, abs/2403.17104.

Yisheng Song, Ting-Yuan Wang, Puyu Cai, Subrota Kumar Mondal, and Jyoti Prakash Sahoo. 2022. A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Computing Surveys*, 55:1 – 40.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

A Prompts

Figure 5 shows the zero-shot prompt we use for generating the question-answer pairs in DOUBLEDIPPER. For the QA prompts, we use the same instructions and prompt template as the original papers (Lost-in-the-middle and FLenQA) and add a simple line for the instructions in other multi-hop QA datasets: *"Please answer the question based on the given passages below."*. For MuSique, since the dataset includes questions that are not answerable, we add the following sentence to the prompt: *"If the question can't be answered given the given passages, please write "unanswerable""*.

B Identification of Supporting Passages

Table 7 presents the F1 results of the tested models for the supporting relevant passages identification. Without any demonstration (zero-shot), all open source models achieve a poor performance, while DOUBLEDIPPER significantly improves performance (+37.8 F1 for Llama2 70B, +38 F1 for Gemma 7B, +30.5 F1 for Mistral).

```
Given the following passage, please write 5 questions that could be asked in the passage.
    The questions should include enough information so that they can be understood without
    the passage and the answer should be concise. For each question, please write also the
    short answer from the text in the following format:
Q1:
A1:
Q2:
A2:
Q3:
A3:
Q4:
A4:
Q5:
A5:
```

Figure 5: Template prompt for the generation question-answer pairs.

		Avg.	Lost	PIR	MonoRel	SRT	HotPotQA	2Wiki	MuSique
Gemini Pro	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	**54.5**	58.2	47.1	**75.9**	46.6	56.3	**58.3**	39.4
	DoubleDipper (Self)	50.0	59.7	**77.4**	38.8	**54.2**	51.2	34.6	34.1
	DoubleDipper (PaLM 2)	50.6	**60.1**	47.3	68.0	21.2	**58.8**	57.9	**41.0**
Gemini Ultra	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	**40.4**	57.4	**30.3**	**39.4**	**17.0**	52.4	43.7	42.9
	DoubleDipper (Self)	37.5	**58.6**	25.5	21.5	11.1	55.4	46.4	**44.3**
	DoubleDipper (PaLM 2)	38.4	58.0	28.0	24.6	12.0	**55.3**	**47.0**	43.7
Gemma-2b-it (v1.1)	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	DoubleDipper (Self)	23.3	**10.4**	43.1	37.5	13.5	**20.1**	13.0	**25.3**
	DoubleDipper (PaLM 2)	**25.6**	9.1	**48.7**	**48.7**	**15**	18.7	**14**	**25.3**
Gemma-7b-it (v1.1)	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	DoubleDipper (Self)	**38.0**	**39.7**	44.8	**59.8**	7.1	**41.8**	**36.5**	**36.1**
	DoubleDipper (PaLM 2)	36.9	39.2	**45.2**	58.6	**7.3**	40.1	32.8	35
Mistral-7b-instruct (v0.2)	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	11.5	6.7	0.7	2.9	0.0	17.2	18.2	34.5
	DoubleDipper (Self)	**42.0**	48.2	66.2	**60.6**	5.8	**41.7**	**29.3**	**42.2**
	DoubleDipper (PaLM 2)	41.5	**48.3**	**70.0**	58.6	**6.8**	40.0	26.0	40.6
Llama-2-7b-chat	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	0.8	0.2	0.0	0.0	3.2	0.0	0.0	2.1
	DoubleDipper (Self)	19.4	**17.5**	27.7	20.2	**5.7**	21.7	19.7	**23.3**
	DoubleDipper (PaLM 2)	**19.6**	15.8	**29.3**	**21.6**	**5.7**	**22.2**	**20.7**	22.0
Llama-2-13b-chat	Baseline	-	-	-	-	-	-	-	-
	Zero-shot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	DoubleDipper (Self)	28.8	**16.5**	51.0	**32.0**	**18.4**	29.6	22.2	31.7
	DoubleDipper (PaLM 2)	**30.3**	13.4	**51.6**	30.5	15.6	**36.4**	**27.3**	**36.9**

Table 7: Performance (F1) of supporting paragraph(s) prediction. The baseline does not predict supporting paragraphs. DoubleDipper provides a significant performance boost for all open source models, whose performance is close to 0 (except from Mistral) in the zero shot setting. The performance of the zero-shot experiment is close to zero for open source models except from Mistral, whereas DoubleDipper successfully instructs the models to retrieve the relevant paragraphs.