DOUBLEDIPPER: Improving Long-Context LLMs via Context Recycling

Anonymous ACL submission

Abstract

Despite recent advancements in Large Language Models (LLMs), their performance on tasks involving long contexts remains sub-optimal. In this work, we propose DOUBLEDIPPER, a novel In-Context-Learning method that automatically generates 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 as 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 (+16 absolute points on average across models) on various QA datasets with long context. Surprisingly, despite introducing only single-hop ICL examples, LLMs successfully generalize to multi-hop long-context QA using our approach.

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., 2023b) 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, especially when the relevant information is buried in the middle of the context (Liu et al., 2023b) or obscured by numerous irrelevant details (Levy et al., 2024).

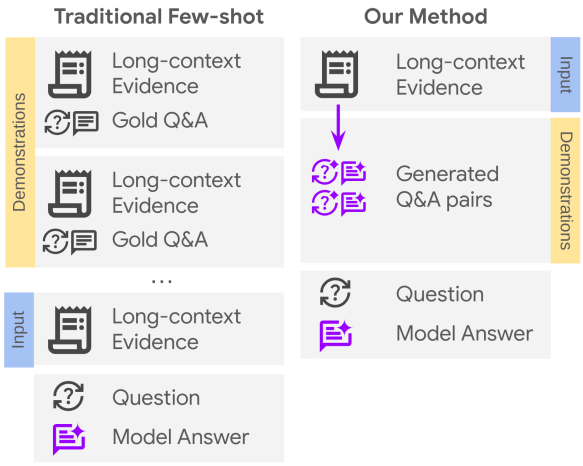

Figure 1: Comparison of traditional In-Context-Learning (ICL) and our new method. In traditional ICL (left), each example comprises a possibly lengthy context, accompanied by a query and an answer, typically derived from the training dataset. Conversely, our approach (right) simplifies each example to just a question and an answer, both of which are generated directly from the provided input context.

In-Context Learning (ICL) with few-shot examples may be an appealing solution to enhance LLM performance in long contexts. However, applying ICL in real-world scenarios without access to training data introduces significant challenges. Developers need to maintain a demonstration pool for retrieving semantically similar demonstrations to any given query (Liu et al., 2022; Rubin et al., 2022). Furthermore, ICL adds a substantial token overhead to the prompt, an issue that becomes even more pronounced with long-context inputs.

In this work, we introduce a novel method to enhance the QA performance of LLMs in long

input setups. Our approach, termed DOUBLEDIPPER, leverages LLMs' In-Context Learning capability and is based on two principles. First, instead of typical ICL, where each few-shot example is standalone with a separate lengthy 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 1 illustrates the differences between the traditional ICL with few-shot examples and DOUBLEDIPPER. Second, we enhance each ICL demonstration by *explicitly* instructing the model to identify relevant information prior to generating an answer. Explicitly identifying relevant passages 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. First, recycling the same context for ICL demonstrations ensures that the few-shot examples refer to the exact same domain as the input question, thus obviating the need for external retrieval of similar demonstrations. Also, 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. Finally, DOUBLEDIPPER generates answers with attribution to relevant paragraphs, improving the model's lookup ability and offering transparency, which substantially simplifies human evaluation (Menick et al., 2022; Gao et al., 2023; Liu et al., 2023a; Slobodkin et al., 2024).

We applied DOUBLEDIPPER to 12 LLMs, both commercial (Gemini Pro, Nano, Flash (Reid et al., 2024); GPT-4 (Achiam et al., 2023)) and open-source ranging from 2B to 70B parameters (Llama 3.1 (Dubey et al., 2024); Mistral (Jiang et al., 2023); Mixtral (Jiang et al., 2024); Gemma, (Riviere et al., 2024)). We evaluate our method on 7 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 by 16 absolute points on average across models. 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 and 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 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; Tan et al., 2024); via data, e.g., to incorporate longer inputs and more challenging long-context scenarios into training (Chen et al., 2024b; He et al., 2024; Chen et al., 2024a); via attention intervention (Hsieh et al., 2024) or by considering question likelihood as a signal for prompt reordering (Liu et al., 2024b). However, this challenging problem stubbornly remains in competitive models (Liu et al., 2023b; Bishop et al., 2024; Levy et al., 2024). In contrast to the above methods, DOUBLEDIPPER is a simple method that does not involve training or architectural changes.

Many benchmarks targetting long-context have been proposed, such as Scrolls and ZeroScrolls (Shaham et al., 2022, 2023), Loogle (Li et al., 2023), LongBench (Bai et al., 2023b), L-Eval (An et al., 2024), inter alia. The problem of designing informative and reliable benchmarks in long-context is an an active, ever-changing area of research (Goldman et al., 2024; Yen et al., 2024). We describe the most relevant evaluation benchmarks used in this work in Section 4.

In-Context Learning In-context Learning (ICL) consists of adding demonstrations to the prompt in order to steer or improve model behavior (Min et al., 2022). These demonstrations are either handcrafted (Song et al., 2022), or retrieved from a large

set of training examples (Liu et al., 2022; Rubin et al., 2022; Paranjape et al., 2023). While ICL provides a flexible approach to learn new tasks without updating parameters (Brown et al., 2020b; an Luo et al., 2024), applying ICL in real-world scenarios is challenging notably because there is no available training data for each user query.

More closely related to our work, a few recent studies propose to prompt LLMs for automatically generating in-context demonstrations for various short context tasks. For instance, Kim et al., 2022 focus on sentence classification tasks and prompt LLMs to generate full demonstrations conditioned on a label (e.g., "write a negative review") and (Chen et al., 2023a; Yasunaga et al., 2024; Li et al., 2024) generate relevant exemplars to the query for reasoning problems. While effective for reasoning tasks with short contexts, these methods are not directly applicable to long-context scenarios because LLMs would need to generate not just a question and an answer, but also an entire long context for each demonstration, which is both computationally expensive and prone to hallucination. DOUBLEDIPPER addresses these challenges by introducing a novel and efficient strategy: rather than generating entirely new contexts, it *recycles* the input context and generates only the question-answer (QA) pairs needed as demonstrations.

3 DOUBLEDIPPER

Given a long input text $\mathcal{C}$ composed of n paragraphs $\mathcal{C} = \{p_1, p_2, ..., p_n\}$ 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 2 illustrates DOUBLEDIPPER. Starting with the input paragraphs $\mathcal{C}$, 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 B). 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 2:

$$\begin{aligned} \text{Question} &: q_i \\ \text{Evidence} &: p_i \\ \text{Answer} &: a_i \end{aligned}$$

Here, p_i indicates the index of the paragraph associated with the QA pair (q_i, a_i). Given a context $\mathcal{C}$ and a test question q, we compile a list of generated demonstrations $\mathcal{D}_{\text{demo}} = (q_1, p_1, a_1, \ldots, q_k, p_k, a_k)$ to predict the output $y \sim p_\theta(y \mid \mathcal{C}, \mathcal{D}_{\text{demo}}, q)$ where the output y is the concatenation of one or more indices of the supporting paragraph(s) S and the answer a.

Unlike traditional few shot examples that instruct the model about a specific *task*, DOUBLEDIPPER aims to coach the model on how to "handle" the input context. This is achieved by guiding the model to explicitly localize relevant information before generating the answer. Also, 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 offers transparency and substantially eases human evaluation.

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 to the extra demonstrations is minimal (5%), leading to a low additional cost and computation compared to the traditional In-Context-Learning. 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).

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 includes 5.5K instances, with statistics of each dataset given in Table 1.

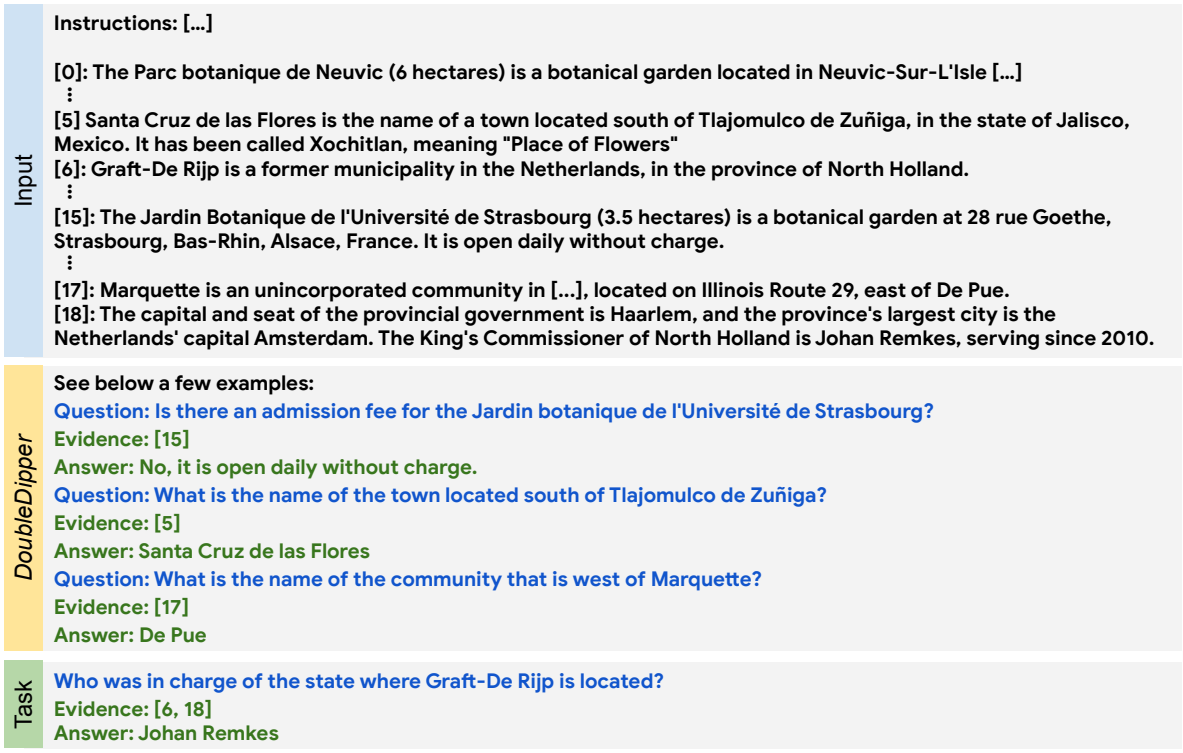

Figure 2: Example of DOUBLEDIPPER applied to the MuSique dataset. Given 20 passages as input, DOUBLEDIPPER randomly selects 3 passages (specifically passages 15, 5, 17) and automatically generates a question-pair for each one. As each QA is associated with its respective paragraph, we form the demonstrations to instruct the model to identify the relevant passage(s) and the correct answer.

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 vanilla prompt.

The Lost-in-the-middle dataset (Liu et al., 2023b) includes examples from NaturalQuestions-Open (Kwiatkowski 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 LLMs to the position of relevant information, Liu et al. (2023b) 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.

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 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, 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 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.

In addition, we evaluate our method on common multi-hop QA benchmarks. We sampled

500 instances from HotPotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), and MuSiQue (Trivedi et al., 2021). 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 DOUBLEDIPPER to a variety of models, both commercial and open-source. The commercial models include Gemini 1.5 Pro, Gemini 1.5 Flash (Reid et al., 2024) and GPT-4o-mini (Achiam et al., 2023). The open-source models we tested are Llama 3.1 8B, Llama 3.1 70B (Dubey et al., 2024), Gemma 2B (v2), Gemma 9B (v2) and Gemma 27B (Riviere et al., 2024), Mistral-7B-Instruct (v0.2) (Jiang et al., 2023), Mixtral-8x7B-Instruct (v0.1) (Jiang et al., 2024) and Mistral Nemo Instruct 2407[1].

Few-shot generation in DOUBLEDIPPER is an auxiliary task and should ideally run in an efficient time without requiring heavy resources. Therefore, in our main experiments, we employ Gemma 2B to generate the demonstrations at it is the smallest and most efficient model used in our experiments. See Section 6 for an ablation analysis of the effect of the chosen model for generating the demonstrations.

Baselines We evaluate DOUBLEDIPPER against two main baselines. The first is a vanilla baseline that takes as input the entire context $\mathcal{C}$ and the query q and generates only the answer a, a common prompting strategy in recent studies on long context (Liu et al., 2023b; Levy et al., 2024). The second baseline, *Zero-shot + Evidence Retrieval*, prompts the model in a zero-shot manner to first identify relevant passages before generating the answer, following common practices in generating with attribution (Gao et al., 2023; Slobodkin et al., 2024; Fierro et al., 2024).

Evaluation We evaluate each dataset with the original evaluation metrics. Namely, we report Accuracy for Lost-in-the-middle (Liu et al., 2023b) 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., 2021).

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., 2021).

[1] https://mistral.ai/news/mistral-nemo/

Implementation Details We randomly select three passages from the input, each containing at least two sentences, and ask the model to generate a single QA pair for each passage. See Section 6 for an analysis of the number of self-generated demonstrations on the performance

5 Results

Result 1: DOUBLEDIPPER offers a substantial performance boost. Table 2 presents the QA performance of the baseline, *Zero-shot + Evidence Retrieval* and DOUBLEDIPPER on our evaluation set. For brevity, we report the results of eight models here and four models in Appendix C in Table 7, which show similar trends. The results first show that prompting models to explicitly identify the relevant paragraphs before generating the answer (*Zero-shot + Evidence Retrieval*) leads to a performance improvement of 9.7 points on average across models over the vanilla baseline. DOUBLEDIPPER, leveraging demonstrations generated with the efficient Gemma 2B parameters, offers an additional substantial boost of 6.3 points for all models on average, **culminating in an overall improvement of 16 absolute points over the vanilla baseline**. Notably, while DOUBLEDIPPER produces simple QAs answerable from a single paragraph, *it always surpasses the baseline in multi-hop QA datasets* (HotPotQA, 2Wiki and MuSique). Likewise, DOUBLEDIPPER outperforms the baseline also on most FLenQA datasets (PIR, MonoRel and SRT), which involve synthetic True/False questions although the demonstrations in DOUBLEDIPPER are typically simple factoid questions.

While DOUBLEDIPPER exhibits strong performance overall, we observe nuanced behavior on the SRT dataset, with performance gains varying across models (improvement for Gemini, Llama 3.1 8B, Mistral 7B and Mistral Nemo). This discrepancy is likely due to the fact that SRT demands a specific type of reasoning, where models must reason over both a rule (e.g., "If X is big and X is good then X is tall") and dispersed facts (e.g., "Erin is Good" and "Erin is furry") to determine whether a statement (e.g., "Erin is tall") can be derived from the context. Finally, the Lost dataset highlights a specific characteristic of DOUBLEDIPPER: while the baseline's known positional bias (Liu et al., 2024a) masks the *average* improvement for some models, DOUBLEDIPPER substantially boost performance when relevant information appear in the

	Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemini Pro (vanilla)	60.5	24.9	95.0	97.6	64.4	46.7	71.6	23.1
Zero-shot + Evidence Retrieval	62.3	32.5	94.6	95.8	62.4	49.6	**74.8**	26.5
DOUBLEDIPPER	**70.4**	**46.8**	**97.4**	**99.0**	**79.6**	**60.9**	72.4	**36.4**
Gemini Flash (vanilla)	42.9	10.2	70.0	86.0	57.6	10.0	59.5	7.3
Zero-shot + Evidence Retrieval	58.2	30.2	78.8	90.6	65.0	44.9	**67.4**	30.2
DOUBLEDIPPER	**66.1**	**48.0**	**85.8**	**95.0**	**68.6**	**60.6**	65.0	**39.7**
Gemma 2 9B (v2) (vanilla)	44.0	11.4	74.8	81.8	55.6	13.8	61.0	9.3
Zero-shot + Evidence Retrieval	58.7	38.6	82.0	83.4	**59.4**	56.1	**64.4**	26.8
DOUBLEDIPPER	**61.2**	**41.7**	**84.0**	**95.0**	51.4	**61.2**	61.8	**33.3**
Gemma 2 27B (v2) (vanilla)	48.0	11.2	85.2	79.2	**63.8**	14.3	62.4	19.9
Zero-shot + Evidence Retrieval	59.7	34.3	90.6	87.4	59.0	51.7	**65.2**	29.8
DOUBLEDIPPER	**64.2**	**42.0**	**92.0**	**96.4**	58.6	**64.2**	62.5	**33.9**
Llama 3.1 8B (vanilla)	37.2	11.8	56.2	52.2	48.6	20.7	**63.5**	7.4
Zero-shot + Evidence Retrieval	53.1	**42.2**	71.8	65.4	50.8	55.3	62.0	24.5
DOUBLEDIPPER	**59.9**	38.7	**91.2**	**90.6**	**51.0**	**59.3**	58.1	**30.1**
Llama 3.1 70B (vanilla)	67.5	46.5	93.2	95.6	**83.2**	57.0	69.6	27.6
Zero-shot + Evidence Retrieval	71.0	57.6	97.8	97.6	81.0	62.0	**71.5**	29.5
DOUBLEDIPPER	**72.9**	**62.3**	**98.6**	**98.8**	73.0	**71.5**	66.0	**40.2**
Mistral 7B (v0.3) (vanilla)	37.4	14.1	59.2	57.4	50.2	15.8	**60.8**	4.6
Zero-shot + Evidence Retrieval	44.0	23.8	66.2	62.4	49.6	34.1	58.6	13.6
DOUBLEDIPPER	**51.0**	**28.6**	**68.4**	**88.8**	**50.6**	**43.4**	60.7	**16.7**
Mistral-Nemo	44.0	17.2	**72.6**	67.0	51.0	28.9	59.7	11.9
Zero-shot + Evidence Retrieval	46.7	29.1	59.8	67.2	51.0	39.8	60.6	19.4
DOUBLEDIPPER	**53.3**	**38.7**	58.0	**81.0**	**51.4**	**51.9**	**62.9**	**29.5**

Table 2: Accuracy of the QA task for the vanilla baseline (prompting the model to only answer the question), *Zero-shot + Evidence Retrieval* (prompting the model to explicitly identify the relevant passage(s) before generating the answer) and DOUBLEDIPPER with 3 demonstrations generated by Gemma 2 2B.

middle of the context (further elaborated in Result 3), demonstrating its efficacy in mitigating positional biases in long-context settings.

Result 2: Learning to retrieve the evidence(s) with DOUBLEDIPPER is more effective in commercial and large open source models. Table 3 presents the performance of the supporting paragraphs prediction for the *Zero-shot + Evidence Retrieval* and DOUBLEDIPPER on our evaluation set. For all commercial models, Llama 3.1 70B and the recent Mistral-Nemo-Instruct-2407, DOUBLEDIPPER predicts better the supporting paragraphs than in the zero-shot setting (+2.6 F1 for Gemini Pro, +3.4 F1 for Gemini Flash, +2.7 F1 for Llama 3.1 70B and +6.4 F1 for Mistral-Nemo-Instruct-2407). Conversely, DOUBLEDIPPER slightly hurts the performance of common open source models (e.g., -2.8 F1 for Mistral, -0.3 F1 for Llama 3.1, -1.3 F1 for Gemma 27B, etc.).

This discrepancy appears to stem from shortcut learning (Tang et al., 2023). Indeed, our demonstrations in DOUBLEDIPPER use a single evidence paragraph, and smaller models tend to overfit to this pattern, learning to retrieve only one passage even when multiple are needed. For example, under DOUBLEDIPPER, Gemma 2 9B retrieves only 1.2 paragraphs on average, compared to 2 in the "unconstrained" zero-shot setting. Larger models like Gemini Pro do not exhibit this behavior, correctly generalizing to predict an average of 2 evidence paragraphs even with single-paragraph demonstrations. This hypothesis is strongly supported by the "Lost" dataset, which requires only a single evidence paragraph. On this dataset, nearly all models, including smaller ones, benefit substantially from DOUBLEDIPPER (e.g., +12.2 F1 for Mistral 7B). Crucially, despite their suboptimal retrieval performance on multi-evidence tasks, these smaller models still produce better final answers with DOUBLEDIPPER than the zero-shot baseline, highlighting the net benefit of the approach.

Result 3: DOUBLEDIPPER makes models more robust to the position of relevant information. Following Liu et al. (2023b), Figure 3 shows the performance of Gemma 2 9B, Mixtral 8x7B and Gemini Flash for both the baseline and DOUBLEDIPPER on our sample of the Lost-of-the-middle dataset, according to the position of the document that contains the answer. See Appendix D for the performance curve of the other tested mod-

		Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemini Pro	*Zero-shot + Evidence Retrieval*	83.7	**96.7**	97.7	**97.5**	62.8	**92.1**	63.6	75.3
	DOUBLEDIPPER	**86.3**	94.4	**99.8**	97.1	**80.9**	90.0	**66.4**	**75.4**
Gemini Flash	*Zero-shot + Evidence Retrieval*	75.7	82.3	90.5	72.6	70.4	80.9	**67.3**	65.9
	DOUBLEDIPPER	**79.1**	**83.7**	**98.3**	**80.2**	**71.2**	**84.6**	66.1	**69.5**
Gemma 2 9B	*Zero-shot + Evidence Retrieval*	**61.9**	**76.7**	**69.3**	**60.3**	**43.2**	76.5	51.5	55.9
	DOUBLEDIPPER	57.0	74.2	52.5	59.0	23.1	**78.4**	**55.3**	**56.8**
Gemma 2 27B	*Zero-shot + Evidence Retrieval*	**85.1**	96.2	**97.9**	96.1	**84.1**	90.7	53.4	77.3
	DOUBLEDIPPER	83.8	**97.5**	85.4	**97.6**	74.9	**93.0**	**59.7**	**78.8**
Llama 3.1 8B	*Zero-shot + Evidence Retrieval*	**61.7**	53.2	**86.5**	**66.9**	**67.8**	63.2	41.1	53.5
	DOUBLEDIPPER	61.4	**68.9**	71.4	54.9	52.8	**73.7**	**53.0**	**54.8**
Llama 3.1 70B	*Zero-shot + Evidence Retrieval*	85.1	98.2	98	89.2	82.3	93.9	52.5	**81.6**
	DOUBLEDIPPER	**87.8**	**98.6**	**100**	**92.8**	**83.8**	**96.5**	**61.6**	**81.6**
Mistral 7B (v0.3)	*Zero-shot + Evidence Retrieval*	**46.6**	62.4	**49.8**	**46.2**	**17.5**	64.0	43.4	**43.0**
	DOUBLEDIPPER	43.8	**63.4**	33.8	42.5	4.2	**66.7**	**55.6**	40.2
Mixtral 7x8B v(0.1)	*Zero-shot + Evidence Retrieval*	**60.0**	**72.4**	69.4	**64.6**	**43.4**	**76.9**	40.6	52.4
	DOUBLEDIPPER	58.9	70.4	**81.6**	63.6	18.3	75.0	**50.4**	**53.2**
Mistral-Nemo	*Zero-shot + Evidence Retrieval*	69.7	91.1	85.0	76.9	35.9	88.8	39.8	70.3
	DOUBLEDIPPER	**76.1**	**95.9**	**95.1**	**81.5**	**39.0**	**93.7**	**54.8**	**73.0**

Table 3: Performance (F1) of supporting paragraph(s) prediction.

	$k=1$	$k=3$	$k=5$	$k=10$
Gemini Nano	60.0	62.1	62.2	62.3
Gemini Flash	65.3	66.1	65.9	66.1
Gemma 2B (v2)	47.0	49.5	49.6	49.9
Gemma 9B (v2)	58.7	61.2	61.4	61.3
Llama 3.1	57.7	59.9	60.6	61.4
Mistral 7B (v0.3)	48.9	51.0	51.1	51.4
Mixtral 7x8B (v0.1)	49.3	52.2	51.7	52.2

Table 4: Average performance on our evaluation set with various numbers of self-generated few shot demonstrations (k) in DOUBLEDIPPER. See Appendix E.1 for the results on each evaluation dataset.

els, which show similar trends.

Overall, the performance curve for DOUBLEDIPPER consistently surpasses the baseline when the relevant information appears "in the middle" and sometimes also at the beginning and/or the end (e.g., Gemini Flash). 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. This reveals that beyond improving performance, DOUBLEDIPPER can make the model more robust to the position of the relevant document.

6 Ablation Studies

How many examples are needed? In Table 4, we explore for some models the impact of varying k, the number of self-generated few-shot examples in DOUBLEDIPPER to 1, 3, 5, and 10. On average, a single demonstration already provides an improvement over the baseline. Three demonstrations adds another boost of 2 points, while increasing the number of demonstrations to 5 and 10 leads to a marginal improvement. This finding is in line with previous work (Brown et al., 2020a; Min et al., 2022). We conclude that a small number of examples carries most of the benefit with our method, but given additional computation budget, adding more examples does carry additional minor benefit.

DOUBLEDIPPER without identification of supporting paragraphs To ablate the second core principle of DOUBLEDIPPER—the explicit identification of supporting paragraphs prior to answer generation—we prompt open-source models with self-generated few-shot examples consisting solely of question-answer pairs, without instructing the model to retrieve the relevant passage(s). These demonstrations may undermine DOUBLEDIPPER's objective by encouraging models to produce answers without grounding them in the source text. The results confirms our hypothesis: removing evidence identification consistently degrades QA performance compared to the full DOUBLEDIPPER approach. When averaging results across models and datasets, this omission leads to a substantial performance drop from 54.8 to 46.6. Detailed results are provided in Appendix E.3, Table 10.

Investigating the effect of the few-shot generator To understand the impact of the default chosen model (Gemma 2 2B) for generating the demon-

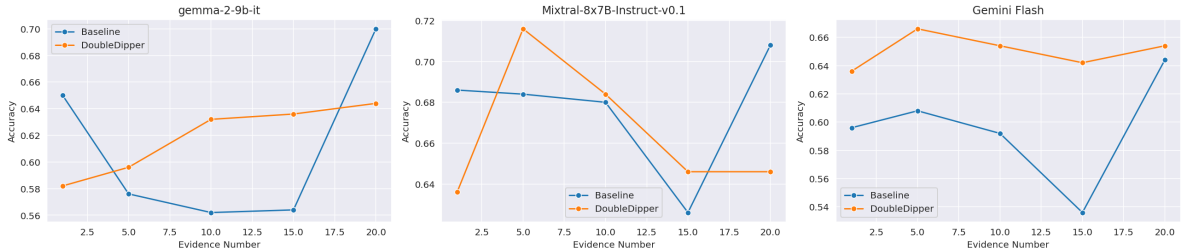

Figure 3: Performance (accuracy) of Gemma 2 9B, Mixtral 8x7B and Gemini Flash with and without DOUBLEDIPPER on our sample of the Lost-in-the-middle dataset (Liu et al., 2023b) according to the position of the document that contains the answer.

	Gemma 2B	Self	Gemini Pro
Gemini Pro	70.4	**71.6**	**71.6**
Gemini Flash	66.1	67.5	**68.1**
Gemini Nano	62.1	61.7	**62.8**
Gemma 2B	49.5	49.5	**51.0**
Gemma 9B	61.2	62.0	**63.5**
Llama 3.1	59.9	60.1	**61.5**
Mistral v0.3	51.0	49.8	**52.6**
Mixtral	52.2	49.8	**54.4**

Table 5: Average performance of DOUBLEDIPPER with different models for generating the demonstrations. See Appendix E.2 for the results on each evaluation dataset.

	Baseline	ICL	DOUBLEDIPPER
Gemma 2 9B	44.0	51.0	**61.2**
Gemma 2 27B	48.0	54.9	**64.2**
Llama 3.1 8B	37.2	40.7	**59.9**
Llama 3.1 70B	67.5	65.4	**72.9**
Mistral v0.3	37.4	42.0	**51.0**
Mixtral v0.1	42.6	45.1	**52.2**
GPT 4o mini	51.3	56.1	**60.8**

Table 6: Comparison of traditional In-Context Learning (ICL) where each demonstration example comprises a full text, a question and an answer from an external dataset to DOUBLEDIPPER where the demonstrations contain only question-answer pairs, automatically generated on the same input text.

strations, we conducted two additional experiments. The first experiment is SELF in which we use the same model for generating the demonstrations and for answering the original question. In the second experiment, we generate the demonstrations with the best LLM used in our experiments, namely Gemini Pro. The average results are reported in Table 5 and the performance for each evaluation dataset is presented in Appendix E.2. The results show that generating the demonstrations with Gemma 2 or SELF achieves similar performance, while Gemini Pro leads to a consistent increase in performance across models, indicating that future better models can improve further the performance. Please refer to Appendix E.2 for additional ablations on the few-shot generation.

DOUBLEDIPPER vs. Traditional ICL Another alternative to use ICL in practice is to prepend each QA prompt by a fixed set of QA demonstrations, each composed of a context, a question and an answer. Although the demonstrations are not necessarily from the same distribution as each user query, this common practice is helpful for task recognition and for an overview of the overall format (Min et al., 2022; Pan et al., 2023). For the demonstrations, we randomly selected 3 examples from the SQuAD 2.0 dataset (Rajpurkar et al., 2018). We compare the average results of the baseline, ICL and DOUBLEDIPPER in Table 6 and report full results in Appendix E.4 in Table 11. While ICL is effective and outperforms the baseline, DOUBLEDIPPER provides an additional performance boost of 9.5 points on average.

Qualitative analysis: Correctness of the generated QA pairs We manually analyze 150 QAs generated by Gemma 2B as demonstrations. 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 introduce DOUBLEDIPPER, a simple method for enhancing the performance of LLMs with long context. By recycling the input context to generate the demonstrations, DOUBLEDIPPER successfully addresses the practical challenges of ICL with long-context and outperforms multiple baselines in various QA settings, including distractor passages in the input, True/False questions and multi-hop QA.

8 Limitations

Our work has several limitations.

First, our current work focuses on multiple variants of QA tasks (distractor passages, True/False and multi-hop), where the demonstrations teach the models to identify specific evidence paragraph(s) and extract answers. Future research can extend our work to other QA settings (e.g., information seeking) and additional tasks (e.g., summarization).

Second, our evaluation set is constrained to instances that are solely in English and range between 1,000 to 4,000 tokens. While this demonstrates the method's effectiveness, its scalability and performance on much longer contexts (e.g., 100k+ tokens) and in multilingual settings remain open questions.

Finally, while DOUBLEDIPPER is significantly more token-efficient than traditional ICL, the initial step of generating demonstrations introduces a computational and latency overhead compared to a zero-shot baseline. This presents a trade-off between inference cost and the substantial performance gains our method provides.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and 260 others. 2023. Gpt-4 technical report.

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.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2024. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14388–14411, Bangkok, Thailand. Association for Computational Linguistics.

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.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, and 31 others. 2023a. Qwen technical report. *ArXiv*, abs/2309.16609.

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. 2023b. Longbench: A bilingual, multitask benchmark for long context understanding. *ArXiv*, abs/2308.14508.

Jennifer A. Bishop, Sophia Ananiadou, and Qianqian Xie. 2024. LongDocFACTScore: Evaluating the factuality of long document abstractive summarisation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10777–10789, Torino, Italia. ELRA and ICCL.

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, and 12 others. 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, and 12 others. 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.

Longze Chen, Ziqiang Liu, Wanwei He, Yinhe Zheng, Hao Sun, Yunshui Li, Run Luo, and Min Yang. 2024a. Long context is not long at all: A prospector of long-dependency data for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8222–8234, Bangkok, Thailand. Association for Computational Linguistics.

Wei-Lin Chen, Cheng-Kuang Wu, Yun-Nung Chen, and Hsin-Hsi Chen. 2023a. Self-ICL: Zero-shot in-context learning with self-generated demonstrations.

In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15651–15662, Singapore. Association for Computational Linguistics.

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

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. LongloRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Constanza Fierro, Reinald Kim Amplayo, Fantine Huot, Nicola De Cao, Joshua Maynez, Shashi Narayan, and Mirella Lapata. 2024. Learning to plan and generate text with citations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11397–11417, Bangkok, Thailand. Association for Computational Linguistics.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, Singapore. Association for Computational Linguistics.

Omer Goldman, Alon Jacovi, Aviv Slobodkin, Aviya Maimon, Ido Dagan, and Reut Tsarfaty. 2024. Is it really long context if all you need is retrieval? towards genuinely difficult long context nlp. *ArXiv*, abs/2407.00402.

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.

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.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14982–14995, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *ArXiv*, abs/2401.04088.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram'e, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gael Liu, and 191 others. 2025. Gemma 3 technical report. *ArXiv*, abs/2503.19786.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang goo Lee. 2022. Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator. *ArXiv*, abs/2206.08082.

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.

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.

Rui Li, Guoyin Wang, and Jiwei Li. 2024. Are human-generated demonstrations necessary for in-context learning? In *The Twelfth International Conference on Learning Representations*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Nelson Liu, Tianyi Zhang, and Percy Liang. 2023a. Evaluating verifiability in generative search engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7001–7025, Singapore. Association for Computational Linguistics.

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

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Tianyu Liu, Jirui Qi, Paul He, Arianna Bisazza, Mrinmaya Sachan, and Ryan Cotterell. 2024b. Likelihood as a performance gauge for retrieval-augmented generation. *ArXiv*, abs/2411.07773.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes. *ArXiv*, abs/2203.11147.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. 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.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319, Toronto, Canada. 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.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, and 654 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv*, abs/2403.05530.

Gemma Team Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L'eonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram'e, Johan Ferret, Peter Liu, Pouya Dehghani Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, and 176 others. 2024. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3344, Bangkok, Thailand. Association for Computational Linguistics.

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.

Sijun Tan, Xiuyu Li, Shishir G Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E. Gonzalez, and Raluca Ada Popa. 2024. LLoCO: Learning long contexts offline. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17605–17621, Miami, Florida, USA. Association for Computational Linguistics.

Ruixiang Tang, Dehan Kong, Longtao Huang, and Hui Xue. 2023. Large language models can be lazy learners: Analyze shortcuts in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4645–4657, Toronto, Canada. Association for Computational Linguistics.

H. Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2021. 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.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2024. Large language models as analogical reasoners. In *The Twelfth International Conference on Learning Representations*.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2024. Helmet: How to evaluate long-context language models effectively and thoroughly.

A Appendix

B Prompts

Figure 4 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""*.

C Additional Results

Table 7 shows the QA performance of the baseline, *Zero-shot + Evidence Retrieval* and DOUBLEDIPPER on our evaluation set when applied to Gemma 2B, Gemini Nano and GPT-4o-mini. Her

D Lost-in-the-middle

Figure 5 shows the QA accuracy of the models Gemma 2 2B, Mistral 7B, Gemini Nano and Gemini Pro on our subset of the "Lost-in-the-middle" dataset.

E Analysis

E.1 Impact of the Number of Demonstrations in DOUBLEDIPPER

Table 8 presents the results of DOUBLEDIPPER with 1, 3 (main experiment in the paper), 5 and 10 generated demonstrations. For all these experiments, the demonstrations were generated by Gemma 2 2B.

Figure 6 shows the QA accuracy of DOUBLEDIPPER on "Lost" according to the position of the relevant passage for each $k \in \{1, 3, 5, 10\}$.

E.2 Impact of the few-shot generator

Table 9 presents the detailed QA performance of all models with different models for generating DOUBLEDIPPER's demonstrations. As mentioned in the paper (Section 6), generating the demonstrations with the best model (ie. Gemini Pro) achieves the best performance overall.

Can we generate demonstrations with smaller LLMs? We next study whether smaller, more

```
Given the following TEXT, please write a simple question whose answer appears verbatim in
    the text.The question should include enough information so that it can be understood
    without the text.The answer should be concise.Please write both the question ans answer
    in the following format:
Q:
A:
TEXT: [PARAGRAPH]
```

Figure 4: Template prompt for generating the QA pairs in DOUBLEDIPPER.

	Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemini Nano (vanilla)	41.6	10.8	72.2	66.8	55.4	21.3	59.6	5.2
Zero-shot + Evidence Retrieval	56.5	32.0	82.4	82.4	**56.4**	56.7	**60.5**	25.2
DOUBLEDIPPER	**62.1**	**40.6**	**86.6**	**95.4**	56.2	**65.1**	**60.5**	**30.4**
GPT-4o-mini (vanilla)	51.3	19.0	78.6	89.6	**71.6**	23.3	67.8	9.3
Zero-shot + Evidence Retrieval	53.8	19.4	89.6	93.0	63.2	26.2	**67.9**	17.6
DOUBLEDIPPER	**60.8**	**29.8**	**94.8**	**96.4**	61.6	**53.7**	64.7	**24.5**
Gemma 2 2B (v2) (vanilla)	38.6	8.9	71.8	68.6	**51.2**	13.3	**49.6**	6.5
Zero-shot + Evidence Retrieval	42.0	22.3	66.8	70.6	40.2	30.6	47.6	16.2
DOUBLEDIPPER	**49.5**	**23.7**	**85.8**	**81.6**	50.0	**39.9**	46.7	**18.8**
Mixtral 7x8B (v0.1) (vanilla)	42.6	13.7	73.0	66.2	**51.0**	18.2	67.7	8.4
Zero-shot + Evidence Retrieval	47.4	18.8	81.8	73.6	50.6	26.3	**67.9**	13.1
DOUBLEDIPPER	**52.2**	**22.3**	**91.8**	**86.0**	47.8	**35.1**	66.6	**16.0**

Table 7: Accuracy of the QA task for the vanilla baseline (prompting the model to only answer the question), *Zero-shot + Evidence Retrieval* (prompting the model to explicitly identify the relevant passage(s) before generating the answer) and DOUBLEDIPPER with 3 demonstrations generated by Gemma 2 2B.

		Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemini Nano	$k=1$	60.03	37.68	85.20	88.20	56.40	62.50	60.68	29.56
	$k=3$	62.12	40.55	86.60	95.40	56.20	65.12	60.52	30.44
	$k=5$	62.25	41.79	87.00	95.60	55.20	65.05	60.56	30.52
	$k=10$	62.33	43.16	86.00	96.20	55.20	65.37	60.44	29.96
Gemini Flash	$k=1$	65.34	48.83	84.80	90.80	66.40	60.17	65.16	41.19
	$k=3$	66.11	48.03	85.80	95.00	68.60	60.58	65.04	39.71
	$k=5$	65.87	47.49	87.20	95.00	66.60	61.13	64.20	39.48
	$k=10$	66.08	46.13	86.20	95.20	69.00	62.03	63.80	40.18
Gemma 2B (v2)	$k=1$	47.05	24.05	73.60	77.00	50.20	41.32	47.04	16.13
	$k=3$	49.48	23.66	85.80	81.60	50.00	39.85	46.68	18.77
	$k=5$	49.59	26.70	85.40	80.40	48.40	40.34	46.56	19.30
	$k=10$	49.92	26.43	85.80	81.40	49.00	40.91	47.68	18.26
Gemma 9B (v2)	$k=1$	58.77	37.58	81.00	87.60	51.60	58.46	63.44	31.68
	$k=3$	61.20	41.70	84.00	95.00	51.40	61.21	61.80	33.31
	$k=5$	61.40	44.42	84.20	94.80	50.00	62.99	60.76	32.60
	$k=10$	61.34	43.78	84.40	95.80	50.40	63.01	59.96	32.05
Llama 3.1 8B	$k=1$	57.74	38.20	85.40	84.80	51.80	57.50	60.96	25.49
	$k=3$	59.86	38.75	91.20	90.60	51.00	59.29	58.12	30.09
	$k=5$	60.58	41.56	89.80	91.40	51.00	61.31	57.68	31.28
	$k=10$	61.41	44.68	89.00	91.00	51.00	63.88	56.36	33.98
Mistral 7B	$k=1$	48.91	28.86	67.60	77.40	50.00	40.62	62.00	15.91
	$k=3$	51.03	28.65	68.40	88.80	50.60	43.44	60.68	16.66
	$k=5$	51.12	29.86	69.20	89.00	49.40	43.83	60.36	16.23
	$k=10$	51.44	30.14	68.20	89.40	49.00	46.49	59.96	16.91
Mixtral 7x8B	$k=1$	49.26	19.83	86.00	74.80	49.40	31.73	69.00	14.07
	$k=3$	52.22	22.31	91.80	86.00	47.80	35.10	66.56	15.97
	$k=5$	51.65	23.55	91.40	82.40	47.20	34.64	64.92	17.46
	$k=10$	52.18	23.49	91.80	84.40	47.00	36.23	64.00	18.32

Table 8: Performance of DOUBLEDIPPER on our evaluation set with various numbers of self-generated few shot demonstrations (k).

efficient models could be used to generate demonstrations for DOUBLEDIPPER. To do this, we created demonstrations using two smaller LLMs, Gemma 3 1B (Kamath et al., 2025) and Qwen 2.5 0.5B (Bai et al., 2023a), and provid them to a subset of our evaluation models (Llama 3.1, Gemma

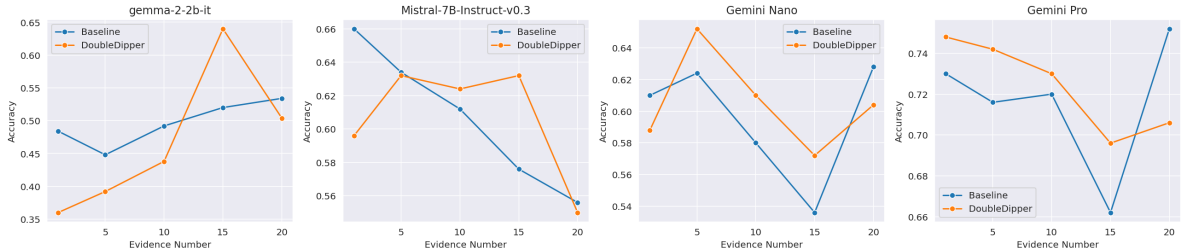

Figure 5: Performance (accuracy) of Gemma 2 2B, Mistral 7B, Gemini Nano and Gemini Pro with and without DOUBLEDIPPER on our sample of the Lost-in-the-middle dataset (Liu et al., 2023b) according to the position of the document that contains the answer.

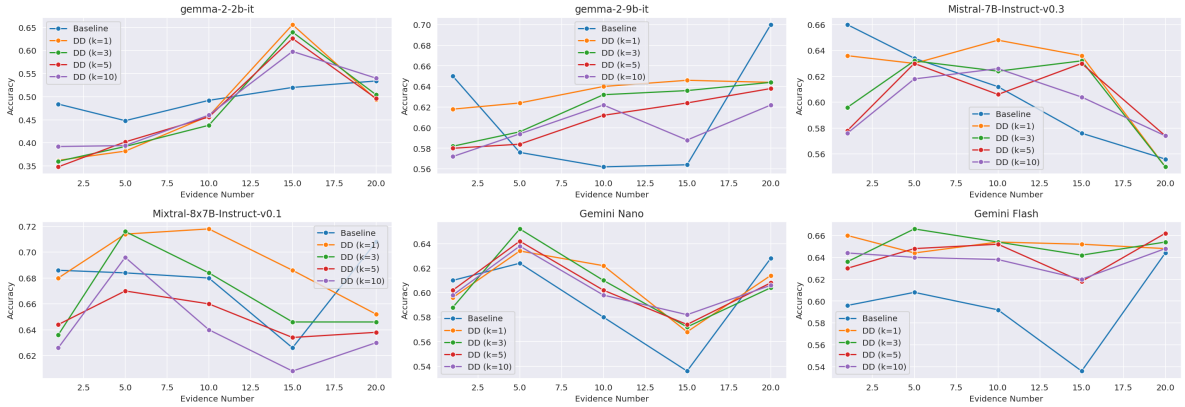

Figure 6: Performance (accuracy) of several models with DOUBLEDIPPER according to the number of self-generated demonstrations in the prompt.

		Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemini Pro	Gemma 2B	70.4	46.8	97.4	99.0	**79.6**	60.9	72.4	36.4
	Self	**71.6**	**49.6**	**98.2**	**99.6**	78.8	**62.5**	**72.5**	**40.1**
Gemini Flash	Gemma 2B	66.1	48.0	85.8	95.0	68.6	60.6	65.0	39.7
	Self	67.5	49.5	88.6	**96.8**	65.6	64.3	**65.2**	**42.5**
	Gemini Pro	**68.1**	**50.3**	**90.4**	96.0	**69.0**	**64.8**	64.8	41.6
Gemini Nano	Gemma 2B	62.1	**40.6**	86.6	95.4	**56.2**	65.1	60.5	30.4
	Self	61.7	39.8	85.8	95.2	**56.2**	64.3	**60.7**	30.0
	Gemini Pro	**62.8**	40.4	**87.2**	**99.0**	54.4	**66.8**	60.4	**31.7**
Gemma 2 2B	Self	49.5	23.7	**85.8**	81.6	**50.0**	39.9	**46.7**	**18.8**
	Gemini Pro	**51.0**	**25.2**	83.8	**93.4**	48.0	**41.7**	46.2	18.4
Gemma 2 9B	Gemma 2B	61.2	41.7	84.0	95.0	51.4	61.2	61.8	33.3
	Self	62.0	44.8	82.4	97.0	50.6	62.9	61.5	**34.9**
	Gemini Pro	**63.5**	**46.2**	**86.4**	**99.2**	**52.2**	**64.4**	**61.4**	34.8
Llama 3.1 8B	Gemma 2B	59.9	38.7	**91.2**	90.6	51.0	59.3	**58.1**	30.1
	Self	60.1	**41.3**	88.6	89.8	50.8	61.5	57.8	31.0
	Gemini Pro	**61.5**	41.0	89.8	**95.6**	**51.6**	**63.5**	57.5	**31.7**
Mistral 7B (v0.3)	Gemma 2B	51.0	28.6	68.4	88.8	**50.6**	43.4	60.7	16.7
	Self	49.8	25.2	**68.8**	93.8	49.6	36.5	60.6	14.3
	Gemini Pro	**52.6**	**31.4**	67.2	**96.4**	48.2	**45.7**	**62.1**	**17.0**
Mixtral 7x8B (v0.1)	Gemma 2B	52.2	**22.3**	91.8	86.0	47.8	35.1	66.6	16.0
	Self	49.8	16.0	91.8	86.0	48.8	25.5	**67.8**	12.8
	Gemini Pro	**54.4**	20.8	**94.8**	**96.2**	**49.0**	**36.3**	66.6	**16.9**

Table 9: Performance of DOUBLEDIPPER according to the model for generating the demonstrations (Gemma 2B, Self or Gemini Pro).

2 9B, Gemma 2 27B, and Mistral Nemo). The results show that DOUBLEDIPPER maintains its advantage over baselines, exhibiting only a modest performance drop of less than 2 points compared to using demonstrations from the larger Gemma 2 2B. This finding suggests that DOUBLEDIPPER remains effective even when its demonstration generation component is replaced with more lightweight

models.

Can DOUBLEDIPPER benefit from incorrect demonstrations? To answer this question, we prompt Gemma 2 2B to generate a question and an *incorrect* answer and provide these demonstrations to a sample of our tested models (Llama 3.1, Gemma 2 9B, Gemma 2 27B and Mistral Nemo). The results are mixed and not conclusive: some models barely benefit or suffer from incorrect demonstrations (+1.6 for Gemma 2 9B, -0.4 for Gemma 2 27B), while others somehow benefit from incorrect demonstrations (+4.8 for Llama 3.1 8B and +5.1 for Mistral Nemo over the baseline). Critically, however, all models still perform substantially worse than DOUBLEDIPPER with correct demonstrations. A possible explanation for the unexpected gains is that even incorrect examples provide useful structural guidance for the task format, a phenomenon observed in prior work (Min et al., 2022).

E.3 Impact of the identification of supporting paragraphs in the QA generation

Table 10 compares the performance of DOUBLEDIPPER to DOUBLEDIPPER without evidence identification.

E.4 In-Context-Learning

Table 11 presents the results of our tested models when prepended with three in-context demonstrations, taken from the Squad 2.0 dataset (Rajpurkar et al., 2018).

		Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemma 2B	DOUBLEDIPPER (QA only)	46.4	16.0	84.6	79.2	49.6	31.1	**49.5**	14.7
	DOUBLEDIPPER	**49.5**	**23.7**	**85.8**	**81.6**	**50.0**	**39.9**	46.7	**18.8**
Gemma 9B	DOUBLEDIPPER (QA only)	55.1	22.5	**91.0**	**96.6**	**52.0**	42.9	61.0	20.0
	DOUBLEDIPPER	**61.2**	**41.7**	84.0	95.0	51.4	**61.2**	**61.8**	**33.3**
Mistral 7B (v0.3)	DOUBLEDIPPER (QA only)	49.1	21.8	**73.4**	**89.8**	48.6	38.6	60.3	11.5
	DOUBLEDIPPER	**51.0**	**28.6**	68.4	88.8	**50.6**	**43.4**	**60.7**	**16.7**
Mixtral 8x7B (v0.1)	DOUBLEDIPPER (QA only)	49.5	17.0	**91.4**	**86.8**	50.8	22.9	65.8	11.9
	DOUBLEDIPPER	**52.2**	**22.3**	91.8	86.0	**47.8**	**35.1**	**66.6**	**16.0**
Llama 3.1 8B	DOUBLEDIPPER (QA only)	32.7	25.0	9.2	41.4	46.6	33.5	56.7	16.6
	DOUBLEDIPPER	**59.9**	**38.7**	**91.2**	**90.6**	**51.0**	**59.3**	**58.1**	**30.1**

Table 10: Performance of DOUBLEDIPPER and DOUBLEDIPPER without instructing the models to retrieve the evidence (QA only) on the QA datasets.

	Avg.	2Wiki	MonoRel	PIR	SRT	HotPotQA	Lost	MuSique
Gemma 2 2B	40.4	11.5	69.4	59.8	50.6	33.1	48.8	9.7
Gemma 2 9B	51.0	21.7	76.8	78.8	53.6	49.1	60.0	17.3
Gemma 2 27B	54.9	21.3	85.0	76.6	58.2	53.3	61.5	28.3
Llama 3.1 8B	40.7	16.4	59.6	53.0	45.6	36.8	62.5	11.0
Llama 3.1 70B	65.4	38.1	93.0	91.4	75.4	59.7	67.6	32.4
Mistral v0.3	42.0	25.5	56.4	52.8	50.2	42.7	58.2	8.3
Mixtral v0.1	45.1	19.6	73.6	65.4	50.4	30.7	66.6	9.6
GPT 4o mini	56.1	29.7	77.4	86.8	65.2	49.4	65.6	18.5

Table 11: Accuracy of the QA task for the ICL experiment with 3 fixed demonstrations prepended to each instance.