# FROM ARTIFICIAL NEEDLES TO REAL HAYSTACKS: IMPROVING RETRIEVAL CAPABILITIES IN LLMS BY FINETUNING ON SYNTHETIC DATA

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent studies have shown that Large Language Models (LLMs) struggle to accurately retrieve information and maintain reasoning capabilities when processing long-context inputs. To address these limitations, we propose a finetuning approach utilizing a carefully designed synthetic dataset comprising numerical key-value retrieval tasks. Our experiments on models like GPT-3.5 Turbo and Mistral 7B demonstrate that finetuning LLMs on this dataset significantly improves LLMs' information retrieval and reasoning capabilities in longer-context settings. We present an analysis of the finetuned models, illustrating the transfer of skills from synthetic to real task evaluations (e.g., $10.5\%$ improvement on $20$ documents MDQA at position $10$ for GPT-3.5 Turbo). We also find that finetuned LLMs' performance on general benchmarks remains almost constant while LLMs finetuned on other baseline long-context augmentation data can encourage hallucination (e.g., on TriviaQA, Mistral 7B finetuned on our synthetic data cause no performance drop while other baseline data can cause a drop that ranges from $2.33\%$ to $6.19\%$). Our study highlights the potential of finetuning on synthetic data for improving the performance of LLMs on longer-context tasks.

## 1 INTRODUCTION

Recent studies have revealed that Large Language Models (LLMs) struggle to accurately retrieve information and maintain reasoning capabilities when processing longer context inputs or when retrieval is required across different parts of their context (Liu et al., 2023; Levy et al., 2024). These limitations hinder their performance on tasks that involve processing and reasoning over extensive textual information, such as summarization or question answering over long passages.

To address these challenges, we propose a novel approach that involves finetuning LLMs on a carefully designed fully numerical *synthetic* dataset containing key-value dictionary retrieval tasks (*i.e.,* see Figure 1 for an example of such a task). We conduct extensive experiments on popular LLMs, including GPT-3.5 Turbo (OpenAI, 2023) and Mistral 7B (Jiang et al., 2023), and find that our method improves their performance on both information retrieval and long-context reasoning.

Specifically, our approach mitigates the "lost-in-the-middle" phenomenon identified by Liu et al. (2023) and significantly improves performance on the FLenQA benchmark (Levy et al., 2024) that measures LLMs' long-context reasoning capability. Interestingly, we observe that finetuning on our proposed dataset often yields more significant improvement compared to finetuning on the corresponding benchmark's data. In addition, it results in only a slight degradation on popular benchmarks such as MMLU (Hendrycks et al., 2021) and HellaSwag (Zellers et al., 2019), indicating that the overall capabilities of the models remain largely unaffected. Finally, another advantage of our proposed dataset is that it contains no *factual* information; as it was recently discovered by Gekhman et al. (2024), finetuning on previously unseen knowledge may encourage hallucinations. Thus, finetuning on our key-value dataset improves LLMs' retrieval and reasoning without suffering from such unwanted characteristics.

Our findings highlight the potential of finetuning on synthetic data as a promising approach to enhancing the performance of LLMs on real downstream tasks. Our paper is organized as follows: in Section 2 we describe the format of the proposed dataset, and its variations that provide (or not)

**Simple dictionary key-value retrieval**

```
Do a task using the list of dictionaries below.

Dictionary [1] {122: 765, 4548: 1475, 4818: 4782}
Dictionary [2] {526: 290, 9205: 9318, 9278: 1565}
...
Dictionary [32] {2931: 8364, 196: 1464, 812: 5363}
...
Dictionary [85] {344: 1579, 116: 617, 330: 411}

Above is a list of dictionaries such that each key and value is an integer. Report the
value of key 2931 and the dictionary it is in.
```

---

```
Desired answer: The value of key 2931 is 8364 and it is in Dictionary [32].
```

Figure 1: An example prompt with desired answer of simple dictionary key-value retrieval task.

an answer template to the model, in Section 3 we present our experimental results, in Section 4 we discuss the main limitations and possible future directions of our work, and in Section 5 we discuss our main conclusions.

### 1.1 Related work

**Long Context LLMs.** Recent works have observed LLMs' limited retrieval and reasoning capabilities in the long-context setting. Liu et al. (2023) discovered a positional bias when LLMs retrieve information from long contexts. In particular, the authors found out that the retrieval accuracy drops when the desired information lies in the middle of the context. Kamradt (2023) conducted the "needle-in-a-haystack" experiment by placing a random fact (the "needle") in a long input context (the "haystack") and observed that LLMs struggle to spot the needle as the input context length grows. To mitigate this behavior, Yu (2024) and An et al. (2024) finetuned LLMs on long-context augmentation data consisting of long-context question-answering tasks to enhance LLMs' long-context capabilities. Tang et al. (2023) shuffled the prompt and marginalized the prompt order biases in the long-context setting and Zhang et al. (2024) re-scaled the indices in positional encoding. Levy et al. (2024) introduced a benchmark, FLenQA, by extending input samples with varying lengths and types of padding, discovering LLMs' significant degradation in reasoning ability at context lengths much shorter than the maximum limit.

There are also other relevant works on long-context LLMs (Junqing et al., 2023; Mohtashami & Jaggi, 2023; Chen et al., 2023b; Bai et al., 2023; An et al., 2023). Xu et al. (2023) showed that Retrieval Augmented Generation (RAG) can be as accurate as full finetuning on longer context windows. Chen et al. (2023a) extended the LLM's predetermined context limit by treating it as an interactive agent who processes the input through iterative prompting. Jin et al. (2024) extended LLM's context window by remapping the unseen relative positions during inference. Zhu et al. (2024) introduced "LONGEMBED", a benchmark and suite of training-free strategies to extend embedding models' context window up to 32,768 tokens, leveraging Rotary Position Encoding (RoPE) in processing long contexts. Fu et al. (2024) proposed a data engineering recipe for scaling LLMs to $128k$ context lengths through lightweight continual pretraining on a balanced mixture of length-upsampled data. Peysakhovich & Lerer (2023) proposed "attention sorting," a method that improves long context models by iteratively sorting documents based on attention and generating responses with the re-ordered context.

**Data-centric AI.** In recent years, the field of data-centric AI has emerged, which focuses on improving the quality and efficiency of AI systems through data-oriented approaches rather than model-centric techniques (Sener & Savarese, 2018; Ghorbani & Zou, 2019; Zha et al., 2023; Albalak et al., 2024). Gadre et al. (2024) and Mazumder et al. (2024) proposed benchmarks that fix model training code, where the goal is to design better datasets to achieve better performance. Lee et al. (2023) and Zhou et al. (2024) studied the data format in training transformers to learn arithmetic tasks.

**Multi-subkey dictionary key-value retrieval**

```
Do a task using the list of dictionaries below.

Dictionary [1] {(141, 986, 163): 2528, (726, 947, 349, 820): 4130}
Dictionary [2] {(555, 710, 424): 5756, (623, 141, 997): 1633, (957, 634, 969): 7871}
...
Dictionary [6] {(645, 417, 847): 6409, (141, 623, 616): 5617}
...
Dictionary [49] {(710, 105, 141, 799): 5369, (623, 210, 477): 8971, (899, 126, 999):
4409}

Above is a list of dictionaries such that each key is a tuple of integers and each
value is an integer. Report the key that contains the integers 616, 141, 623 (not
necessarily in order), its value, and the dictionary it is in.
```

---

```
Desired answer: The key that contains the integers 616, 141, 623 is (141, 623, 616).
Its value is 5617 and it is in Dictionary [6].
```

Figure 2: An example prompt with desired answer of multi-subkey dictionary key-value retrieval task. Here `(141, 623, 616)` is the *gold key*. Note that `141` and `623` in the *gold key* are also subkeys of other keys.

**LLM Benchmarks and Evals.** Much research has been recently conducted towards the design of meaningful benchmarks that probe the capabilities of LLMs. Benchmarks such as GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019) test whether a model has general language understanding capabilities. MMLU (Hendrycks et al., 2021) aims to measure the models' accuracy across a wide variety of tasks that span STEM, humanities, social sciences, and more, while GSM8k (Cobbe et al., 2021) tests capabilities on school math. In HellaSwag (Zellers et al., 2019) models are presented with an event description and must select the most likely follow-up sentence from a set of carefully selected choices, while HumanEval (Chen et al., 2021) measures their ability to generate code given docstrings. TriviaQA (Joshi et al., 2017) is a reading comprehension benchmark and NQ-Open (Lee et al., 2019; Kwiatkowski et al., 2019a) is an open domain question-answering benchmark where the question-answer pairs are collected from a diverse set of fields.

## 2 Synthetic Dataset of Retrieval Tasks

In this section, we introduce the dataset on which we finetune the models. The dataset consists of two synthetic retrieval tasks: 1) simple dictionary key-value retrieval and 2) multi-subkey dictionary key-value retrieval.

**Simple dictionary key-value retrieval.** In this task, we provide the model with a list of dictionaries of integer keys and values, and ask it to retrieve the value of a specified key (denoted as the *gold key*). Figure 1 shows an example of this task and the detailed algorithm is shown in Algorithm 2.

**Multi-subkey dictionary key-value retrieval.** For models that can already tackle the first task (e.g., for the first task GPT 3.5 Turbo achieves around 0.99 accuracy irrespective of the position of *gold key*), we design a harder version of the key-value retrieval task where each key is a tuple of subkeys. Other keys can share some but not all of the subkeys of the *gold key*. We increase the difficulty of this task by randomizing the order of subkeys in the prompt so that the order is not necessarily the same as that of the *gold key*. Figure 2 shows an example of this task and the detailed algorithm is shown in Algorithm 3.

**Prompt with an answer template.** Note that with the prompt in Figure 1, slightly different answers like "`8364` is the value of key `2931` in dictionary `32`" and "Dictionary [32] has the key `2931` with value `8364`" are also correct. Therefore, since the model is finetuned on the entire answer, during supervised finetuning, it also learns the format of our provided answer besides learning to retrieve the

**Simple dictionary key-value retrieval (with an answer template)**

```
Do a task using the list of dictionaries below.

Dictionary [1] {122: 765, 4548: 1475, 4818: 4782}
Dictionary [2] {526: 290, 9205: 9318, 9278: 1565}
...
Dictionary [32] {2931: 8364, 196: 1464, 812: 5363}
...
Dictionary [85] {344: 1579, 116: 617, 330: 411}

Above is a list of dictionaries such that each key and value is an integer. Report the
value of key 2931 and the dictionary it is in. Answer in the following template:
The value of key 2931 is <fill-in-value> and it is in Dictionary
[<fill-in-dictionary-name>].
```

---

```
Desired answer: The value of key 2931 is 8364 and it is in Dictionary [32].
```

Figure 3: The prompt of the simple dictionary key-value retrieval task is provided with an answer template.

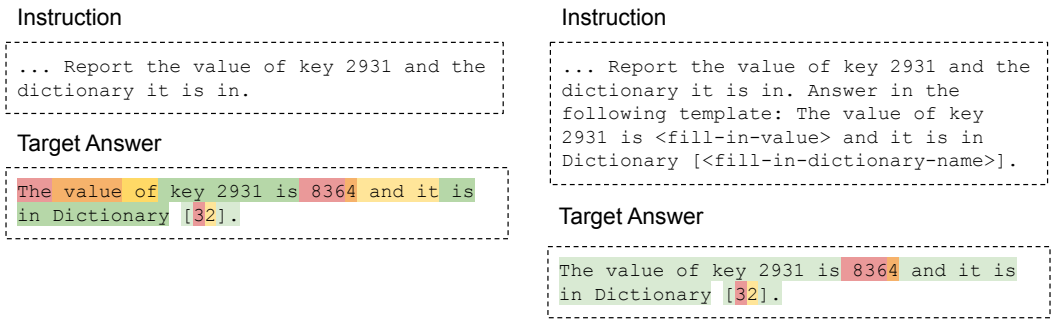



Figure 4: Token-level loss on the target answer when provided with (right) and without (left) an answer template, where red indicates high and green low loss.

desired value. In order to make the model only focus on retrieving the correct value without being affected by the format of the answer, we provide the model with an answer template with which we want the model to answer. Figure 3 shows an example of a prompt with an answer template. In Figure 4 we visualize the token-level loss on the target answer, where red indicates high and green low loss. If an answer template is provided, the loss on the formatting part is small. This lets the model to focus on the important part and learn the right skill rather than how to answer the question.

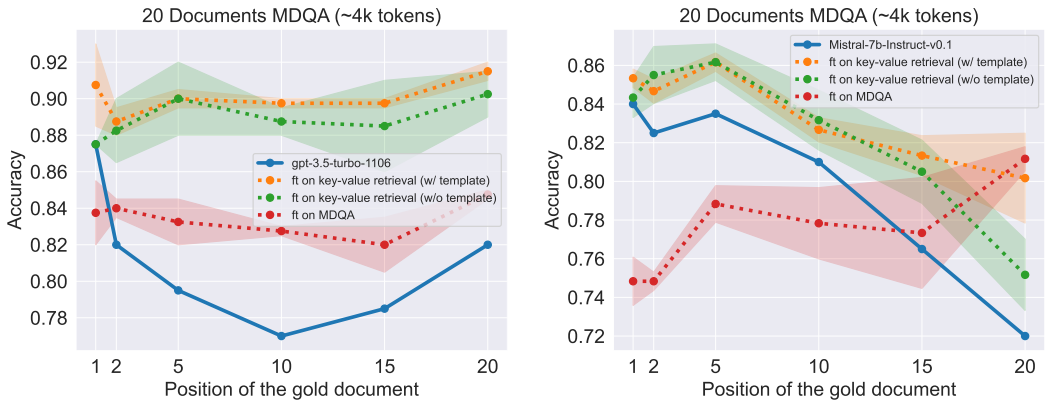
## 3 Experiments and Results

Our goal is to investigate whether finetuning LLMs (in particular, GPT-3.5 Turbo and Mistral 7B [1]) on our proposed synthetic numerical retrieval tasks improves their long context capabilities on natural language tasks: multi-document question answering (MDQA) (Liu et al., 2023) and flexible length question answering (FLenQA) (Levy et al., 2024).

### 3.1 Stage 1: Finetuning LLMs on Synthetic Retrieval Tasks

For Mistral 7B, our dataset consists of 350 samples of simple dictionary key-value retrieval tasks. Each task has 85 dictionaries and each dictionary has 3 to 4 keys, so each prompt has roughly 3900 tokens (to leave space for the tokens in the answer as `Mistral-7B-Instruct-v0.1` uses a sliding window context length of 4096). We finetune the model on only the answer part (masking out

[1] `gpt-3.5-turbo-1106` and `Mistral-7B-Instruct-v0.1`




(a) GPT-3.5 Turbo and the finetuned versions.

(b) Mistral 7B and the finetuned versions.

Figure 5: Performance of GPT-3.5 Turbo, Mistral 7B and their corresponding finetuned versions on the MDQA task.

the instruction part) for 2 epochs. More implementation details are in A.1. Figure 11 shows Mistral 7B's performance on simple dictionary key-value retrieval task before and after finetuning.

Since GPT-3.5 Turbo already performs well on simple dictionary key-value retrieval task, we finetune it on multi-subkey dictionary key-value retrieval tasks. The dataset consists of 150 samples and each sample has 49 dictionaries. We finetune the model for 3 epochs using OpenAI's API.

## 3.2 Stage 2: Evaluations on long context retrieval and reasoning tasks

### 3.2.1 Multi-document question answering (MDQA)

We test models' capabilities of retrieving important information in a long context setting. In MDQA, we provide the model with $k$ documents and prompt it to answer a question such that only 1 of $k$ documents (denoted as the *gold document*) contains the answer and the other $k-1$ documents (denoted as *distractors*) are completely irrelevant to the question. We test the setting of a context with 20 documents (around 4K tokens) and place *gold document* at positions $\{1, 2, 5, 10, 15, 20\}$ [2]. For each position, we test the model on 200 task samples and measure the accuracy using the maximum subspan exact match as in (Liu et al., 2023).

> **Finding 1:** *Finetuning LLMs on synthetic key-value retrieval tasks enhances their performance on practical retrieval tasks, demonstrating effective transfer of learned capabilities.*

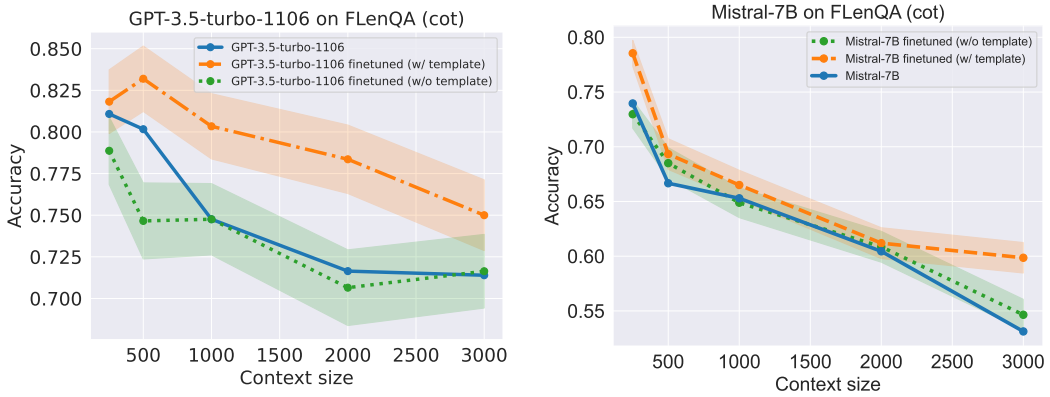
The result of 20 documents MDQA is shown in Figure 5, where x-axis is the position of *gold document*. In Figure 5a, for the original GPT-3.5 Turbo model, there is a U-shaped performance curve, indicating that the performance is highest if the important information is at the beginning or at the end of the input context, with the model struggling to retrieve the answer if the important information is in the middle. Finetuning the models on synthetic retrieval tasks flattens the U-shaped curve and information is much more accurately retrieved over all positions across the input context. In Figure 5b, the original Mistral 7B model has a primacy bias – in the sense that it can more accurately retrieve information that is at the beginning of the input context. Finetuning the models on our proposed data manages to improve the accuracy across all the positions in the input context. In addition, when the finetuning dataset contains a template, Mistral seems to mitigate this primacy bias, showcasing a more uniform accuracy across all the positions in the input context.

> **Finding 2:** *Synthetic data is better than MDQA data even if the goal is to perform better in MDQA task.*

[2] For example, *gold document* placed at position 1 means it is the first document in the context.

As a comparison, we also finetune the models on the MDQA dataset itself for roughly the same number of training tokens and see how finetuned models perform. Since the MDQA dataset only provides the ground truth answers in one or two words, we prompt GPT-3.5 Turbo with correct answers and let it form a complete sentence as the target answer. As shown in Figure 5a, GPT-3.5 Turbo finetuned on our synthetic data perform better than the one finetuned on MDQA. In Figure 5b we can see that despite training on MDQA tasks, Mistral 7B still struggles to perform well on MDQA, with a significant performance drop when *gold document* is at the beginning of the prompt. These findings underscore the effectiveness of our synthetic data generation method, which enhances performance on specific datasets like MDQA, even surpassing direct finetuning on the target dataset.

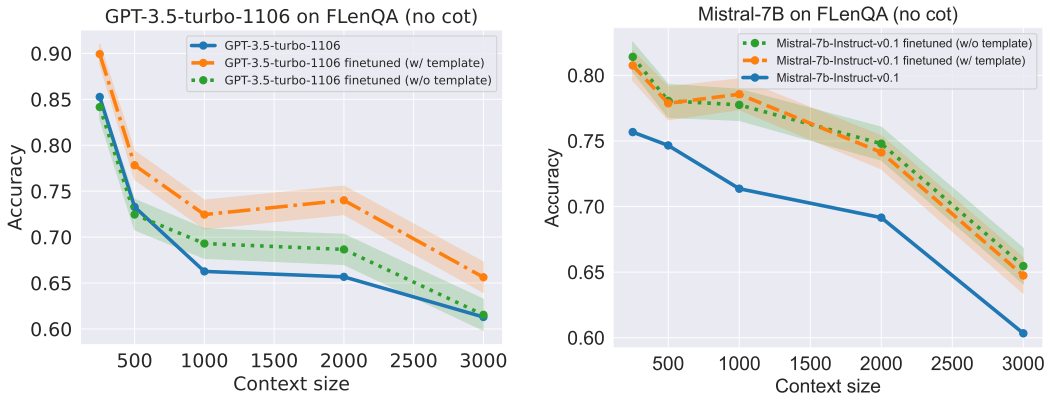
#### 3.2.2 FLEXIBLE LENGTH QUESTION ANSWERING (FLENQA)




(a) GPT-3.5 Turbo and the finetuned versions.

(b) Mistral 7B and the finetuned versions.

Figure 6: Performance of GPT-3.5 Turbo, Mistral 7B and their corresponding finetuned versions on the FLenQA task, using chain-of-thought prompting.




(a) GPT-3.5 Turbo and the finetuned versions.

(b) Mistral 7B and the finetuned versions.

Figure 7: Performance of GPT-3.5 Turbo, Mistral 7B and their corresponding finetuned models on the FLenQA task without employing chain-of-thought prompting.

We also test models' long context reasoning capabilities. FLenQA is a dataset comprising reasoning tasks with varying length that ranges from 250 tokens to 3000 tokens. Each task consists of a context and a "True" or "False" question that can be answered by two key sentences from the context. We test chain-of-thought (Wei et al., 2022) and non chain-of-thought prompting, each with a total of 2000 task samples. For chain-of-thought prompting, we ask the model to produce the result step by step and derive the answer ("True" or "False") at the end, and in the non chain-of-thought prompting we ask the model to directly answer "True" or "False".

**Finding 3:** *Finetuning LLMs on synthetic key-value retrieval tasks improves LLMs' long-context reasoning capabilities, even if explicit chain-of-thought reasoning is not allowed.*

In Figure 6 and 7 we present our results on the FLenQA dataset. The x-axes represent the number of tokens in the context, while the y-axes represent the accuracy of the response. Figure 6 shows results where chain-of-thought prompting is employed. In Figure 6a, we notice that although the model suffers from a performance drop if finetuned on data without answer template, finetuning GPT-3.5 Turbo on data with answer template significantly improves model's chain-of-thought reasoning capability. In Figure 6b we can also see that finetuning Mistral 7B on data with answer template improves models chain-of-thought capability. We hypothesize that the reason for this is that the finetuned models utilize their improved retrieval capabilities to capture relevant information more accurately, which helps them deduce the answer.

Figure 7 presents results where models are required to directly answer with "True" or "False" without providing explicit reasoning. The results show a notable improvement in performance for finetuned models. This improvement is significant because it demonstrates that, even if explicit reasoning (that is related to retrieval capability) is not allowed, finetuning on our proposed synthetic tasks enhances the models' internal reasoning capabilities.

**Finding 4:** *LLMs finetuned on synthetic tasks with answer templates are better.*

From Figure 5, 6 and 7, we can observe that models finetuned on synthetic key-value retrieval tasks with answer templates perform better on MDQA and FLenQA than that on without answer templates. This verifies our hypothesis that having an answer template helps the model learn the right skill more efficiently. This highlights a key advantage of synthetic data: it allows for greater control over the model's output format. Unlike real-world tasks where developing answer templates can be challenging, synthetic tasks allow for easy implementation of structured response formats, facilitating skill learning.

### 3.3 Stage 3: Evaluation of finetuned models' general capabilities

**Finding 5:** *Finetuning LLMs on synthetic key-value retrieval tasks does not hurt models' general capabilities.*

One possible drawback of our approach is that finetuning on the proposed artificial tasks would severely harm the general purpose capabilities of the tested models. In order to assess this concern, we tested the original and finetuned versions of GPT-3.5 Turbo and Mistral 7B on some general purpose benchmarks. Note that for our assessments we used the codebases of Gao et al. (2023) and Fu et al. (2023).

| MODEL | MMLU | HellaSwag | GSM8K | Triviaqa | NQ-Open |
|---|---|---|---|---|---|
| Mistral-7B | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| Mistral-7B ft (w/template) | 53.44 (+0.02) | 56.22 (−0.09) | 34.34 (−0.31) | 47.74 (+0.11) | 11.98 (+0.37) |
| Mistral-7B ft (w/o template) | 53.42 (−0.00) | 56.30 (−0.01) | 34.14 (−0.51) | 47.62 (−0.01) | 11.40 (−0.21) |
| GPT-3.5-turbo | 68.07 | - | 72.33 | - | - |
| GPT-3.5-turbo ft (w/template) | 67.75 (−0.32) | - | 71.65 (−0.68) | - | - |
| GPT-3.5-turbo ft (w/o template) | 68.16 (+0.09) | - | 75.06 (+2.73) | - | - |

Table 1: Model's performance evaluated on general ability benchmarks. All numbers are reported in percentage. Here "w/" and "w/o" denote the models that are finetuned on the the synthetic tasks that were described in Section 2.

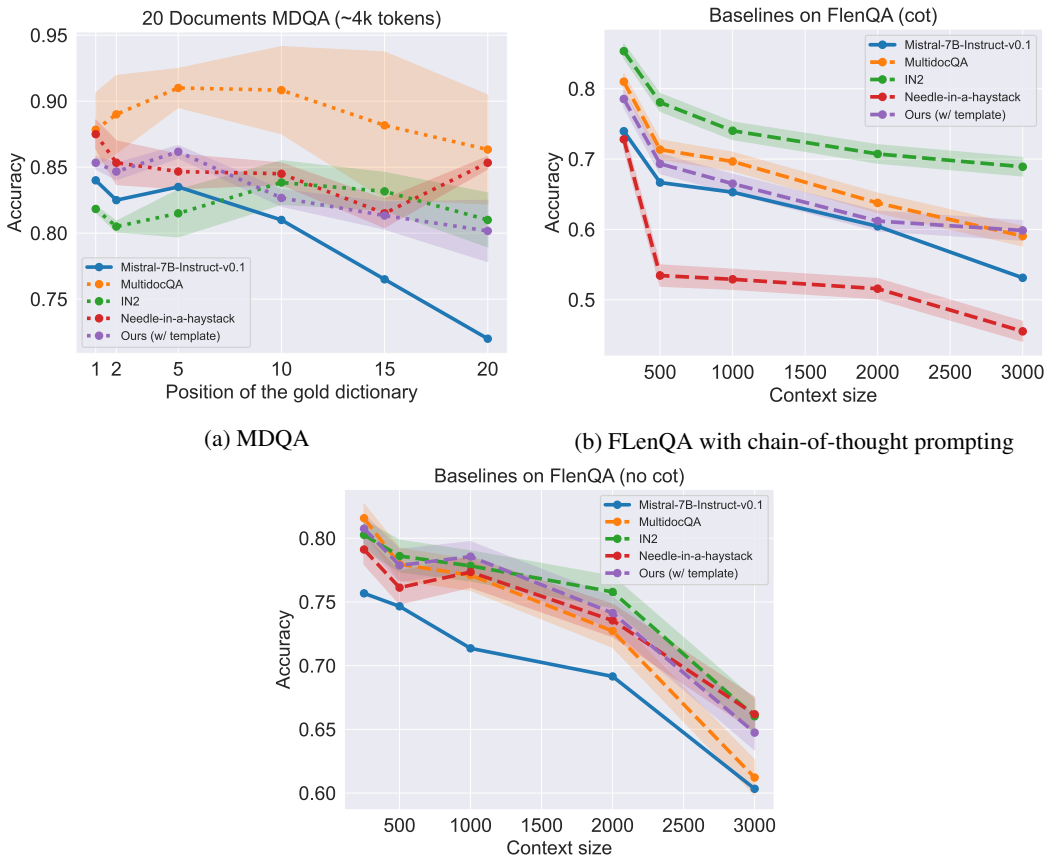
The results can be seen in Table 1. In particular, we consider five widely used benchmarks: MMLU (Hendrycks et al., 2021)[3], HellaSwag (Zellers et al., 2019), GSM8k (Cobbe et al., 2021), TriviaQA

[3]Due to computational constraints, we did not evaluate GPT-3.5 Turbo on all benchmarks, and for MMLU we use 20% of the full dataset.

(Joshi et al., 2017) and NQ-Open (Kwiatkowski et al., 2019b). What we can observe is that all the finetuning strategies result in no significant degradation on the general purpose benchmarks mentioned above.

### 3.4 STAGE 4: COMPARISONS WITH OTHER BASELINES

We also consider three additional long-context augmentation datasets as baselines: MultidocQA (Yu, 2024), IN2 (An et al., 2024), and Needle-in-a-haystack (Kamradt, 2023). MultidocQA is a dataset of multiple documents question and answering where the model needs to paraphrase the document before answering. IN2 is a long-context question answering dataset where the answer can be deduced from one or multiple parts of the context. Needle-in-a-haystack is a widely used long-context test set where the model is prompted to identify some key information (the needle) within a long context (the haystack). We finetune Mistral 7B on these baselines, using roughly the same number of training tokens and report their performance on MDQA, FLenQA, and general purpose benchmarks.




(a) MDQA

(b) FLenQA with chain-of-thought prompting

(c) FLenQA without chain-of-though prompting

Figure 8: Performance of finetuned Mistral 7B on (a) MDQA, (b) FLenQA with chain-of-thought prompting, and (c) FLenQA without chain-of-thought prompting.

**Finding 6:** *Synthetic data do not encourage hallucinations that other baselines may yield.*

From Figure 8 and Table 2, we can see that while some baselines outperform our proposed data on either MDQA or FLenQA, they all have more significant degradation on the general benchmarks we test, especially on TriviaQA and NQ-Open. One possible reason is that all other baselines contain factual information. Gekhman et al. (2024) shows that finetuning on factual information encourages hallucinations, something that we verify observing the significant degradation on TriviaQA and NQ-Open, which are knowledge-based benchmarks. In contrast, our proposed dataset is purely

| Finetuning dataset | MMLU | HellaSwag | GSM8K | Triviaqa | NQ-Open |
|---|---|---|---|---|---|
| Original Mistral-7B | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| Ours (w/template) | 53.44 (+0.02) | 56.22 (−0.09) | 34.34 (−0.31) | 47.74 (+0.11) | 11.98 (+0.37) |
| MultidocQA (Yu, 2024) | 53.19 (-0.22) | 56.27 (-0.04) | 33.28 (-1.36) | 45.20 (-2.43) | 8.69 (-2.91) |
| IN2 (An et al., 2024) | 53.49 (+0.07) | 56.44 (+0.13) | 34.98 (+0.32) | 45.44 (-2.19) | 9.80 (-1.81) |
| Needle-in-a-haystack (Kamradt, 2023) | 52.83 (-0.59) | 56.22 (-0.09) | 33.79 (-0.86) | 41.30 (-6.33) | 4.88 (-6.73) |
| MDQA (Liu et al., 2023) | 52.94 (-0.47) | 56.23 (-0.07) | 34.72 (-0.07) | 44.77 (-2.85) | 7.64 (-3.96) |

Table 2: Mistral 7B and finetuned versions' performance evaluated on general ability benchmarks. All numbers are reported in percentage.

synthetic, comprising of key-value pairs, and as a result, does not encourage hallucinations. We also highlight another benefit of our synthetic data: since it does not contain any factual information, it will not have the problem of containing potential outdated information that further encourages hallucinations, from which other long-context augmentation datasets may suffer.

### 3.5 Stage 5: Evaluation on longer-context setting

We also test the longer-context setting. We finetune `Mistral-7b-Instruct-v0.2` on simple key-value retrieval task with maximum context length of 24K and test it on MDQA. We observe a clear improvement over the original model as shown in Figure 9.

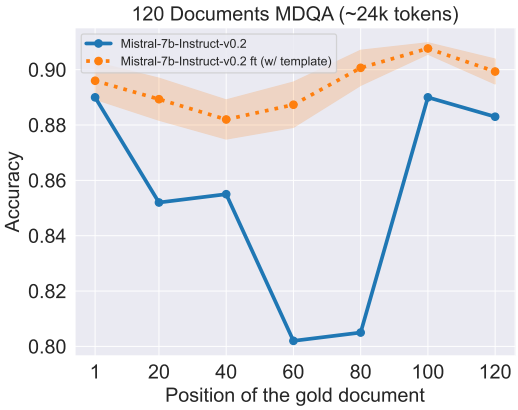



Figure 9: Performance of finetuned `Mistral-7b-Instruct-v0.2` on 120 documents MDQA.

## 4 Limitations and future work

Our dataset does have a limitation. MDQA benchmark also has another version where *distractors* are relevant distractors, meaning that they are documents retrieved by a retrieval system (based on the relevance score) that do not contain the answer. Models finetuned on our dataset will not improve in this setting, as is shown in Figure 10. A possible future work of this study is to add our synthetic retrieval dataset as a small part of a larger instruction finetuning dataset and see the difference between models finetuned with and without synthetic retrieval data and observe how they perform differently on long context retrieval and reasoning tasks.

## 5 Conclusion

In this work, we introduce a novel finetuning approach that leverages carefully designed synthetic datasets to enhance the information retrieval and reasoning capabilities of LLMs in real downstream tasks. Our study demonstrates that finetuning on our proposed synthetic data significantly improves the performance of the tested models on tasks like MDQA and FLenQA, mitigating the "lost-in-the-middle" behavior that was observed in Liu et al. (2023). On the other hand, we find that after

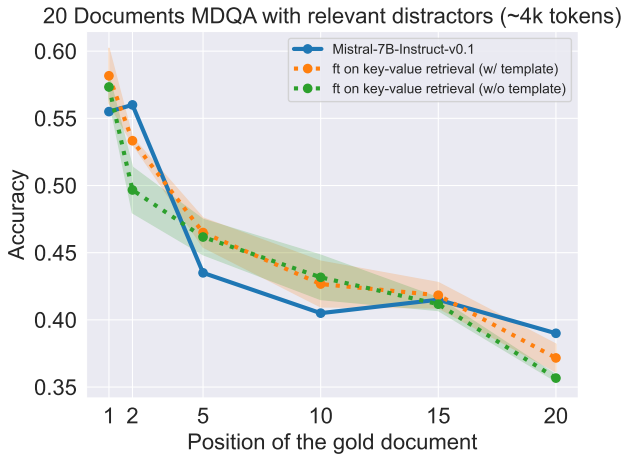



Figure 10: Mistral 7B and the finetuned versions on MDQA with relevant distractors. The finetuned variants do not show a significant improvement over the original model.

finetuning, the models' performance on general benchmarks remains almost constant, something that indicates that their overall capabilities are mostly unaffected. We also find that compared to other long-context augmentation datasets that contain factual information, our purely artificial data does not encourage hallucinations. Moreover, it will not have the problem of containing potential outdated information. Thus, we believe that our study demonstrates the potential of finetuning LLMs on carefully crafted synthetic datasets to enhance their capabilities on downstream tasks. We hope that our findings will inspire further research into the development of effective synthetic datasets.

## REFERENCES

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

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

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. Make your llm fully utilize the context. *arXiv preprint arXiv:2404.16811*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *arXiv preprint arXiv:2305.17306*, 2023.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*, 2024.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251, 2019.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. Llm maybe longlm: Selfextend llm context window without tuning. In *Forty-first International Conference on Machine Learning*, 2024.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

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

G Kamradt. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

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. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019a. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612. URL https://www.aclweb.org/anthology/P19-1612.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

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

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, et al. Dataperf: Benchmarks for data-centric ai development. *Advances in Neural Information Processing Systems*, 36, 2024.

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

OpenAI. Chatgpt, 2023. URL https://openai.com/blog/chatgpt. Accessed: 2024-03-29.

Alexander Peysakhovich and Adam Lerer. Attention sorting combats recency bias in long context language models. *arXiv preprint arXiv:2310.01427*, 2023.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *arXiv preprint arXiv:2310.07712*, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Yijiong Yu. Training with "paraphrasing the original text" improves long-context performance, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. Data-centric ai: Perspectives and challenges. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pp. 945–948. SIAM, 2023.

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

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.

Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. Longembed: Extending embedding models for long context retrieval. *arXiv preprint arXiv:2404.12096*, 2024.

# A Training Details

## A.1 Finetuning Mistral 7B and GPT 3.5 Turbo

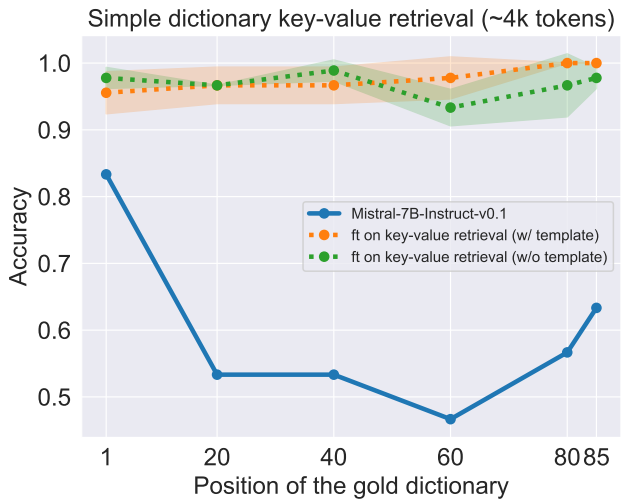



Figure 11: Mistral 7B and the finetuned versions on simple dictionary key-value retrieval.

For Mistral 7B, we choose simple dictionary key-value retrieval as the task to finetune on. We use two prompting strategies to prepare the dataset: with and without an answer template as described in Section 2. For each prompting strategy we generate 3 different datasets using the same configuration but with different seeds. Each dataset consists of 350 simple dictionary key-value retrieval tasks (roughly 4K tokens in each task). Each task has 85 dictionaries and each dictionary has 3 to 4 keys. Each key and value is an integer of 3 to 4 digits (in particular, we choose $l_{min} = r_{min} = 3, l_{max} = r_{max} = 4$). We finetune Mistral 7B on all attention layers and use a global batch size of 16 and finetune the model for 2 epochs on each dataset with learning rate $5 \times 10^{-6}$. For evaluation results, we average across 3 runs, each with different training data and seed.

For GPT-3.5 Turbo, we choose multi-subkey key-value retrieval as the task to finetune on (in particular, we choose $\texttt{num_dict} = 49, l_{min} = r_{min} = 3, l_{max} = r_{max} = 4, \texttt{n_keys} = 3, \texttt{n_common} = 2. p_{\texttt{share}} = 0.5$). For each prompting strategy, we generate 2 different datasets. Each dataset consists of 150 multi-subkey key-value retrieval tasks (roughly 4K tokens in each task). Each task has 49 dictionaries. We finetune GPT-3.5 Turbo for 2 epochs on each dataset using OpenAI API. For evaluation results, we average across 2 runs.

# B ADDITIONAL ABLATION STUDY

In this section, we provide additional ablation studies to investigate the effect of training epochs, training data size, and training Mistral on different class of synthetic tasks.

## B.1 THE EFFECT OF TRAINING EPOCHS AND TRAINING DATA SIZE

To investigate how the amount of training (in particular, training data size and the number of training epochs) affect the model's performance on long-context tasks (MDQA and FlenQA) and general benchmarks, we train `Mistral-7B-Instruct-v0.1` on simple dictionary key-value retrieval (denoted as "sd") using the same configuration as in Section 3 but train it for 1 epoch (labeled as "sd (ep1)"), 4 epochs (labeled as "sd (ep4)") and 2 epochs but with double training data (labeled as "sd x2 (ep2)"). We test the finetuned models on MDQA, FLenQA and general benchmarks, and compare the result with the original model (labeled as "original") and the model we used in Section 3 (labeled as "sd (ep2)"); the results are shown in Figure 12 and Table 3 respectively. In Figure 12, we can observe that training on larger dataset ('sd x2 (ep2)") slightly boosts the performance on FLenQA while having a slight degradation on MDQA; training with more epochs slight hurt the performance on MDQA and achieves comparable performance compared to epoch 2 case. However, these performance changes are marginal. On the other hand, epoch 1 case suffers a more significant degradation compared to other three cases on MDQA as shown in Figure 12a. From Table 3, we can see that there is no significant degradation, except in the performance of GSM8K, where more training tokens (correspond to the case "sd (ep 4)" and "sd x2 (ep2)") can cause slightly more degradation. A possible reason for this is that we choose integers as keys and values for retrieval, so it might hurt the model's performance on understanding numbers. A possible future extension is to instead use special tokens as retrieval tokens and train the model on tasks that use such retrieval tokens.

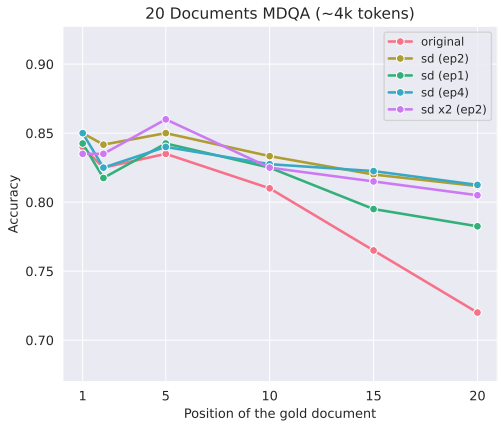
| **Finetuning dataset** | **MMLU** | **HellaSwag** | **GSM8K** | **TriviaQA** | **NQ-Open** |
|---|---|---|---|---|---|
| Original | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| sd (ep1) | 53.38 ($-0.04$) | 56.26 ($-0.05$) | 34.58 ($-0.07$) | 47.54 ($-0.09$) | 11.97 ($+0.36$) |
| sd (ep2) | 53.44 ($+0.02$) | 56.22 ($-0.09$) | 34.34 ($-0.31$) | 47.74 ($+0.11$) | 11.98 ($+0.37$) |
| sd (ep4) | 53.29 ($-0.13$) | 56.20 ($-0.11$) | 34.19 ($-0.46$) | 47.63 ($+0.00$) | 11.85 ($+0.25$) |
| sd x2 (ep2) | 53.35 ($-0.07$) | 56.30 ($-0.01$) | 33.89 ($-0.76$) | 47.83 ($+0.20$) | 11.95 ($+0.34$) |

Table 3: Mistral 7B and finetuned versions' performance evaluated on general ability benchmarks. All numbers are reported in percentage.

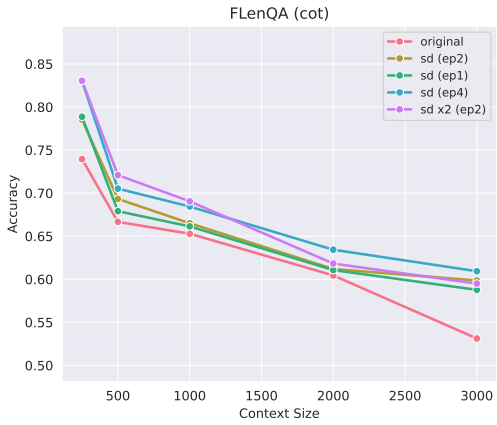
As a control, we also conduct the same experiment on MultidocQA dataset and IN2 dataset. The results for MultidocQA are shown in Figure 13 and Table 4, and the results for IN2 are shown in Figure 14 and Table 5. We can observe that, while training the model with more training tokens on MultidocQA and IN2 can boost the model's performance on MDQA and FLenQA, it can hurt the model more significantly, especially on knowledge-based evaluation sets like TriviaQA and NQ-Open, indicating a greater level of hallucination.

| **Finetuning dataset** | **MMLU** | **HellaSwag** | **GSM8K** | **TriviaQA** | **NQ-Open** |
|---|---|---|---|---|---|
| Original | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| MultidocQA (ep1) | 53.16 ($-0.26$) | 56.16 ($-0.15$) | 34.08 ($-0.57$) | 45.70 ($-1.93$) | 8.57 ($-3.04$) |
| MultidocQA (ep2) | 53.19 ($-0.22$) | 56.27 ($-0.04$) | 33.28 ($-1.36$) | 45.20 ($-2.43$) | 8.69 ($-2.91$) |
| MultidocQA (ep4) | 53.19 ($-0.23$) | 56.37 ($+0.06$) | 33.05 ($-1.60$) | 44.93 ($-2.70$) | 7.63 ($-3.98$) |
| MultidocQA x2 (ep2) | 52.89 ($-0.53$) | 56.20 ($-0.11$) | 33.00 ($-1.65$) | 44.77 ($-2.86$) | 8.15 ($-3.46$) |

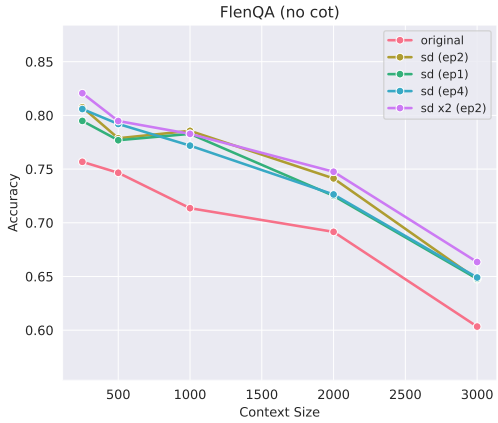
Table 4: Mistral 7B and finetuned versions' performance evaluated on general ability benchmarks. All numbers are reported in percentage.




(a) MDQA




(b) FLenQA with chain-of-thought prompting

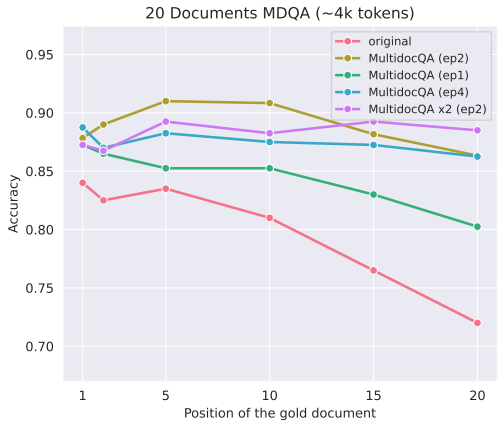



(c) FLenQA without chain-of-though prompting

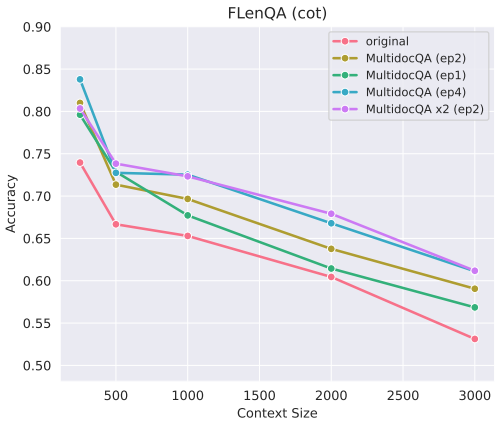
Figure 12: Performance of finetuned Mistral 7B with different training epochs and training sizes, e.g., "sd (ep2)" denotes training on simple dictionary key-value retrieval task (sd) with 2 epochs; "sd x2 (ep2)" denotes training on sd task with 2 epochs but with training data twice as large. Subplots show the average performance of (a) MDQA, (b) FLenQA with chain-of-thought prompting, and (c) FLenQA without chain-of-thought prompting.

| **Finetuning dataset** | **MMLU** | **HellaSwag** | **GSM8K** | **TriviaQA** | **NQ-Open** |
|---|---|---|---|---|---|
| Original | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| IN2 (ep1) | 53.27 (−0.15) | 56.26 (−0.05) | 34.65 (+0.00) | 45.59 (−2.03) | 10.00 (−1.61) |
| IN2 (ep2) | 53.49 (+0.07) | 56.44 (+0.13) | 34.98 (+0.32) | 45.44 (−2.19) | 9.80 (−1.81) |
| IN2 (ep4) | 53.37 (−0.05) | 56.69 (+0.38) | 34.91 (+0.26) | 43.98 (−3.65) | 7.47 (−4.14) |
| IN2 x2 (ep2) | 53.31 (−0.11) | 56.68 (+0.37) | 33.89 (−0.76) | 44.80 (−2.83) | 9.43 (−2.18) |

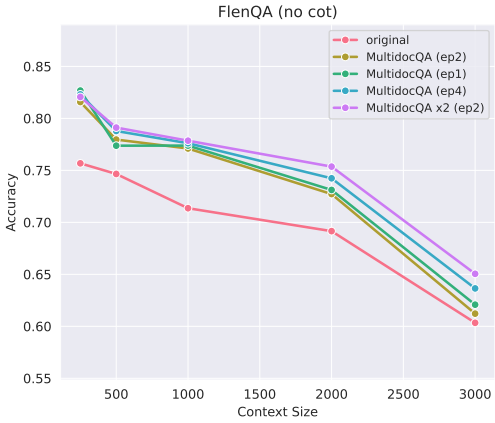
Table 5: Mistral 7B and finetuned versions' performance evaluated on general ability benchmarks. All numbers are reported in percentage.




(a) MDQA

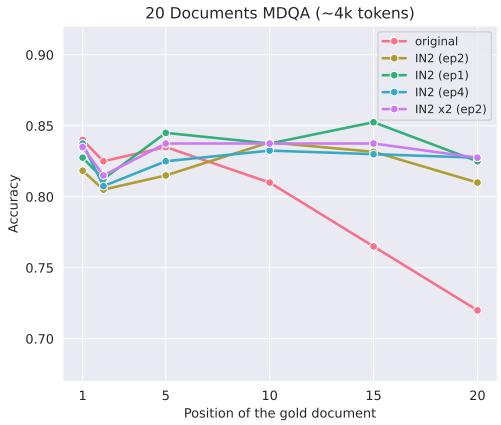



(b) FLenQA with chain-of-thought prompting

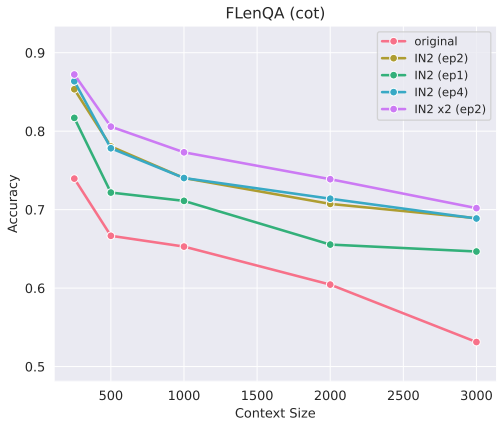



(c) FLenQA without chain-of-though prompting

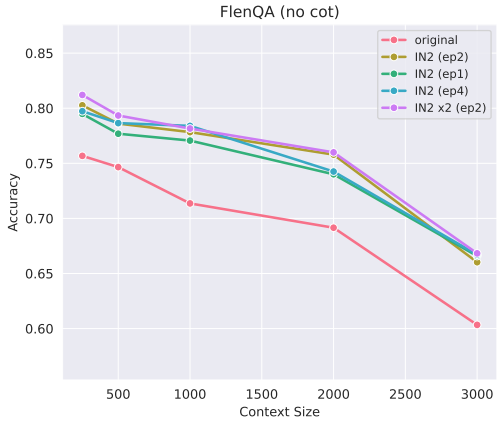
Figure 13: Performance of finetuned Mistral 7B with different training epochs and training sizes, e.g., "MultidocQA (ep2)" denotes training on MultidocQA data with 2 epochs; "MultidocQA x2 (ep2)" denotes training on MultidocQA data with 2 epochs but with training data twice as large. Subplots show the average performance of (a) MDQA, (b) FLenQA with chain-of-thought prompting, and (c) FLenQA without chain-of-thought prompting.




(a) MDQA




(b) FLenQA with chain-of-thought prompting




(c) FLenQA without chain-of-though prompting

Figure 14: Performance of finetuned Mistral 7B with different training epochs and training sizes, e.g., "IN2 (ep2)" denotes training on IN2 data with 2 epochs; "IN2 x2 (ep2)" denotes training on IN2 data with 2 epochs but with training data twice as large. Subplots show the average performance of (a) MDQA, (b) FLenQA with chain-of-thought prompting, and (c) FLenQA without chain-of-thought prompting.

### B.2 TRAINING MISTRAL ON DIFFERENT RETRIEVAL TASKS

In Section 3, we trained Mistral 7B on simple dictionary key-value retrieval task (denoted as "sd") and observe a performance boost on MDQA and FLenQA. In this section we further investigate the model's performance if trained on other retrieval tasks. In particular, we consider multi-subkey dictionary key-value retrieval (denoted as "msd") and a variant of simple dictionary key-value retrieval (denoted as "sdvar") where multiple dictionaries have the *gold key*, but each *gold key* corresponds to a different *gold value* and we ask the model to report all *gold values* in ascending order of values. A example is shown in Figure 15 and the detailed algorithm is shown in Algorithm 5. For this experiment, we choose $\texttt{num_dict} = 63, l_{min} = r_{min} = 3, l_{max} = r_{max} = 4, \texttt{n_common_dicts} = 3$.

**Simple dictionary key-value retrieval variant (with an answer template)**

```
Do a task using the list of dictionaries below.

...
Dictionary [36] {240: 188, 542: 1885, 592: 747, 3183: 113}
...
Dictionary [57] {9230: 930, 240: 6240, 578: 627}
...
Dictionary [63] {457: 1914, 2551: 4180, 240: 7277, 973: 219}
...

Above is a list of dictionaries such that each key and value is an integer. The key
240 appears three times across different dictionaries with varying values. Please
find all three values associated with the key 240 and list them in ascending order of
the values. Answer in the following format:
Three values of key <gold_key_str> in ascending order of value: [<fill-in-value1>,
<fill-in-value2>, <fill-in-value3>].
```

```
Desired answer: Three values of key 240 in ascending order of value: [188, 6240,
7277].
```

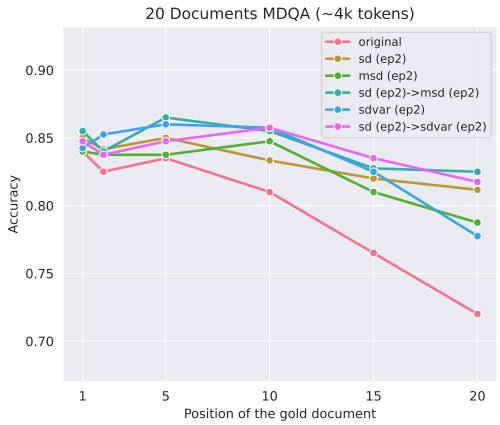
Figure 15: The task requires retrieving and sorting all values associated with the key 240 from a filtered list of dictionaries.

In addition, since simple dictionary key-value retrieval is a relatively simple task, we also consider the cases where we first train on "sd" and then train on "msd" or "sdvar". In particular, we consider the following cases (all datasets have size 350 where each sample has roughly 4K tokens): (1) "msd (ep2)", (2) "sd (ep2)→msd (ep2)", (3) "sdvar (ep2)", and (4) "sd (ep2)→sdvar (ep2)", where here "→" represents the training order. For example "sd (ep2)→msd (ep2)" means first train on "sd" for 2 epochs and then train on "msd" for 2 epochs. The results are shown in Figure 16 and Table 6. Interestingly, first training on "sd" (for 2 epochs) and then training on "msd" or "sdvar" (for 2 epochs) can boost the performance on MDQA and on FLenQA cot version. On the other hand, the model suffers from slightly more degradation on GSM8K benchmark (possibly due to the fact that we use integers as keys and values in the retrieval tasks).

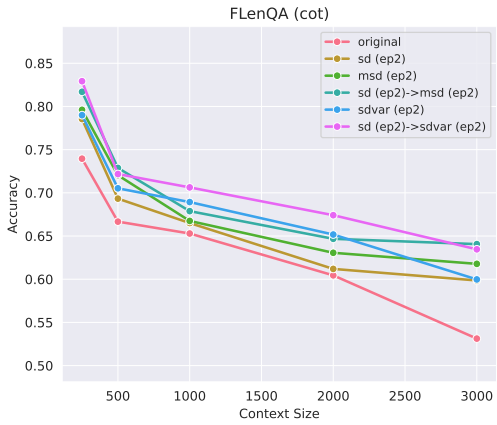
| Finetuning dataset | MMLU | HellaSwag | GSM8K | TriviaQA | NQ-Open |
|---|---|---|---|---|---|
| Original | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| msd (ep2) | 53.36 (−0.06) | 56.29 (−0.02) | 34.31 (−0.34) | 47.81 (+0.18) | 11.84 (+0.23) |
| sd (ep2)→msd (ep2) | 53.28 (−0.14) | 56.21 (−0.10) | 33.78 (−0.87) | 47.81 (+0.18) | 11.82 (+0.21) |
| sdvar (ep2) | 53.39 (−0.03) | 56.26 (−0.05) | 34.28 (−0.37) | 47.66 (+0.03) | 11.81 (+0.20) |
| sd (ep2)→advar (ep2) | 53.16 (−0.26) | 56.15 (−0.16) | 33.72 (−0.93) | 47.60 (−0.03) | 11.89 (+0.28) |

Table 6: Mistral 7B and finetuned versions' performance evaluated on general ability benchmarks. All numbers are reported in percentage.

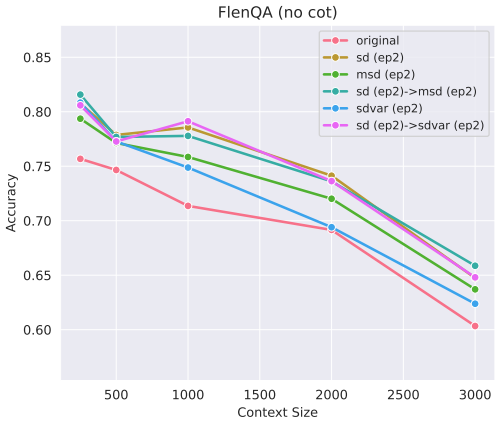
As a control, we also train the model with "IN2 (ep2)→IN2 (ep2)" and "MultidocQA (ep2)→MultidocQA (ep2)". Model's performance on MDQA, FLenQA and general benchmarks are shown in Figure 17 and Table 7.




(a) MDQA




(b) FLenQA with chain-of-thought prompting

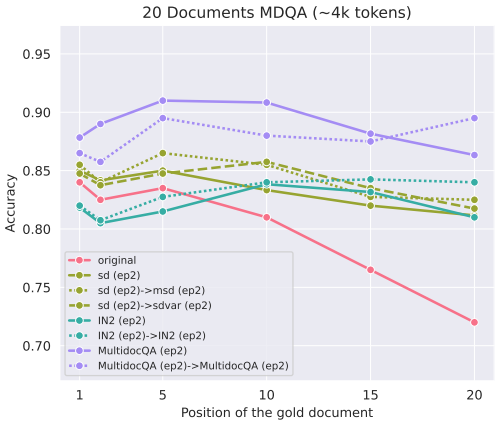



(c) FLenQA without chain-of-though prompting

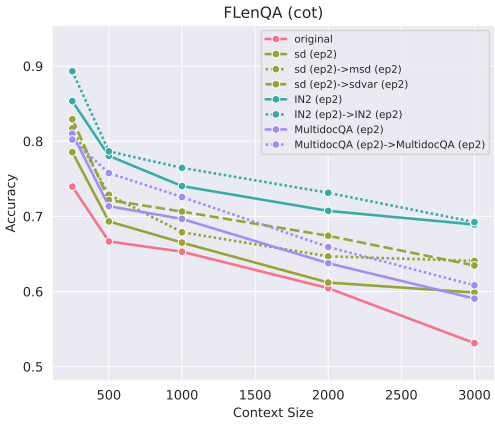
Figure 16: Performance of finetuned Mistral 7B with different retrieval tasks.

| Finetuning dataset | MMLU | HellaSwag | GSM8K | TriviaQA | NQ-Open |
|---|---|---|---|---|---|
| Original | 53.42 | 56.31 | 34.65 | 47.63 | 11.61 |
| sd (ep2)→msd (ep2) | 53.28 (−0.14) | 56.21 (−0.10) | 33.78 (−0.87) | 47.81 (+0.18) | 11.82 (+0.21) |
| sd (ep2)→advar (ep2) | 53.16 (−0.26) | 56.15 (−0.16) | 33.72 (−0.93) | 47.60 (−0.03) | 11.89 (+0.28) |
| IN2 (ep2)→IN2 (ep2) | 53.45 (+0.03) | 56.36 (+0.05) | 34.25 (−0.40) | 44.72 (−2.91) | 9.58 (−2.03) |
| MultidocQA (ep2)→MultidocQA (ep2) | 53.24 (−0.18) | 56.22 (−0.09) | 31.77 (−2.88) | 44.80 (−2.83) | 9.36 (−2.25) |

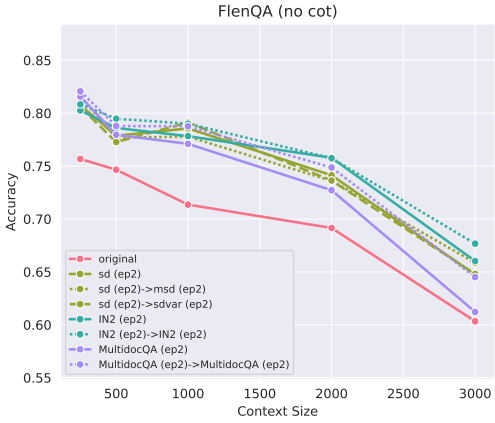
Table 7: Mistral 7B and finetuned versions' performance evaluated on general ability benchmarks. All numbers are reported in percentage.




(a) MDQA




(b) FLenQA with chain-of-thought prompting




(c) FLenQA without chain-of-though prompting

Figure 17: Performance of finetuned Mistral 7B with different retrieval tasks.

# C DETAILS ON GENERATING RETRIEVAL TASKS

In this section we provide the pseudocodes on generating retrieval tasks introduced in the paper: (1) simple dictionary key-value retrieval and (2) multi-subkey dictionary key-value retrieval. We will also provide the actual codebase.

## C.1 SIMPLE DICTIONARY KEY-VALUE RETRIEVAL

**Algorithm 1:** Gen_key_val

**Input:** min and max number of digits of key / value $\mathtt{r}_{min}, \mathtt{r}_{max}$, gold key `gold_key`
**Output:** `key` and `val` where `key` is different from `gold_key`

```
val ← randint(r_min, r_max)
while True do
    key ← randint(r_min, r_max)
    if key != gold_key then return key, val
```

**Algorithm 2:** Simple dictionary key-value retrieval

**Input:** Number of dictionaries `num_dict`; min and max length of each dictionary $l_{min}, l_{max}$; range of all keys / values $(\mathtt{r}_{min}, \mathtt{r}_{max})$
**Output:** A list of dictionaries `dicts`, the position of gold dictionary `gold_pos`, gold key `gold_key` and gold value `gold_val`.

```
Initialize gold_dict as an empty dictionary
gold_dict_len ← randint(l_min, l_max)
gold_pos ← randint(1, num_dict)
gold_key ← randint(r_min, r_max)
gold_val ← randint(r_min, r_max)
Add (gold_key, gold_val) key-value pair to gold_dict
for i = 1, ..., gold_dict_len − 1 do
    key, val ← Gen_key_val(r_min, r_max, gold_key)
    Add (key, val) key-value pair to gold_dict
Shuffle the order of gold_dict.
Initialize dicts to an empty array of dictionaries
for j = 1, ..., num_dict − 1 do
    Initialize dict as an empty dictionary
    dict_len ← randint(l_min, l_max)
    for k = 1, ..., dict_len do
        key, val ← Gen_key_val(r_min, r_max, gold_key)
        Add (key, val) key-value pair to dict
    Append dict to dicts
Insert gold_dict into dicts at position gold_pos
return dicts
```

### C.2 MULTI-SUBKEY KEY-VALUE RETRIEVAL

**Algorithm 3:** Gen_multikey_val

**Input:** range for all keys / values: $(\texttt{r}_{min}, \texttt{r}_{max})$, gold multi-key: `gold_key_tuple`, number of keys in each multi-key: `n_keys`, keys from `gold_key_tuple` that can be shared with the output `key_tuple`: `common_subkey`, probability of key sharing: $p_{\texttt{share}}$

**Output:** `key_tuple` and corresponding `val`

**assert** `len(common_subkey) < n_keys`
`val` $\leftarrow$ randint$(\texttt{r}_{min}, \texttt{r}_{max})$
**while** *True* **do**
  $\texttt{key}_i \leftarrow$ randint$(\texttt{r}_{min}, \texttt{r}_{max}),\ \forall i = 1, 2, \ldots, \texttt{n_keys}$
  $\texttt{key_tuple} = (\texttt{key}_1, \texttt{key}_2, \ldots, \texttt{key}_{\texttt{n_keys}})$
  **for** $i = 1, ..., \textit{len(common_subkey)}$ **do**
    With probability $p_{\texttt{share}}$ replace $\texttt{key}_i$ with $\texttt{common_subkey}_i$.
  Shuffle the elements of `key_tuple`.
  **if** *`key_tuple` and `gold_key_tuple` share at most `len(common_subkey)` keys* **then**
    **return** `key_tuple`, `val`

**Algorithm 4:** Multi-subkey dictionary retrieval

**Input:** Number of dictionaries: `num_dict`, min and max length of each dictionary: $l_{min}, l_{max}$, range of each key / value: $(\texttt{r}_{min}, \texttt{r}_{max})$, number of keys in each multikey: `n_keys`, max number of keys to share among `key_typle`'s: `n_common`, probability of key sharing between keys: $p_{\texttt{share}}$.

**Output:** A list of dictionaries `dicts`, the position of gold dictionary `gold_pos`, gold multi-key `gold_key_tuple` and gold value `gold_val`.

**Assert** `n_common < n_keys`.
Initialize `gold_dict` as an empty dictionary
`gold_dict_len` $\leftarrow$ randint$(l_{min}, l_{max})$
`gold_pos` $\leftarrow$ randint$(1, \texttt{num_dict})$
$\texttt{gold_key}_i =$ randint$(\texttt{r}_{min}, \texttt{r}_{max}),\ \forall i = 1, 2, \ldots, \texttt{n_keys}$
$\texttt{gold_key_tuple} = (\texttt{gold_key}_1, \texttt{gold_key}_2, \ldots, \texttt{gold_key}_{\texttt{n_keys}})$
`gold_val` $\leftarrow$ randint$(\texttt{r}_{min}, \texttt{r}_{max})$
Choose `n_common` random keys from `gold_key_tuple`.
Add (`gold_key_tuple`, `gold_val`) key-value pair to `gold_dict`
**for** $i = 1, \ldots, \textit{gold_dict_len} - 1$ **do**
  `key_tuple`, `val` $\leftarrow$ Gen_multikey_val$(\texttt{r}_{min}, \texttt{r}_{max}, \texttt{gold_key_tuple}, \texttt{n_keys}, p_{\texttt{share}})$.
  Add (`key_tuple`, `val`) multikey-value pair to `gold_dict`
Shuffle the order of `gold_dict`.
Initialize `dicts` to an empty list.
**for** $j = 1, \ldots, \textit{num_dict} - 1$ **do**
  Initialize `dict` as an empty dictionary
  `dict_len` $\leftarrow$ randint$(l_{min}, l_{max})$
  **for** $k = 1, \ldots, \textit{dict_len}$ **do**
    `key_tuple`, `val` $\leftarrow$ Gen_multikey_val$(\texttt{r}_{min}, \texttt{r}_{max}, \texttt{gold_key})$
    Add (`key_tuple`, `val`) multikey-value pair to `dict`
  Append `dict` to `dicts`
Insert `gold_dict` into `dicts` at position `gold_pos`
**return** `dicts`

# D SIMPLE DICTIONARY KEY-VALUE RETRIEVAL VARIANT

**Algorithm 5:** Simple dictionary key-value retrieval variant

**Input:** Number of dictionaries `num_dict`; min and max length of each dictionary $l_{min}, l_{max}$; range of all keys / values $(\mathtt{r}_{min}, \mathtt{r}_{max})$, number of dictionaries that contain `gold_key` (once): `n_common_dicts`

**Output:** A list of dictionaries `gold_dict_list`.

```
gold_key = randint(r_min, r_max)
Initialize gold_dict_list as an empty dictionary
for i = 1, ..., n_common_dicts do
    Initialize gold_dict as an empty dictionary
    gold_dict_len ← randint(l_min, l_max)
    gold_pos ← randint(1, num_dict)
    gold_val ← randint(r_min, r_max)
    Add (gold_key, gold_val) key-value pair to gold_dict
    for j = 1, ..., gold_dict_len − 1 do
        key, val ← Gen_key_val(r_min, r_max, gold_key)
        Add (key, val) key-value pair to gold_dict
    Shuffle the contents of gold_dict.
    Append gold_dict to gold_dict_list.
for i = 1, ..., num_dict - n_common_dicts do
    Initialize dict as an empty dictionary
    dict_len ← randint(l_min, l_max)
    for k = 1, ..., dict_len do
        key, val ← Gen_key_val(r_min, r_max, gold_key)
        Add (key, val) key-value pair to dict
    Append dict to gold_dict_list.
Shuffle dicts
return dicts
```

The example is shown in Figure 15.