Lost in the Middle: How Language Models Use Long Contexts

Nelson F. Liu[1*] **Kevin Lin**[2] **John Hewitt**[1] **Ashwin Paranjape**[3]
Michele Bevilacqua[3] **Fabio Petroni**[3] **Percy Liang**[1]
[1]Stanford University [2]University of California, Berkeley [3]Samaya AI
nfliu@cs.stanford.edu

Abstract

While recent language models have the ability to take long contexts as input, relatively little is known about how well they *use* longer context. We analyze language model performance on two tasks that require identifying relevant information within their input contexts: multi-document question answering and key-value retrieval. We find that performance is often highest when relevant information occurs at the beginning or end of the input context, and significantly degrades when models must access relevant information in the middle of long contexts. Furthermore, performance substantially decreases as the input context grows longer, even for explicitly long-context models. Our analysis provides a better understanding of how language models use their input context and provides new evaluation protocols for future long-context models.

1 Introduction

Language models have become an important and flexible building block in a variety of user-facing language technologies, including conversational interfaces, search and summarization, and collaborative writing. These models perform downstream tasks primarily via prompting: all relevant task specification and data to process is formatted as a textual context, and the model returns a generated text completion. These input contexts can contain thousands of tokens, especially when using language models on lengthy inputs (e.g., legal or scientific documents, conversation histories, etc.) or augmenting them with external information (e.g., relevant documents from a search engine, database query results, etc; Petroni et al., 2020; Ram et al., 2023; Shi et al., 2023; Mallen et al., 2023; Schick et al., 2023, *inter alia*).

Handling these use-cases requires language models to successfully operate over long sequences.

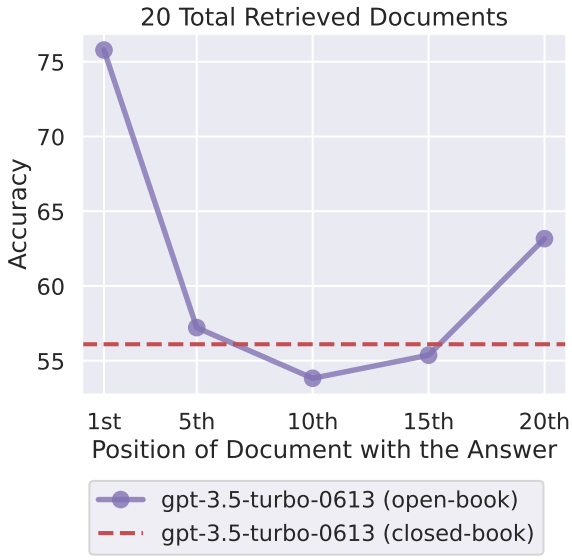

Figure 1: Changing the location of relevant information (in this case, the position of the passage that answers an input question) within the language model's input context results in a U-shaped performance curve—models are better at using relevant information that occurs at the very beginning or end of its input context, and performance degrades significantly when models must access and use information located in the middle of its input context. For example, GPT-3.5-Turbo's open-book performance on the multi-document question task when relevant information is placed in the middle of its input context is lower than its performance when predicting *without any documents* (i.e., the closed-book setting; 56.1%). See Figure 5 for full results.

Language models are generally implemented with Transformers, which scale poorly to long sequences (e.g., since self-attention complexity is quadratic with the input sequence length). As a result, language models are typically trained with relatively small context windows. Recent improvements in hardware (e.g., faster GPUs with more memory) and algorithms (Dai et al., 2019; Dao et al., 2022; Poli et al., 2023; Rubin and Berant, 2023, *inter alia*) have resulted in language models with larger context windows, but it remains unclear how these

[*]Work partially completed as an intern at Samaya AI.

extended-context language models make use of their input contexts when performing downstream tasks.

We empirically investigate this question via controlled experiments with a variety of state-of-the-art open (MPT-30B-Instruct, LongChat-13B (16K)) and closed (OpenAI's GPT-3.5-Turbo and Anthropic's Claude-1.3) language models in settings that require accessing and using information within an input context. We first experiment with multi-document question answering, which requires models to reason over provided documents to find relevant information and use it to answer a given question; this task mimics the retrieval-augmented generation setup underlying many commercial generative search and question answering applications (e.g., Bing Chat). We make controlled changes to the input context size and the position of the relevant information within the input context and study their effects on model performance. In particular, we can increase the input context length by adding more documents to the input context (akin to retrieving more documents in retrieval-augmented generation), and modify the position of the relevant information within the context by changing the order of the documents in the input context to place the relevant document at the beginning, middle or end of the context.

We observe a distinctive U-shaped performance, which can be clearly visualized in Figure 1, as we vary the position of the relevant information —language model performance is highest when relevant information occurs at the very beginning or end of its input context, and performance significantly degrades when models must access and use information in the middle of their input context (§3.3). For example, when relevant information is placed in the middle of its input context, GPT-3.5-Turbo's performance on the multi-document question task is lower than its performance when predicting *without any documents* (i.e., the closed-book setting; 56.1%). In addition, we find that model performance steadily degrades on longer contexts (§3.3), and that extended-context models are not necessarily better at using their input context (§3.3).

Given that language models struggle to retrieve and use relevant information in the multi-document question answering task, to what extent can language models even *retrieve* from their input contexts? We study this question with a synthetic key-value retrieval task, which is designed to be a minimal testbed for the basic ability to retrieve matching tokens from the input context. In this task, models are given a collection of JSON-formatted key-value pairs, and must return the value associated with a specific key. Similar to the multi-document QA task, the key-value retrieval task also admits controlled changes to the input context length (adding more key-value pairs) and the position of relevant information. We observe a similar U-shaped performance curve in this setting; many models struggle to simply retrieve matching tokens that occur in the middle of their input context.

To better understand why language models struggle to access and use information in the middle of their input contexts, we conduct preliminary investigations into the role of model architecture (decoder-only vs. encoder-decoder), query-aware contextualization, and instruction fine-tuning (§5). We find that encoder-decoder models are relatively robust to changes in the position of relevant information within their input context when evaluated on sequences within its training-time sequence length, but they show a U-shaped curve when evaluated on sequences longer than those seen during training (§5.1). In addition, query-aware contextualization (placing the query before *and* after the documents or key-value pairs) enables models to perform the synthetic key-value task perfectly, but minimally changes trends in multi-document QA (§5.2). Finally, even base language models (i.e., without instruction fine-tuning) show a U-shaped performance curve as we vary the position of relevant information in the input context.

Lastly, we perform a case study with retriever-reader models on open-domain question answering to better understand the trade-off between adding more information to an input context and increasing the amount of content that the model must reason over (§6)—in contrast to our controlled multi-document QA task, where the context always contains exactly one document that answers the question, none or many of the top k documents may contain the answer in the open-domain QA setting. When retrieving from Wikipedia to answer queries from NaturalQuestions-Open, we find that model performance saturates long before retriever recall levels off, indicating that models fail to effectively use additional retrieved documents—using more than 20 retrieved documents only marginally improves performance ($\sim$1.5% for GPT-3.5-Turbo

and ∼1% for claude-1.3).

Our analysis provides a better understanding of how language models use their input context and introduces new evaluation protocols for future long-context models. To facilitate further work on understanding and improving how language models use their input context, we release our code and evaluation data.[1]

2 Language Models

We study language models as functions that take a textual input context and return a textual output. Modern language models are most commonly implemented with Transformers (Vaswani et al., 2017). Transformer language models encode input contexts with self-attention, whose time and memory complexity is quadratic in the length of the input, limiting their application to very long sequences. As a result, they are generally pre-trained with relatively small amount of prior context (its *context window*), which accordingly also limits the maximum length of the language model's input contexts.

Increasing language model maximum context length. Recent advances in hardware (e.g., faster GPUs with more memory) and algorithms (e.g., FlashAttention; Dao et al., 2022) have driven a rapid increase in language model maximum context length. OpenAI's GPT-4 model (released in March 2023) has a maximum context window of 32K tokens; in May 2023, Claude's context window was expanded from 8K tokens to 100K tokens. In June 2023, OpenAI announced an extended-context version of its GPT-3.5-Turbo model, increasing its context from 4K to 16K tokens. A variety of open-source long context language models have also been recently released: MPT-30B has a maximum context length of 8K tokens, and LongChat-7B has a maximum context length of 16K tokens. Finally, a variety of recently-proposed architectures model sequences with millions of tokens, raising the potential of further dramatic increases in language model maximum context length (Gu et al., 2022; Fu et al., 2023; Poli et al., 2023; Yu et al., 2023, *inter alia*).

3 Multi-Document Question Answering

Our goal is to better understand how language models use their input context. To this end, we analyze model performance on multi-document question answering, which requires models to find relevant information within an input context and using it to answer the question. In particular, we make controlled changes to the length of the input context and the position of the relevant information and measure changes in task performance.

3.1 Experimental Setup

Our multi-document question answering task closely parallels the retrieval-augmented generation setup underlying commercial search and question answering applications (e.g., Bing Chat). In these experiments, the model inputs are (i) a question to answer and (ii) k documents (e.g., passages from Wikipedia), where *exactly one* the documents contains the answer to the question and $k-1$ "distractor" documents do not. Performing this task requires the model to access the document that contains the answer within its input context and use it to answer the question. Figure 2 presents an example.

We instantiate this task with data from the NaturalQuestions benchmark (Kwiatkowski et al., 2019), which contains historical queries issued to the Google search engine and human-annotated answers extracted from Wikipedia. Specifically, we first take queries from NaturalQuestions-Open (Lee et al., 2019), an open domain question answering benchmark that is derived from NaturalQuestions. Use use passages (chunks of at most 100 tokens) from Wikipedia as documents within our input contexts. For each of these queries, we need a document that contains the answer and $k-1$ distractor documents that do not contain the answer. To obtain a document that answers the question, we use the Wikipedia paragraph that contains the answer from the NaturalQuestions annotations. To collect $k-1$ distractor documents that do not contain the answer, we use the Contriever retrieval system (Izacard et al., 2021) to retrieve the $k-1$ Wikipedia chunks that are most relevant to the question and do not contain any of the NaturalQuestions-annotated answers.[2,3] In the input context, the distractor documents are presented in order of decreasing rele-

[1] nelsonliu.me/papers/lost-in-the-middle

[2] Ambiguity in NaturalQuestions-Open means that a small number of distractor passages may contain a reasonable answer. We additionally run experiments on subset of unambiguous questions, finding similar results and conclusions; see Appendix A.

[3] We also explored using random documents as distractors, see Appendix B for more details.

Input Context

```
Write a high-quality answer for the given question using only the provided search
results (some of which might be irrelevant).

Document [1](Title: Asian Americans in science and technology) Prize in physics for
discovery of the subatomic particle J/ψ. Subrahmanyan Chandrasekhar shared...
Document [2](Title: List of Nobel laureates in Physics) The first Nobel Prize in
Physics was awarded in 1901 to Wilhelm Conrad Röntgen, of Germany, who received...
Document [3](Title: Scientist) and pursued through a unique method, was essentially
in place. Ramón y Cajal won the Nobel Prize in 1906 for his remarkable...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Figure 2: Example of the multi-document question answering task, with an input context and the desired model answer. The relevant document for correctly answering the request is bolded within the input context.

Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: Asian Americans in science and
technology) ...
Document [2](Title: List of Nobel laureates in
Physics) ...
Document [3](Title: Scientist) ...
Document [4](Title: Norwegian Americans) ...
Document [5](Title: Maria Goeppert Mayer) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Figure 3: Modulating the input context length of the multi-document question answering example presented in Figure 2. Adding additional documents that do not contain the answer increases the length of the input context, but does not affect the desired output. The relevant document pair for correctly answering the request is bolded within the input context.

Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: List of Nobel laureates in
Physics) ...
Document [2](Title: Asian Americans in science and
technology) ...
Document [3](Title: Scientist) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Figure 4: Modulating the position of relevant information within the input context for the multi-document question answering example presented in Figure 2. Reordering the documents in the input context does not affect the desired output. The relevant document for correctly answering the request is bolded within the input context.

vance.[4]

To modulate the input context length in this task, we increase or decrease the number of retrieved documents that do not contain the answer (Figure 3). To modulate the position of relevant information within the input context, we adjust the order of the documents in the input context to change the position of the document that contains the answer (Figure 4).

Following Kandpal et al. (2022) and Mallen et al. (2023), we use accuracy as our primary evaluation metric, judging whether any of the correct answers (as taken from the NaturalQuestions annotations) appear in the predicted output. To prevent models from exploiting the metric by simply copying the documents from the input context, we strip model output beyond the first generated newline character. In practice, model responses are generally a single sentence or paragraph; generation is terminated (via producing an end-of-sequence token) without producing any newline characters.

Our experimental setup is similar to the needle-in-a-haystack experiments of Ivgi et al. (2023), who compare question answering performance when the relevant paragraph is placed (i) at the beginning of the input context or (ii) a random position within the input context. They find that encoder-decoder models have significantly higher performance when relevant information is placed at the start of the input context. In contrast, we study

[4]Since there might be a prior over "search results" appearing in ranked order, we explored randomly ordering the $k-1$ distractor documents and mentioning that the documents are randomly ordered in the task description, but found the same trends. See Appendix C for more details.

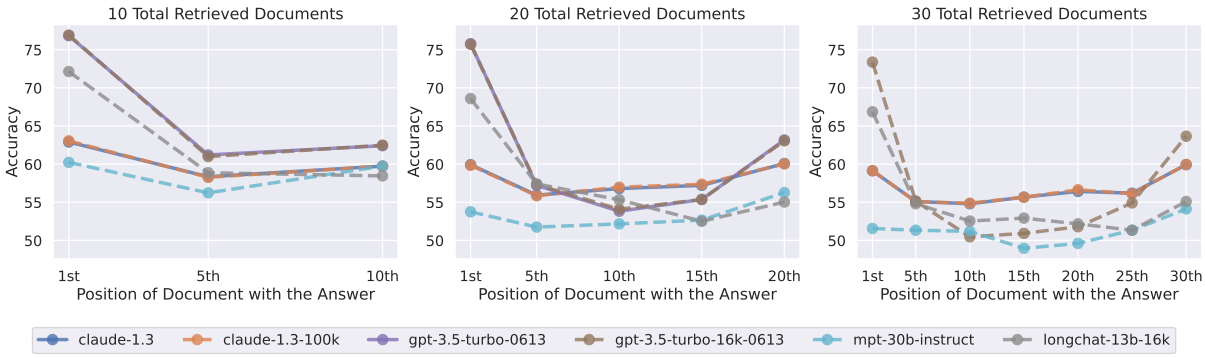

Figure 5: The effect of changing the position of relevant information (document containing the answer) on multi-document question answering performance. Lower positions are closer to the start of the input context. Performance is generally highest when relevant information is positioned at the very start or very end of the context, and rapidly degrades when models must reason over information in the middle of their input context.

finer-grained changes in the position of relevant information.

3.2 Models

We analyze several state-of-the-art open and closed models. We use greedy decoding when generating outputs and leave exploration of other decoding methods to future work. We use a standard set of prompts for each model (depicted in Figure 2). Appendix F tabulates input context lengths (number of tokens) for each model and experimental setting.

Open models. We experiment with MPT-30B-Instruct, which has a maximum context length of 8192 tokens. The model was initially pre-trained on 1 trillion tokens using 2048-token sequences, followed by an additional sequence length adaptation pre-training phase on 50B tokens using 8192-token sequences. MPT-30B-Instruct uses ALiBi (Press et al., 2022) to represent positional information. We also evaluate LongChat-13B (16K) (Li et al., 2023), which builds on LLaMA-13B (original maximum context window of 2048 tokens; Touvron et al., 2023) and extends its context window to 16384 tokens by using condensed rotary positional embeddings before fine-tuning with 16384-token sequences.

Closed models. We use the OpenAI API to experiment with GPT-3.5-Turbo and GPT-3.5-Turbo (16K).[5] GPT-3.5-Turbo has a maximum context length of 4K tokens, and GPT-3.5-Turbo (16K) is a version with an extended maximum context length of 16K tokens. We evaluate Claude-1.3 and Claude-1.3 (100K) with the Anthropic API; Claude-1.3 has a maximum context length of 8K tokens, and Claude-1.3 (100K) has an extended context length of 100K tokens.[6]

3.3 Results and Discussion

We experiment with input contexts containing 10, 20, and 30 documents (2.7K examples each). Figure 5 presents multi-document question answering performance when the position of relevant information within the input context. To better understand the realistic lower- and upper-bounds on performance, we also evaluate performance on the closed-book and oracle settings. In the closed-book setting, models are not given any documents in their input context, and must rely on their parametric memory to generate the correct answer. On the other hand, in the oracle setting, language models are given the single document that contains the answer and must use it to answer the question. GPT-3.5-Turbo and GPT-3.5-Turbo (16K) have the highest closed-book (55%) and oracle (88%) performance; see Appendix E for full closed-book and oracle results on all models.

Model performance is highest when relevant information occurs at the beginning or end of its input context. As the position of relevant information is changed, we see a distinctive U-shaped curve in model performance—models are much better at identifying and using relevant information that occurs at the very beginning and very

[5]We use the `0613` model revisions for all OpenAI API experiments.

[6]We also evaluate GPT-4 on a subset of multi-document QA experiments, finding similar results and trends as other models (though GPT-4 has higher absolute performance). Evaluating GPT-4 on the full multi-document QA and key-value retrieval experiments would cost upwards of $6000. See Appendix D for GPT-4 results and discussion.

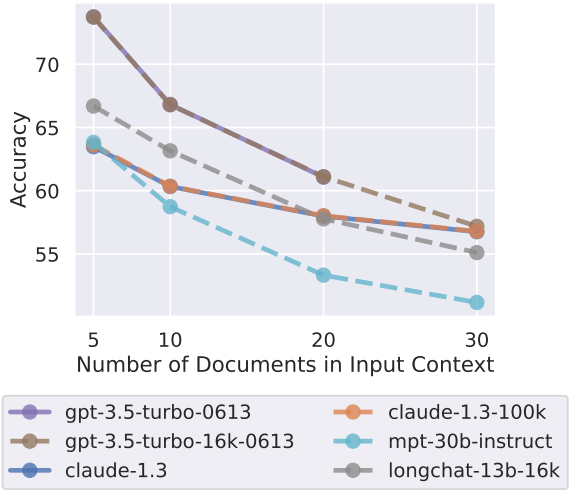

Figure 6: Language model performance (averaged across position of relevant information) on the multi-document question answering task decreases as the input context grows longer.

end of contexts, and suffer degraded performance when forced to use information within the middle of its input context. For example, GPT-3.5-Turbo's multi-document QA performance can drop by more than 20%—at its nadir, performance in 20- and 30-document settings is lower than performance without *any* input documents (i.e., closed-book performance; 56.1%). These results indicate that current models cannot effectively reason over their entire context window when performing downstream tasks, and that models have an easier time retrieving and using information at the very start or end of their input contexts.

Model performance substantially decreases as input contexts grow longer. On both tasks, model performance degrades as the contexts grow longer, indicating that models struggle to retrieve and use relevant information from long input contexts (Figure 6).

This trend continues when comparing models with their corresponding extended-context versions. For example, GPT-3.5-Turbo's lowest performance in the 20-document setting is 52.9% (when the document containing the answer is positioned 10th out of 20). The input contexts of the 30-document setting are too long for GPT-3.5-Turbo, but using its extended-context counterpart GPT-3.5-Turbo (16K) also results in performance decrease (49.5% when the relevant document is positioned 10th out of 30)—although extended-context models can process longer input contexts, they may not be better at reasoning over the information within its context window.

Extended-context models are not necessarily better at using input context. In settings where the input context fits in the context window of both a model and its extended-context counterpart, we see that performance between them is nearly identical. For example, the results for GPT-3.5-Turbo and GPT-3.5-Turbo (16K) are nearly superimposed (solid purple series and dashed brown series, respectively). These results indicate that models with longer maximum context windows are not necessarily better at using this extended context.

4 How Well Can Language Models Retrieve From Input Contexts?

Given that language models struggle to retrieve and use information from the middle of their input contexts in the multi-document question answering task, to what extent can they simply *retrieve* from input contexts? We study this question with a synthetic key-value retrieval task to isolate and study the basic ability of matching and retrieving relevant information from input contexts.

4.1 Experimental Setup

In our synthetic key-value retrieval task, the inputs are (i) a string-serialized JSON object with k key-value pairs, where each of the keys and values are unique, randomly-generated UUIDs and (ii) a particular key within the aforementioned JSON object. The goal is to return the value associated with the specified key. Thus, each JSON object contains one relevant key-value pair (where the value is to be retrieved), and $k - 1$ irrelevant "distractor" key-value pairs. Figure 7 provides an example input context and its corresponding desired output. We use accuracy as our evaluation metric, assessing whether the correct value appears in the predicted output.

Our synthetic key-value retrieval task is designed to provide a minimal testbed for the basic ability to retrieve matching tokens from an input context. This task shares similar goals with the Little Retrieval Test of Papailiopoulos et al. (2023) and the closely-related fine-grained line retrieval task of Li et al. (2023), but we explicitly seek to distill and simplify the task by removing as much natural language semantics as possible (using random UUIDs instead), since language features may

Input Context

```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

Figure 7: Example of the key-value retrieval task, with an input context and the desired model output. All keys and values are 128-bit UUIDs, and the goal of the task is to return the value associated with the specified key. The relevant key-value pair for correctly answering the request is bolded within the input context.

Input Context

```
Extract the value corresponding to the specified key in
the JSON object below.

JSON data:
{"2a8d601d-...-8ad825a74195": "bb3ba2a5-...-a88bb9fa7289",
 "a54e2eed-...-3624e77d6684": "d1ff29be-...-0cea716be3d4",
 "f9130258-...-232e92d369c9": "6fcd02c0-...-16464ce76a13",
 "56e00398-...-4cbdd6c87b53": "bf4700be-...-7ccd57c9df91",
 "85352c2d-...-9edbe756efca": "307d52f4-...-cdd939438915",
 "9f4a92b9-...-403f08dea695": "703a7ce5-...-5836ba5ec71c",
 "7202d68f-...-e352844671fe": "145e4450-...-d8e4576d9a8e",
 "1dc736e1-...-f3296b586348": "43da98d6-...-1544f95782a2",
 "dd52c4b0-...-7a167fbdf8b4": "c88ad889-...-c0f76b4afa42",
 "52a9c80c-...-4a4901bc2ac3": "b2f8ea3d-...-b9823991ebeb",
 "f4eb1c53-...-c2d50851a178": "d733b0d2-...-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Figure 8: Modulating the input context length of the key-value retrieval example presented in Figure 7. Adding random key-value pairs (128-bit UUIDs) increases length of the input context, but does not affect the desired output. The relevant key-value pair for correctly answering the request is bolded within the input context.

present potential confounders (e.g., because Transformer language models may have varying sensitivity to different linguistic features in their input context; O'Connor and Andreas, 2021).

To modulate the input context length in this task, we change the number of input JSON key-value pairs k by adding or removing random keys, changing the number of distractor key-value pairs (Figure 8). To modulate the position of relevant information within the input context, we change the position of the key to retrieve within the serialized JSON object (Figure 9).

4.2 Results and Discussion

Figure 10 presents key-value retrieval performance; We experiment with input contexts containing 75, 140, and 300 key-value pairs (500 examples each). We use the same set of models as the multi-

Input Context

```
Extract the value corresponding to the specified key in
the JSON object below.

JSON data:
{"9f4a92b9-...-403f08dea695": "703a7ce5-...-5836ba5ec71c",
 "2a8d601d-...-8ad825a74195": "bb3ba2a5-...-a88bb9fa7289",
 "a54e2eed-...-3624e77d6684": "d1ff29be-...-0cea716be3d4",
 "52a9c80c-...-4a4901bc2ac3": "b2f8ea3d-...-b9823991ebeb",
 "f4eb1c53-...-c2d50851a178": "d733b0d2-...-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Figure 9: Modulating the position of relevant information within the input context for the key-value retrieval example presented in Figure 7. Re-ordering the key-value pairs does not affect the desired output. All keys and values are random 128-bit UUIDs. The relevant key-value pair for correctly answering the request is bolded within the input context.

document question answering experiments, see §3.2 for more details.

Although the synthetic key-value retrieval task only requires identifying exact match within the input context, not all models achieve high performance—claude-1.3 and claude-1.3-100k do nearly perfectly on all evaluated input context lengths, but other models struggle, especially when retrieving keys from 140 or more key-value pairs.

The results on the key-value retrieval task have largely similar trends to the results on the multi-document question-answering task (excepting models with perfect performance on the key-value retrieval task). In particular, we see the U-shaped performance curve again; model performance is lowest when they must access key-value pairs in the middle of their input context. Furthermore, model performance in this setting generally also decreases on longer input contexts. LongChat-13B

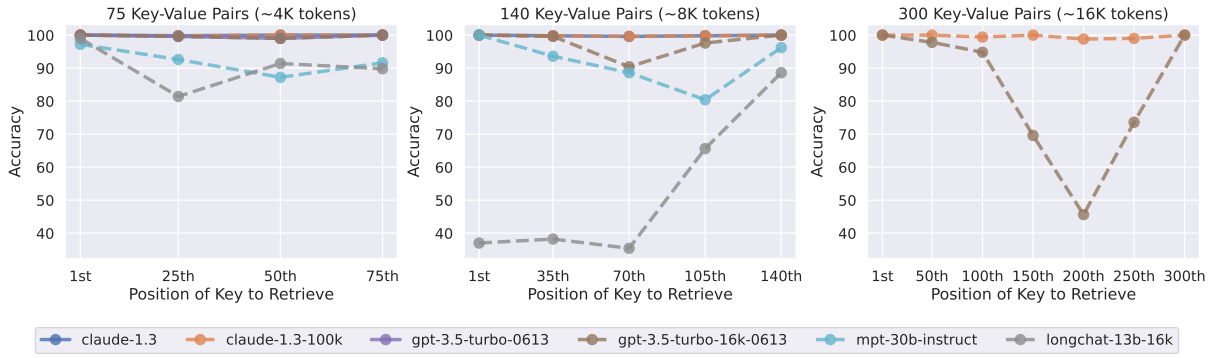

Figure 10: The effect of changing the input context length and the position of relevant information on key-value retrieval performance. Lower positions are closer to the start of the input context. Although some models are largely perfect on this synthetic task (e.g., claude-1.3 and claude-1.3), we see again that performance is often highest when relevant information is occurs at the very start or very end of the context, and rapidly degrades when models must retrieve from the middle of the input context. LongChat-13B (16K) in the 140 key-value setting is a notable outlier; when the relevant information is at the start of the input context, it tends to generate code to retrieve the key, rather than outputting the value itself.

(16K) in the 140 key-value setting is a notable outlier; when the relevant information is at the start of the input context, it tends to generate code to retrieve the key, rather than outputting the value itself.

5 Why Do Language Models Struggle To Use Their Entire Input Context?

Our multi-document question answering and key-value retrieval results show that language model performance degrades significantly when they must access relevant information in the middle of long input contexts. To better understand why, we perform some preliminary investigations into the role of model architecture (e.g., decoder-only vs. encoder-decoder), query-aware contextualization, and the effects of instruction fine-tuning.

5.1 Effect of Model Architecture

The open models we evaluate in §3 and §4 are all decoder-only models—at each timestep, they may only attend to prior tokens. To better understand the potential effects of model architecture on how language model use context, we compare decoder-only and encoder-decoder language models.

We experiment with Flan-T5-XXL (Raffel et al., 2020; Chung et al., 2022) and Flan-UL2 (Tay et al., 2023). Flan-T5-XXL is trained with a sequences of 512 tokens (encoder and decoder). Flan-UL2 is initially trained with sequences of 512 tokens (encoder and decoder), but is then pre-trained for an extra 100K steps with 1024 tokens (encoder and decoder), before instruction-tuning on sequences with 2048 tokens in the encoder and 512 tokens in the decoder. However, since these models use relative positional embeddings, they can (in principle) extrapolate beyond these maximum context lengths; Shaham et al. (2023) find that both models can perform well with sequences of 8K tokens.

Figure 11 juxtaposes the performance of decoder-only and encoder-decoder models. When Flan-UL2 is evaluated on sequences within its 2048 training-time context window, its performance is relatively robust to changes in the position of relevant information within the input context. When evaluated on settings with sequences longer than 2048 tokens, Flan-UL2 performance begins to degrade when relevant information is place in the middle. Flan-T5-XXL shows a similar trend, where longer input contexts result in a greater performance degradation when placing relevant information in the middle of the input context.

We speculate that encoder-decoder models may make better use of their context windows because their bidirectional encoder allows processing each document in the context of future documents, potentially enhancing relative importance estimation between documents.

5.2 Effect of Query-Aware Contextualization

Our experiments in §3 and §4 place the query (i.e., question to answer or key to retrieve) after the data to process (i.e., the documents or the key-value pairs). As a result, decoder-only models cannot attend to query tokens when contextualizing documents or key-value pairs, since the query only

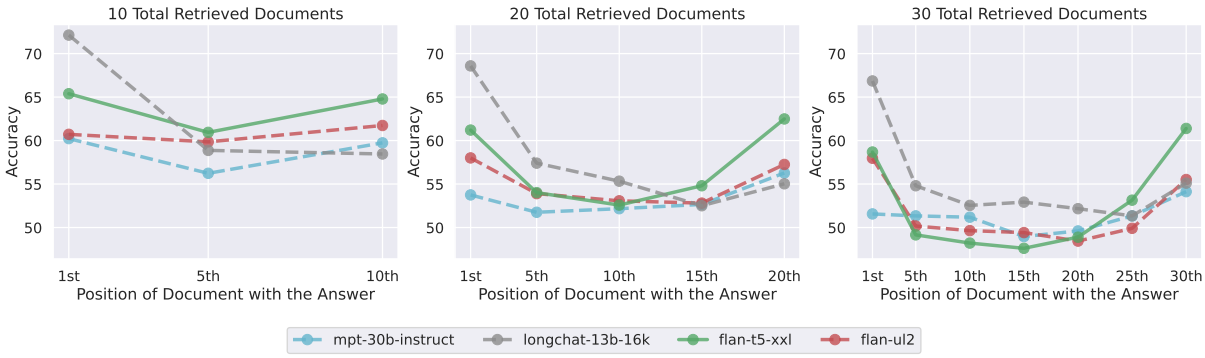

Figure 11: Encoder-decoder models (Flan-UL2 and Flan-T5-XXL) are relatively robust to changes in the position of relevant information within their input context when evaluated on sequences that are shorter than their encoder's training-time maximum sequence length (2048 and 512 tokens, respectively). However, when these models are evaluated on sequences longer than those seen during training (20- and 30-document settings), they also exhibit a U-shaped performance curve, where performance is much higher when the relevant information occurs at the beginning or end of the input context as opposed to the middle.

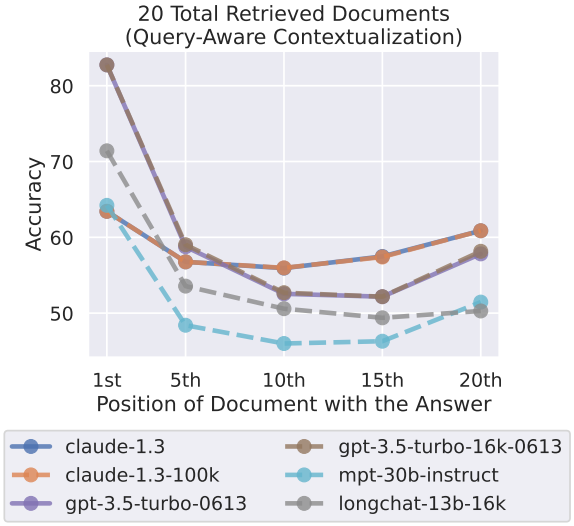

Figure 12: Query-aware contextualization (i.e., placing the question before *and* after the documents in the input context) improves multi-document QA performance when relevant information occurs at the very beginning, but slightly decreases performance otherwise.

appears at the end of the prompt and decoder-only models can only attend to prior tokens at each timestep. On the other hand, encoder-decoder models use a bidirectional encoder to contextualize input contexts, and seem to be more robust to changes in the position of relevant information in their input context—can use this intuition to also improve the performance of decoder-only models by placing the query before *and* after the data, enabling query-aware contextualization of documents (or key-value pairs)?

We find that query-aware contextualization dramatically improves performance on the key-value retrieval task. For example, GPT-3.5-Turbo (16K) (with query-aware contextualization) achieves perfect performance when evaluated with 300 key-value pairs. In contrast, without query-aware contextualization, it achieves a lowest performance of 45.6% in the same setting (Figure 10).

In contrast, query-aware contextualization minimally affects performance trends in the multi-document question answering task. In particular, it improves performance when the relevant information is located at the very beginning of the input context, but slightly decreases performance in other settings.

5.3 Effect of Instruction-Tuning

All of the models that we evaluated in §3 and §4 are instruction-tuned—after their initial pre-training, they undergo supervised fine-tuning on a dataset of instructions and responses. In this supervised instruction-tuning data, the task specification and/or instruction is commonly placed at the beginning of the input context, which might lead instruction-tuned language models to place more weight on the start of the input context.

To better understand the potential effects of instruction-tuning on how language models use long input contexts, we compare the multi-document question answering performance of MPT-30B-Instruct against its base model (i.e., before instruction fine-tuning) MPT-30B. We use the same experimental setup as §3.

Figure 13 compares the multi-document QA performance of MPT-30B and MPT-30B-Instruct as a function of the position of the relevant in-

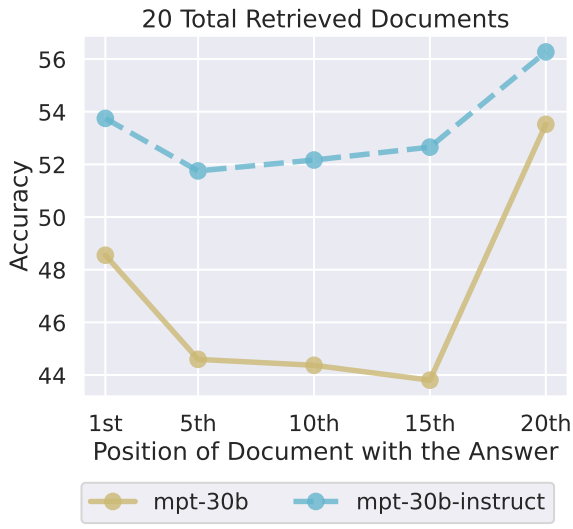

Figure 13: Multi-document QA performance of MPT-30B-Instruct compared against its base model (i.e., before instruction fine-tuning) MPT-30B. Both models have a U-shaped performance curve, where performance is much higher when relevant information occurs at the start or end of the input context, indicating that the instruction tuning process itself is not necessarily responsible for these performance trends.

formation in the input context. Surprisingly, we see that both MPT-30B and MPT-30B-Instruct exhibit a U-shaped performance curve, where performance is highest when relevant information occurs at the very beginning or very end of the context. Although the absolute performance of MPT-30B-Instruct is uniformly higher than that of MPT-30B, their overall performance trends are quite similar.

These observations complement prior work, which found that (non-instruction-tuned) language models are biased towards recent tokens (i.e., the end of the input context; Khandelwal et al., 2018; Press et al., 2021). This recency bias has been observed in past work when evaluating models on next-word prediction of contiguous text, a setting where language models minimally benefit from long-range information (Sun et al., 2021). In contrast, our results show that language models are capable of using longer-range information (i.e., the beginning of the input context) when prompted with instruction-formatted data. We hypothesize that language models learn to use these contexts from similarly-formatted data that may occur in webtext seen during pre-training, e.g., StackOverflow questions and answers.

6 Is More Context Is Always Better? A Case Study With Open-Domain QA

In practical settings, there is often a trade-off with increased the input context length—providing the instruction-tuned language model with more information may help improve downstream task performance, but also increases the amount of content that the model must reason over. Even if a language model can take in 16K tokens, is it actually beneficial to provide 16K tokens of context? The answer to this question is downstream task-specific since it depends on the marginal value of the added context and the model's ability to effectively use long input contexts, but we perform a case study with open-domain question answering on NaturalQuestions-Open to better understand this trade-off.

We use models in a standard retriever-reader setup. A retrieval system (Contriever, fine-tuned on MS-MARCO) takes an input query from NaturalQuestions-Open and returns k documents from Wikipedia. To condition instruction-tuned language models on these retrieved documents, we simply include them in the prompt. We evaluate retriever recall and reader accuracy (whether any of the annotated answers appear in the predicted output) as a function of the number of retrieved documents k. We use a subset of NaturalQuestions-Open where the long answer is a paragraph (as opposed to a table or a list).

Figure 14 presents open-domain QA results. We see that reader model performance saturates long before retriever performance levels off, indicating that readers are not effectively using the extra context. Using more than 20 retrieved documents only marginally improves reader performance ($\sim$1.5% for GPT-3.5-Turbo and $\sim$1% for Claude-1.3), while significantly increasing the input context length (and thus latency and cost). These results, coupled with the observation that models are better at retrieving and using information at the start or end of the input contexts, suggest that effective reranking of retrieved documents (pushing relevant information closer to the start of the input context) or ranked list truncation (returning fewer documents when necessary; Arampatzis et al., 2009) may be promising directions for improving how language-model-based readers use retrieved context.

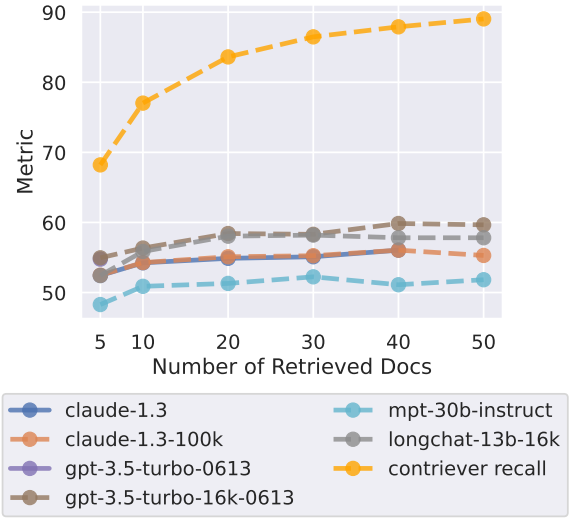

Figure 14: Retriever recall and model performance as a function of the number of retrieved documents. Model performance saturates long before retriever recall saturates, indicating that the models have difficulty making use of the extra retrieved documents.

7 Related Work

7.1 Long-context language models

There is a rich line of work in designing performant language models with cheaper scaling than Transformers in the context length. Many lines of work pursue Transformer variants with attention modifications like recurrence (Dai et al., 2019), factorizing attention into computationally less intensive approximations (Beltagy et al., 2020; Zaheer et al., 2020), or low-rank approximations (Wang et al., 2020; Peng et al., 2021); see Tay et al. (2022) for a comprehensive overview. Dao et al. (2022) instead provide a faster exact attention by a carefully-crafted IO-aware CUDA kernel. Separately, there are attempts to do away with attention entire to remove quadratic sequence length complexity, often through convolution and/or linear RNNs, e.g., in RWKV (Peng, 2023), S4 (Gu et al., 2022), or Hyena (Poli et al., 2023).

Many prior efforts evaluate perplexity on a diverse web corpus as a proxy for the ability to process long contexts; this work shows that precise knowledge access on long contexts may be an added challenge. However, a variety of work has proposed benchmarks for long-text understanding. Tay et al. (2021) propose the Long Range Arena, which evaluates long-context models on a variety of natural language, visual reasoning, and synthetic tasks. However, only two of its constituent tasks involve natural language, which limits its applicability to evaluating long-context capabilities of pre-trained language models. In contrast, the SCROLLS benchmark (Shaham et al., 2022) and its zero-shot extension ZeroSCROLLS (Shaham et al., 2023) evaluate model performance on a variety of NLP tasks that require understanding long input contexts (e.g., summarization and question answering over long documents).

7.2 How do language models use context?

The pioneering work of Khandelwal et al. (2018) showed that small LSTM language models make increasingly coarse use of longer-term context; Sankar et al. (2019) found similar results in dialogue models. In a similar vein, Daniluk et al. (2017) find that attentive LSTM language models tend to mainly use recent history. Petroni et al. (2020) were among the first to demonstrate the potential of combining context from an information retrieval system with a pretrained language models for unsupervised question answering. O'Connor and Andreas (2021) found that many information-destroying operations had marginal effects on Transformer LMs' predictions. Krishna et al. (2022) found that long-context neural generation in modestly-sized Transformer language models degenerates because models fail to properly condition on long context. Finally, studying long-context models, Sun et al. (2021) found that longer contexts improves prediction of only a few tokens, an empirical finding consistent with the theory of Sharan et al. (2018), who showed that sequence distributions with bounded mutual information necessarily lead to marginal *average* prediction benefits from increasingly long context.

Qin et al. (2023) analyze how efficient Transformers perform on a variety of long-context downstream NLP tasks, finding that long-context transformers are recency-biased and do not effectively use long-range context. Furthermore, they also observe that query-aware contextualization can improve performance, although their analysis focuses on fine-tuned models with bidirectional encoders (while we primarily study zero-shot prompting with decoder-only language models).

7.3 The serial-position effect

The U-shaped curve we observe in this work has a connection in psychology known as the *serial-position effect* (Ebbinghaus, 1913; Murdock Jr, 1962), that states that in free-association recall

of elements from a list, humans tend to best remember the first and last elements of the list. The serial-position effect plays a role in understanding how humans develop short- and long-term memory. Observing a serial-position-like effect in LLMs is perhaps surprising, since the self-attention mechanisms underlying Transformer LLMs is technically equally capable of retrieving any token from their contexts.

8 Conclusion

We empirically study how language models use long input contexts via a series of controlled experiments on two tasks that require identifying and using relevant information in-context: multi-document question answering and key-value retrieval. We find that language models often struggle to use information in the middle of long input contexts, and that performance decreases as the input context grows longer. We conduct a preliminary investigation of the role of (i) model architecture, (ii) query-aware contextualization, and (iii) instruction-tuning to better understand how each of these factors might affect how language models use context. Finally, we conclude with a practical case study of open-domain question answering, finding that the performance of language model readers saturates far before retriever recall. Our results and analysis provide a better understanding of how language models use their input context and provides new evaluation protocols for future long-context models.

Acknowledgments

We thank Sewon Min for her help with the AmbigQA dataset. In addition, we thank Eric Wallace and Sang Michael Xie for feedback and discussions that helped improve this work. This work was supported by the Stanford Center for Research on Foundation Models (CRFM), by OpenAI via an API credits grant to the Stanford CRFM, and by Anthropic via the Claude academic access program.

References

Avi Arampatzis, Jaap Kamps, and Stephen Robertson. 2009. Where to stop reading a ranked list? threshold optimization using truncated score distributions. In *Proc. of SIGIR*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. ArXiv:2004.05150.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. ArXiv:2210.11416.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of ACL*.

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In *Proc. of ICLR*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. ArXiv:2205.14135.

Hermann Ebbinghaus. 1913. Memory: A contribution to experimental psychology. *H. A. Ruger & C. E. Bussenius, Trans.*

Daniel Y. Fu, Tri Dao, Khaled Kamal Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. 2023. Hungry hungry hippos: Towards language modeling with state space models. In *Proc. of ICLR*.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *Proc. of ICLR*.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient long-text understanding with short-text models. *Transactions of the Association for Computational Linguistics*, 11:284–299.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. ArXiv:2112.09118.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proc. of EACL*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large language models struggle to learn long-tail knowledge. ArXiv:2211.08411.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proc. of ACL*.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. 2022. RankGen: Improving text generation with large ranking models. In *Proc. of EMNLP*.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proc. of ACL*.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, , and Hao Zhang. 2023. How long can open-source LLMs truly promise on context length?

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proc. of ACL*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proc. of EMNLP*.

Bennet B. Murdock Jr. 1962. The serial position effect of free recall. *Journal of experimental psychology*, 64(5):482.

Joe O'Connor and Jacob Andreas. 2021. What context features can Transformer language models use? In *Proc. of ACL*.

Dimitris Papailiopoulos, Kangwook Lee, and Jyyong Sohn. 2023. A little retrieval test for large language models. https://github.com/anadim/the-little-retrieval-test.

Bo Peng. 2023. RWKV-LM. https://github.com/BlinkDL/RWKV-LM.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *Proc. of ICLR*.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Proc. of AKBC*.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *Proc. of ICML*.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *Proc. of ICLR*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. In *Proc. of ACL*.

Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The NLP task effectiveness of long-range transformers. In *Proc. of EACL*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. ArXiv:2302.00083.

Ohad Rubin and Jonathan Berant. 2023. Long-range language modeling with self-retrieval. ArXiv:2306.13421.

Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. In *Proc. of ACL*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. ArXiv:2305.14196.

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 *Proc. of EMNLP*.

Vatsal Sharan, Sham Kakade, Percy Liang, and Gregory Valiant. 2018. Prediction with a short memory. In *Proc. of STOC*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. REPLUG: Retrieval-augmented black-box language models. ArXiv:2301.12652.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do long-range language models actually use long-range context? In *Proc. of EMNLP*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *Proc. of ICLR*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6).

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: Unifying language learning paradigms. ArXiv:2205.05131.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. ArXiv:2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. MEGABYTE: Predicting million-byte sequences with multiscale Transformers. ArXiv:2305.07185.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for longer sequences. In *Proc. of NeurIPS*, volume 33.

A Ambiguity in Multi-Document QA Distractor Documents

Following past work on NaturalQuestions-Open (Izacard et al., 2021; Izacard and Grave, 2021, *inter alia*), we use a Wikipedia dump from late 2018 as our retrieval corpus. However, this standard Wikipedia dump has a small amount of temporal mismatch with the data in NaturalQuestions.

For example, consider the question "what nfl team does robert griffin iii play for". The NaturalQuestions annotated answer is "currently a free agent". However, the Wikipedia retrieval corpus contains the information that he plays for the "Baltimore Ravens", since he was released from the team between the Wikipedia dump's timestamp and the NaturalQuestions annotation process.

We use the ambiguity annotations of Min et al. (2020) to create a subset unambiguous questions. Experiments on this unambiguous subset of the data show similar results and conclusions as the experiments on the full questions collection (Figure 15).

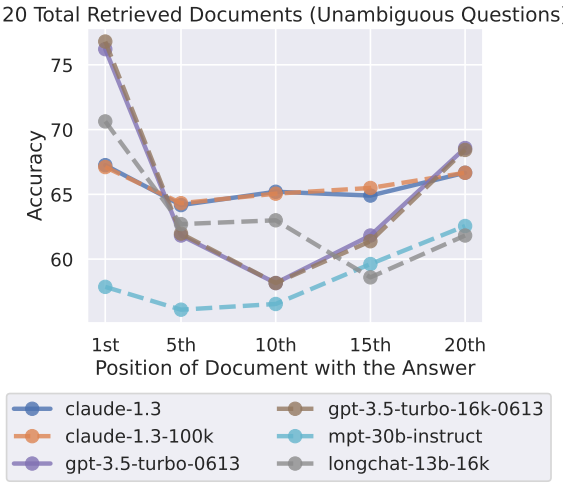

Figure 15: Language model performance on a unambiguous subset of questions.

B Random Distractors in Multi-Document QA

We also run multi-document question answering experiments with random Wikipedia documents as distractors, which allows us to ablate the impact of retrieved distractors (hard negatives). Note that in this setting, the the document containing the answer can often be identified with simple heuristics (e.g., lexical overlap with the query). Figure 16 presents the results of this experiment. Although all models have higher absolute accuracy in this setting, they surprisingly still struggle to reason over their entire input context, indicating that their performance degradation is not solely due to an inability to identify relevant documents.

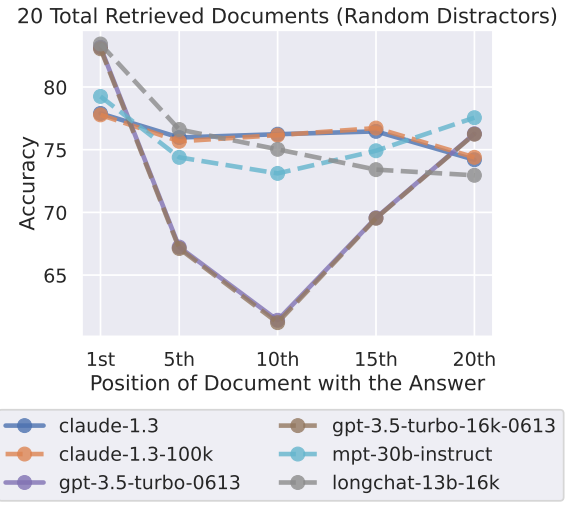

Figure 16: Language model performance on multi-document QA when using random distractors, rather than retrieved distractors.

C Randomizing Distractor Order in Multi-Document QA

Our prompt instructs the language model to use the provided search results to answer the question. There may be a prior in the pre-training or instruction-tuning data to treat search results as sorted by decreasing relevance (i.e., the documents near the beginning of the input context are more likely to be useful than those at the end). To validate that our conclusions are not simply a byproduct of this bias, we run experiments with the modified instruction "Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant). The search results are ordered randomly." In addition, we randomly shuffle the $k-1$ distractor documents.

Figure 17 presents the results of this experiment. We continue to see a U-shaped performance curve, with performance degrading when language models must use information in the middle of their input contexts. Comparing the results in §3.3 with those when randomizing the distractor order and mentioning such in the prompt, we see that randomization slightly decreases performance when the relevant information is at the very beginning of the context, and slightly increases performance

when using information in the middle and end of the context.

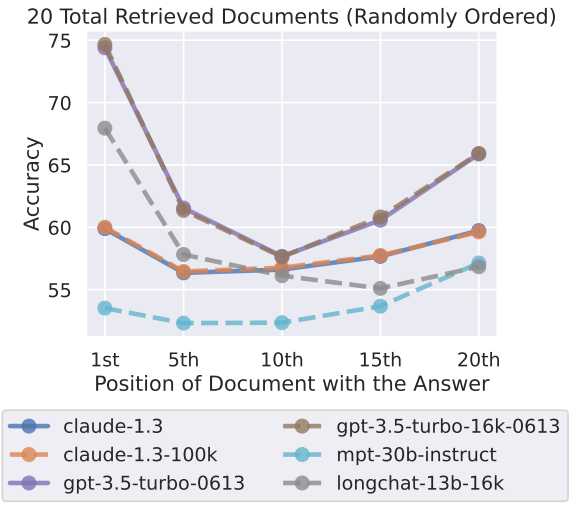

Figure 17: Language model performance when randomizing the order of the distractors (rather than presenting them in order of decreasing relevance) and mentioning as such in the prompt.

D GPT-4 Performance

We evaluate GPT-4 on a subset of 500 random multi-document QA examples with 20 total documents in each input context (Figure 18). GPT-4 achieves higher absolute performance than any other language model, but still shows a U-shaped performance curve—its performance is highest when relevant information occurs at the very start or end of the context, and performance degrades when it must use information in the middle of its input context.

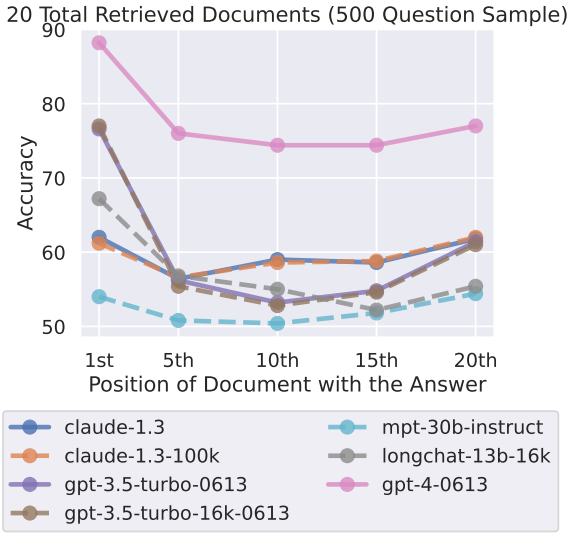

Figure 18: Although GPT-4 has higher absolute performance than other models, its performance still degrades when relevant information occurs in the middle of the input context.

E Closed-book and Oracle Performance

Table 1 presents language model performance on the closed-book and oracle settings for multi-document question answering. In the closed-book setting, language models are not given any documents in their input context, and must rely on their parametric memory to generate the correct answer. In the oracle setting, language models are given the single document that contains the answer, and must use it to answer the question. This represents an upper-bound on task performance.

Model	Closed-Book	Oracle
LongChat-13B (16K)	35.0%	83.4%
MPT-30B-Instruct	31.5%	81.9%
GPT-3.5-Turbo	56.1%	88.3%
GPT-3.5-Turbo (16K)	56.0%	88.6%
Claude-1.3	48.3%	76.1%
Claude-1.3 (100K)	48.2%	76.4%

Table 1: Closed-book and oracle accuracy of language models on the multi-document question answering task.

F Token Counts

Table 2, Table 3, and Table 4 present the average and maximum number of tokens in each of the input contexts for all experimental settings. Note that MPT-30B and MPT-30B-Instruct use the same tokenizer, GPT-3.5-Turbo and GPT-3.5-Turbo (16K) use the same tokenizer, and Claude-1.3 and Claude-1.3 (100K) use the same tokenizer. Furthermore, the Claude-1.3 tokenizer is the same as the GPT-3.5-Turbo tokenizer, modulo some additional special tokens that do not appear in our data. As a result, the token counts for these two model families is the same in our experimental settings.

	Closed-Book		Oracle	
	avg ± stdev	max	avg ± stdev	max
LongChat-13B (16K)	55.6 ± 2.7	70	219.7 ± 48.5	588
MPT-30B	43.5 ± 2.2	58	187.9 ± 41.8	482
GPT-3.5-Turbo	15.3 ± 2.2	29	156.0 ± 41.8	449
Claude-1.3	15.3 ± 2.2	29	156.0 ± 41.8	449

Table 2: Token count statistics for each of the evaluated models on the closed-book and oracle multi-document question answering settings.

	10 docs		20 docs		30 docs	
	avg ± stdev	max	avg ± stdev	max	avg ± stdev	max
LongChat-13B (16K)	1749.9 ± 112.4	2511	3464.6 ± 202.3	4955	5181.9 ± 294.7	7729
MPT-30B	1499.7 ± 88.5	1907	2962.4 ± 158.4	3730	4426.9 ± 230.5	5475
GPT-3.5-Turbo	1475.6 ± 86.5	1960	2946.2 ± 155.1	3920	4419.2 ± 226.5	6101
Claude-1.3	1475.6 ± 86.5	1960	2946.2 ± 155.1	3920	4419.2 ± 226.5	6101

Table 3: Token count statistics for each of the evaluated models on each of the document question answering settings.

	75 KV pairs		140 KV pairs		300 KV pairs	
	avg ± stdev	max	avg ± stdev	max	avg ± stdev	max
LongChat-13B (16K)	5444.5 ± 19.1	5500	10072.4 ± 24.1	10139	21467.3 ± 35.9	21582
MPT-30B	4110.5 ± 23.8	4187	7600.9 ± 31.1	7687	16192.4 ± 46.6	16319
GPT-3.5-Turbo	3768.7 ± 25.6	3844	6992.8 ± 34.1	7088	14929.4 ± 50.7	15048
Claude-1.3	3768.7 ± 25.6	3844	6992.8 ± 34.1	7088	14929.4 ± 50.7	15048

Table 4: Token count statistics for each of the evaluated models on each of the key-value (KV) retrieval settings.

G Full Multi-Document Question Answering Results

This section tabulates model performance when evaluated on the multi-document QA task with varying numbers of documents (Figure 5). "Index n" indicates performance when the document with the answer occurs at position $n + 1$, where lower indices are closer to the start of the input context. For example, index 0 refers to performance when the document with the answer is placed at the very start of the context (i.e., first amongst all documents).

G.1 10 Total Retrieved Documents

Model	Index 0	Index 4	Index 9
Claude-1.3	62.9%	58.3%	59.7%
Claude-1.3 (100K)	63.1%	58.3%	59.7%
GPT-3.5-Turbo	76.8%	61.2%	62.4%
GPT-3.5-Turbo (16K)	76.9%	61.0%	62.5%
MPT-30B-Instruct	60.2%	56.2%	59.7%
LongChat-13B (16K)	72.1%	58.9%	58.5%

Table 5: Model performance when evaluated on the multi-document QA task with 10 total retrieved documents.

G.2 20 Total Retrieved Documents

Model	Index 0	Index 4	Index 9	Index 14	Index 19
Claude-1.3	59.9%	55.9%	56.8%	57.2%	60.1%
Claude-1.3 (100K)	59.8%	55.9%	57.0%	57.4%	60.0%
GPT-3.5-Turbo	75.8%	57.2%	53.8%	55.4%	63.2%
GPT-3.5-Turbo (16K)	75.7%	57.3%	54.1%	55.4%	63.1%
MPT-30B-Instruct	53.7%	51.8%	52.2%	52.7%	56.3%
LongChat-13B (16K)	68.6%	57.4%	55.3%	52.5%	55.0%

Table 6: Model performance when evaluated on the multi-document QA task with 20 total retrieved documents.

G.3 30 Total Retrieved Documents

Model	Index 0	Index 4	Index 9	Index 14	Index 19	Index 24	Index 29
Claude-1.3	59.1%	55.1%	54.8%	55.7%	56.4%	56.2%	59.9%
Claude-1.3 (100K)	59.1%	55.1%	54.9%	55.7%	56.6%	56.1%	60.0%
GPT-3.5-Turbo (16K)	73.4%	55.1%	50.5%	50.9%	51.8%	54.9%	63.7%
MPT-30B-Instruct	51.6%	51.3%	51.2%	49.0%	49.6%	51.3%	54.1%
LongChat-13B (16K)	66.9%	54.8%	52.5%	52.9%	52.2%	51.3%	55.1%

Table 7: Model performance when evaluated on the multi-document QA task with 30 total retrieved documents.