Lost in the Middle: How Language Models Use Long Contexts

Nelson F. Liu[1*] **Kevin Lin**[2] **John Hewitt**[1] **Ashwin Paranjape**[3,4]
Michele Bevilacqua[3] **Fabio Petroni**[3] **Percy Liang**[1]

[1]Stanford University, USA [2]University of California, Berkeley, USA
[3]Samaya AI, UK [4]Samaya AI, USA
nfliu@cs.stanford.edu

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

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 the performance of language models on two tasks that require identifying relevant information in their input contexts: multi-document question answering and key-value retrieval. We find that performance can degrade significantly when changing the position of relevant information, indicating that current language models do not robustly make use of information in long input contexts. In particular, we observe 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, 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 language 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 (Shuster et al., 2022; Thoppilan et al., 2022; Lee et al., 2022, *inter alia*). These models perform downstream tasks primarily via prompting: all relevant task specification and data to process is formatted as a textual input context, and the model returns a generated text completion. These input contexts can contain thousands of tokens, especially when language models are used to process long documents (legal or scientific documents, conversation histories, etc.) or when language models are augmented with external information (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. Existing language models are generally implemented with Transformers (Vaswani et al., 2017), which require memory and compute that increases quadratically in sequence length. As a result, Transformer language models were often trained with relatively small context windows (between 512 and 2048 tokens). 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 (e.g., 4096, 32K, and even 100K tokens), but it remains unclear how these 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. In particular, our experiments make controlled changes to the input context size and the position of the relevant information within the input context and study their effects on language model performance. If language models can robustly use information within long input contexts, then their performance should be *minimally affected* by the position of the relevant information in the input context.

*Work partially completed as an intern at Samaya AI.

Transactions of the Association for Computational Linguistics, vol. 12, pp. 157–173, 2024. https://doi.org/10.1162/tacl_a_00638
Action Editor: Luke Zettlemoyer. Submission batch: 8/2023; Revision batch: 10/2023; Published 2/2024.
© 2024 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

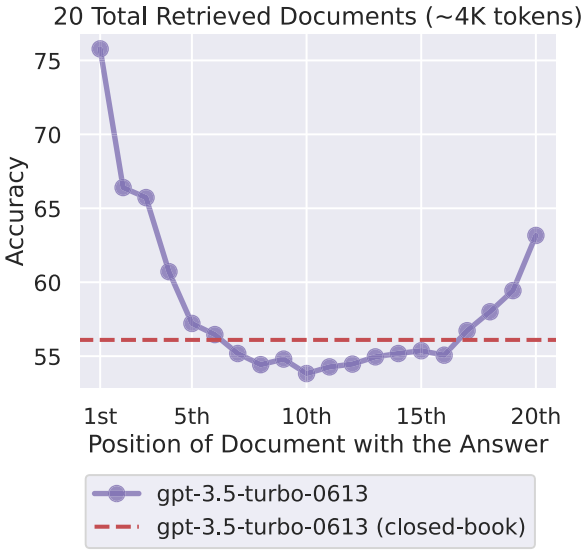

Figure 1: Changing the location of relevant information (in this case, the position of the passage that answers an input question) within the language model's input context results in a U-shaped performance curve—models are better at using relevant information that occurs at the very beginning (primacy bias) or end of its input context (recency bias), and performance degrades significantly when models must access and use information located in the middle of its 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). In this setting, we control (i) the input context length by changing the number of documents in the input context (akin to retrieving more or less documents in retrieval-augmented generation), and (ii) control the position of the relevant information within the input context by changing the order of the documents to place the relevant document at the beginning, middle or end of the context.

We find that changing the position of relevant information in the input context can substantially affect model performance, indicating that current language models do not robustly access and use information in long input contexts. Furthermore, we observe a distinctive U-shaped performance curve (Figure 1); language model performance is highest when relevant information occurs at the very beginning (primacy bias) or end of its input context (recency bias), and performance significantly degrades when models must access and use information in the middle of their input context (§2.3). 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%). Furthermore, we find that models often have identical performance to their extended-context counterparts, indicating that extended-context models are not necessarily better at using their input context (§2.3).

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 admits controlled changes to the input context length (adding more key-value pairs) and the position of relevant information. Although some models perform the synthetic key-value retrieval task perfectly, other models struggle to simply retrieve matching tokens that occur in the middle of their input context and continue to exhibit a U-shaped performance curve.

To better understand why language models struggle to robustly access and use information in their input contexts, we study the role of model architecture (decoder-only vs. encoder-decoder), query-aware contextualization, and instruction fine-tuning (§4). We find that:

- Encoder-decoder models are relatively robust to changes in the position of relevant information within their input context, but only when evaluated on sequences within its training-time sequence length. When evaluated on sequences longer than those seen during training, we observe a U-shaped performance curve (§4.1).

- Query-aware contextualization (placing the query before *and* after the documents or

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

key-value pairs) enables near-perfect performance on the synthetic key-value task, but minimally changes trends in multi-document QA (§4.2).

- 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.

Our results indicate that prompting language models with longer input contexts is a trade-off—providing the language model with more information may help it perform the downstream task, but it also increases the amount of content that the model must reason over, potentially decreasing accuracy. To better understand this trade-off in practice, we perform a case study with retriever-reader models on open-domain question answering (§5). 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 saturates, indicating that current models fail to effectively use additional retrieved documents—using 50 documents instead of 20 retrieved documents only marginally improves performance ($\sim$1.5% for GPT-3.5-Turbo and $\sim$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 claim that a language model can robustly use information within long input contexts, it is necessary to show that its performance is minimally affected by the position of the relevant information in the input context (e.g., minimal difference in best- and worst-case performance). To facilitate further work on understanding and improving how language models use their input context, we release our code and evaluation data.[1]

2 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 use 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.

2.1 Experimental Setup

In the multi-document question answering task, the model inputs are (i) a question to answer and (ii) k documents (e.g., passages from Wikipedia), where *exactly one* of the documents contains the answer to the question and $k - 1$ ''distractor'' documents do not. 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 NaturalQuestions-Open (Lee et al., 2019; Kwiatkowski et al., 2019), which contains historical queries issued to the Google search engine, coupled with human-annotated answers extracted from Wikipedia. In particular, we take the 2655 queries where the annotated long answer is a paragraph (as opposed to a list or a table). We use passages (chunks of at most 100 tokens) from Wikipedia as documents within our input contexts. For each of the 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 a retrieval system (Contriever, fine-tuned on MS-MARCO; Izacard et al., 2021) to retrieve the $k-1$ Wikipedia chunks that are most relevant to the query and do not contain any of the NaturalQuestions-annotated answers.[2,3] In the input context, the distractor

[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.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

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 document containing the answer is bolded within the input context here for clarity.

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 3: Modulating the position of relevant information within the input context for the multi-document question answering example presented in Figure 2. Re-ordering the documents in the input context does not affect the desired output.

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 4: Modulating the input context length of the multi-document question answering example presented in Figure 2. Adding documents that do not contain the answer increases the length of the input context, but does not affect the desired output.

documents are presented in order of decreasing relevance.[4]

To modulate the position of relevant information within the input context, we adjust the order of the documents to change the position of the document that contains the answer (Figure 3). 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 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.

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 or (ii) a random position within the input. 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 finer-grained changes in the position of relevant information.

2.2 Models

We analyze several state-of-the-art open and closed language models. We use greedy decoding

[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.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

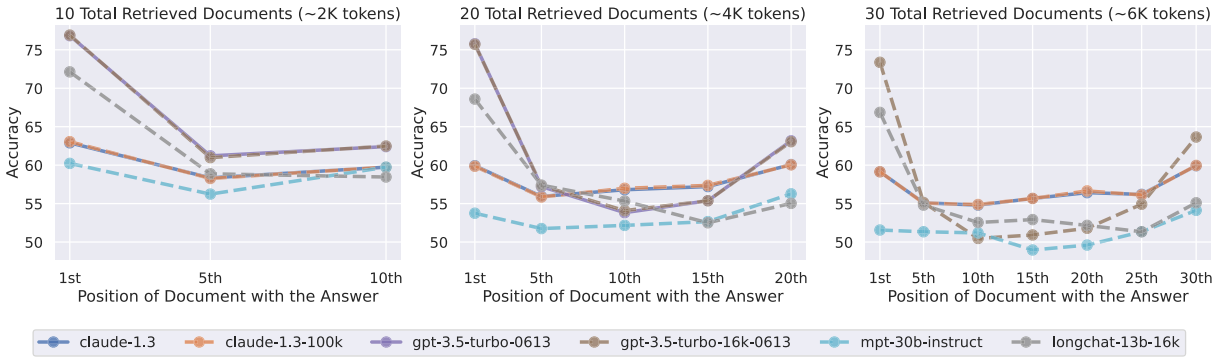

Figure 5: The effect of changing the position of relevant information (document containing the answer) on multi-document question answering performance. Lower positions are closer to the start of the input context. Performance is highest when relevant information occurs at the very start or end of the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

when generating outputs and leave exploration of other decoding methods to future work. We use a standard set of prompts for each model (Figure 2).

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 50 billion 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 extends the LLaMA-13B (Touvron et al., 2023a) context window from 2048 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]

[5]We use the `0613` OpenAI model versions.

[6]We also evaluate GPT-4 (8K) 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.

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.

2.3 Results and Discussion

We experiment with input contexts containing 10, 20, and 30 total documents. Figure 5 presents multi-document question answering performance when varying the position of relevant information within the input context. To contextualize model performance, we also evaluate on the closed-book and oracle settings (Table 1). 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.

Model performance is highest when relevant information occurs at the beginning or end of its input context. As illustrated in Figure 5, changing the position of relevant information in the input context leads to substantial decreases in model performance. In particular, we see a distinctive U-shaped performance curve—models

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

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 6: Example of the key-value retrieval task, with an input context and the desired model output. Given a key, the goal is to return the associated value. All keys and values are 128-bit UUIDs. The relevant key-value pair for answering the query is bolded here within the input context for clarity.

are often much better at using relevant information that occurs at the very beginning (primacy bias) and very end of contexts (recency bias), 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%—in the worst case, 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 prompted for downstream tasks.

Extended-context models are not necessarily better at using input context. When 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 10- and 20-document settings both fit in the context window of GPT-3.5-Turbo and GPT-3.5-Turbo (16K), and we observe that their performance as a function of position of relative information is nearly superimposed (solid purple and dashed brown series in Figure 5). These results indicate that extended-context models are not necessarily better than their non-extended counterparts at using their input context.

3 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, which is designed to provide a minimal testbed for the basic ability to retrieve matching tokens from an input context.

3.1 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 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 returned), and $k-1$ irrelevant ''distractor'' key-value pairs. Figure 6 provides an example input context and its corresponding desired output. We again measure accuracy by evaluating whether the correct value appears in the predicted output.

Our synthetic key-value retrieval task shares similar goals with the Little Retrieval Test of Papailiopoulos et al. (2023) and the 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 present potential confounders. For example, Transformer language models may have varying sensitivity to different linguistic features in their input (O'Connor and Andreas, 2021).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

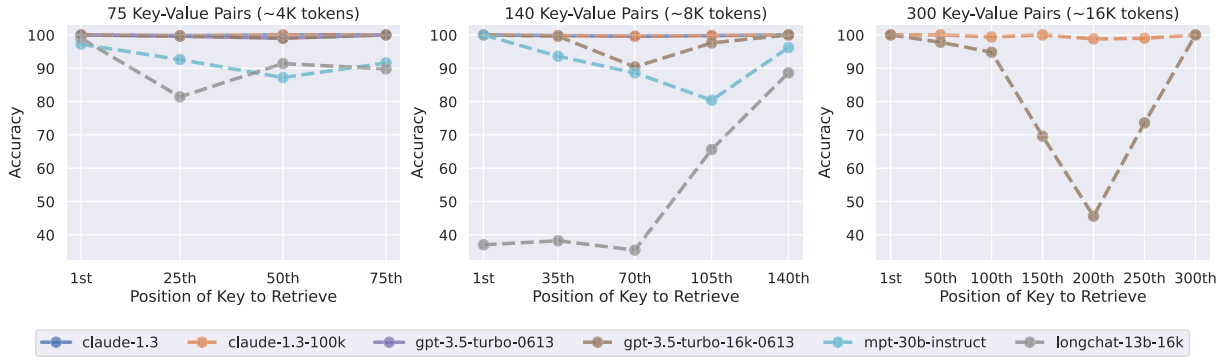

Figure 7: The effect of changing the input context length and the position of relevant information on key-value retrieval performance. Lower positions are closer to the start of the input context. Although some models show perfect accuracy on this synthetic task (e.g., Claude-1.3 and Claude-1.3 (100K)), we see again that performance is often highest when relevant information is occurs at the very start or end of the context, and rapidly degrades when models must retrieve from the middle of the input context.

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. To modulate the input context length, 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.

3.2 Results and Discussion

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-document question answering experiments, see §2.2 for more details.

Figure 7 presents key-value retrieval performance. Claude-1.3 and Claude-1.3 (100K) do nearly perfectly on all evaluated input context lengths, but other models struggle, especially when contexts have 140 or 300 key-value pairs—although the synthetic key-value retrieval task only requires identifying exact match within the input context, not all models achieve high performance.

Similar to our multi-document QA results, GPT-3.5-Turbo, GPT-3.5-Turbo (16K), and MPT-30B-Instruct have the lowest performance when they must access key-value pairs in the middle of their input context. LongChat-13B (16K) exhibits a different trend in the 140 key-value setting; we qualitatively observe that when relevant information is placed at the start of the input context, LongChat-13B (16K) tends to generate code to retrieve the key, rather than outputting the value directly.

4 Why Are Language Models Not Robust to Changes in the Position of Relevant Information?

Our multi-document question answering and key-value retrieval results show that language models struggle to robustly access and use information in long input contexts, since performance degrades significantly when changing the position of relevant information. To better understand why, we perform some preliminary investigations into the role of model architecture (decoder-only vs. encoder-decoder), query-aware contextualization, and instruction fine-tuning.

4.1 Effect of Model Architecture

The open models we evaluated 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 fine-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

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

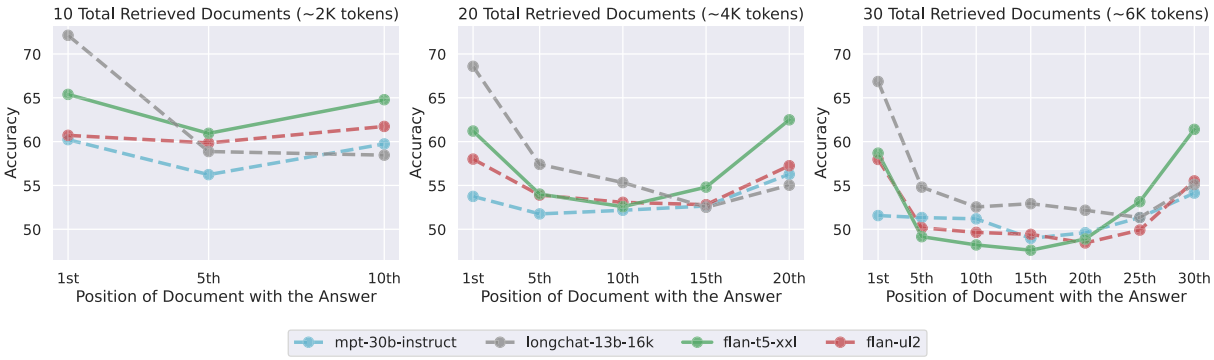

Figure 8: When encoder-decoder models (Flan-UL2 and Flan-T5-XXL) evaluated on sequences that are *shorter* than their encoder's training-time maximum sequence length (2048 and 512 tokens, respectively), they are relatively robust to changes in the position of relevant information within their input context (left subplot). In contrast, when these models are evaluated on sequences *longer* than those seen during training (center and right subplots), we observe a U-shaped performance curve—performance is higher when relevant information occurs at the beginning or end of the input context, as opposed to the middle of the input context.

these maximum context lengths; Shaham et al. (2023) find that both models can perform well with sequences of up to 8K tokens.

Figure 8 compares the performance of decoder-only and encoder-decoder models. When Flan-UL2 is evaluated on sequences within its 2048-token training-time context window (Figure 8; left subplot), its performance is relatively robust to changes in the position of relevant information within the input context (1.9% absolute difference between best- and worst-case performance). When evaluated on settings with sequences longer than 2048 tokens (Figure 8; center and right), Flan-UL2 performance begins to degrade when relevant information is placed 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 hypothesize 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 improving relative importance estimation between documents.

4.2 Effect of Query-Aware Contextualization

Our multi-document QA and key-value retrieval experiments 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 appears at the end of the prompt and decoder-only models can only attend to prior tokens at each timestep. In contrast, encoder-decoder models (which seem more robust to changes in the position of relevant information; §4.1) use a bidirectional encoder to contextualize input contexts—can we use this observation to improve 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—all models achieve near-perfect performance on the 75, 140, and 300 key-value pair settings. 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, the worst-case performance is 45.6% (Figure 7). Despite the significant impact on key-value retrieval performance, query-aware contextualization minimally affects performance trends in the multi-document question answering task (Figure 9); it slightly improves performance when the relevant information is located at the very beginning of the input context, but slightly decreases performance in other settings.

4.3 Effect of Instruction Fine-Tuning

The models we evaluated are all instruction fine-tuned—after their initial pre-training, they

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

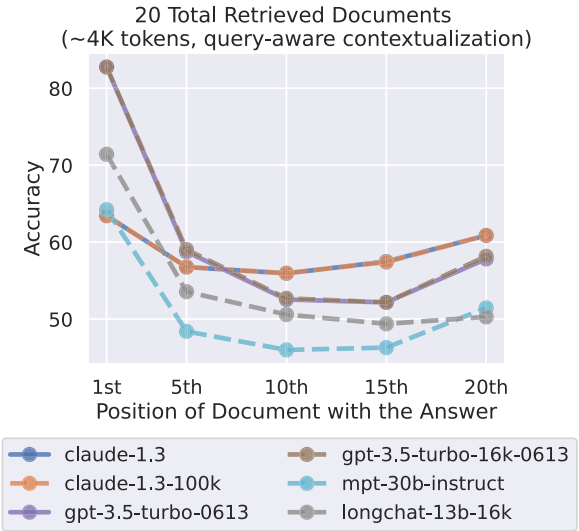

Figure 9: Query-aware contextualization (placing the query before *and* after the documents) does not substantially improve robustness of language models to changing the position of relevant information in multi-document QA; performance slightly increases when relevant information occurs at the very beginning, but otherwise slightly decreases.

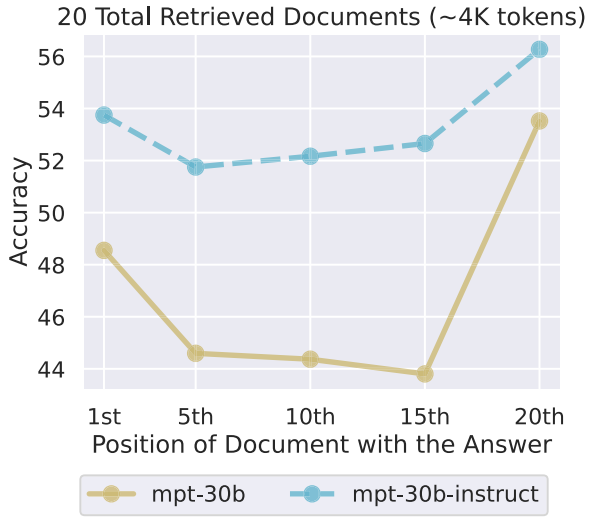

Figure 10: Multi-document QA performance of MPT-30B-Instruct compared against its base model (i.e., before instruction fine-tuning) MPT-30B. Both models have a U-shaped performance curve, where performance is much higher when relevant information occurs at the start or end of the input context, indicating that the instruction fine-tuning process itself is not necessarily responsible for these performance trends.

undergo supervised fine-tuning on a dataset of instructions and responses. The task specification and/or instruction is commonly placed at the beginning of the input context in supervised instruction fine-tuning data, which might lead instruction fine-tuned language models to place more weight on the start of the input context. To better understand the potential effects of instruction fine-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 §2.

Figure 10 compares the multi-document QA performance of MPT-30B and MPT-30B-Instruct as a function of the position of the relevant information 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 similar. We also observe that instruction fine-tuning slightly reduces the worst-case performance disparity from nearly 10% between the base model best- and worst-case performance to around 4%.

These observations complement prior work, which found that non-instruction fine-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 non-instruction fine-tuned language models learn to use these long contexts from similarly formatted data that may occur in Internet text seen during pre-training, e.g., StackOverflow questions and answers.

To better understand the effect of additional fine-tuning and model scale, we also experimented with Llama-2 models of varying sizes (7B, 13B, and 70B) with and without additional supervised fine-tuning and reinforcement learning from human feedback (Appendix E). We find that the U-shaped performance curve only appears in sufficiently large language models (with or without

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

additional fine-tuning)—the 7B Llama-2 models are solely recency-biased, while the 13B and 70B models exhibit a U-shaped performance curve. In addition, we see that the Llama-2 supervised fine-tuning and reinforcement learning from human feedback procedure slightly mitigates the positional bias in smaller models (13B, akin to trends shown when comparing MPT-30B and MPT-30B-Instruct), but minimally affects trends on larger models (70B).

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

Our results indicate that prompting language models with longer input contexts is a trade-off—providing the language model with more information may help it perform the downstream task, but it also increases the amount of content that the model must reason over, potentially decreasing accuracy. 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 ultimately 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 in existing language models.

We use language 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 the k documents from Wikipedia with the highest relevance score. To condition 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 11 presents retriever recall and open-domain QA results. We see that reader model performance saturates long before retriever performance saturates, indicating that readers are not effectively using the extra context. Using more than 20 retrieved documents only marginally improves reader performance (∼1.5% for GPT-3.5-Turbo and ∼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 often 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 (retrieving fewer documents when appropriate; Arampatzis et al., 2009) may be promising directions for improving how language-model-based readers use retrieved context.

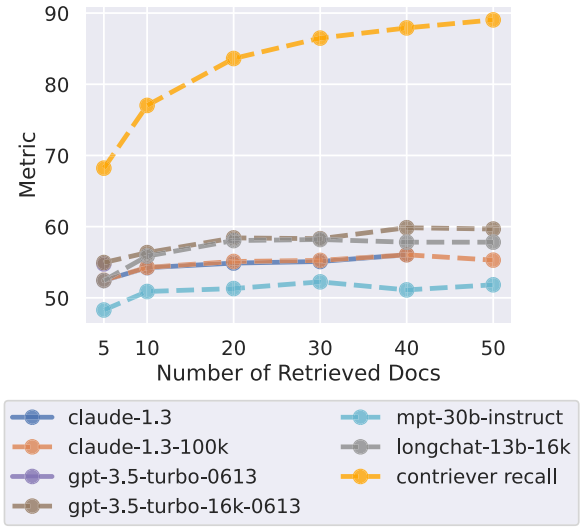

Figure 11: Retriever recall and model performance as a function of the number of retrieved documents. Model performance saturates long before retriever recall, indicating that the models have difficulty making use of the extra retrieved documents.

6 Related Work

6.1 Long-Context Language Models

There is much prior 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). 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 entirely to remove quadratic sequence length complexity, often through convolution and/or linear

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

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.

6.2 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.

6.3 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 language models is perhaps surprising, since the self-attention mechanisms underlying Transformer language models is technically equally capable of retrieving any token from their contexts.

7 Conclusion

We empirically study how language models use long input contexts via a series of controlled experiments. We show that language model performance degrades significantly when changing the position of relevant information, indicating that models struggle to robustly access and use information in long input contexts. In particular, performance is often lowest when models must use information in the middle of long input contexts. We conduct a preliminary investigation of the role of (i) model architecture, (ii) query-aware contextualization, and (iii) instruction fine-tuning to better understand how they 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 would like to thank Luke Zettlemoyer, who served as our TACL action editor, and the anonymous reviewers for their comments and feedback. We also thank Claudiu Leoveanu-Condrei, Megan Leszczynski, Dmytro Okhonko, Maithra Raghu, Eric Wallace, and Sang Michael Xie for feedback and discussions that helped improve this work. Further, we are grateful to Sewon Min for her help with the AmbigQA dataset. 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 *Proceedings of SIGIR*. https://doi.org/10.1145/1571941.1572031

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

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 *Proceedings of ACL*. https://doi.org/10.18653/v1/P19-1285

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In *Proceedings 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.* https://doi.org/10.1037/10011-000

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *Proceedings 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. https://doi.org/10.1162/tacl_a_00547

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 *Proceedings of EACL*. https://doi.org/10.18653/v1/2021.eacl-main.74

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 *Proceedings of ACL*. https://doi.org/10.18653/v1/P18-1027

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. 2022. RankGen: Improving text generation with large ranking models. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2022.emnlp-main.15

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. https://doi.org/10.1162/tacl_a_00276

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of ACL*. https://doi.org/10.18653/v1/P19-1612

Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a human-AI collaborative writing dataset for exploring language model capabilities. In *Proceedings of CHI*. https://doi.org/10.1145/3491102.3502030

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

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of ACL*. https://doi.org/10.18653/v1/2023.acl-long.546

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.emnlp-main.466

Bennet B. Murdock Jr. 1962. The serial position effect of free recall. *Journal of Experimental Psychology*, 64(5):482. https://doi.org/10.1037/h0045106

Joe O'Connor and Jacob Andreas. 2021. What context features can Transformer language models use? In *Proceedings of ACL*. https://doi.org/10.18653/v1/2021.acl-long.70

Dimitris Papailiopoulos, Kangwook Lee, and Jy-yong 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 *Proceedings 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 *Proceedings 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 *Proceedings of ICML*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. In *Proceedings of ACL*. https://doi.org/10.18653/v1/2021.acl-long.427

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

Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The NLP task effectiveness of long-range transformers. In *Proceedings of EACL*. https://doi.org/10.18653/v1/2023.eacl-main.273

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*. https://doi.org/10.1162/tacl_a_00605

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 *Proceedings of ACL*. https://doi.org/10.18653/v1/P19-1004

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*. https://doi.org/10.18653/v1/2023.findings-emnlp.536

Vatsal Sharan, Sham Kakade, Percy Liang, and Gregory Valiant. 2018. Prediction with a short memory. In *Proceedings of STOC*. https://doi.org/10.18653/v1/K18-3013

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*.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Y-Lan Boureau, Melanie Kambadur, and Jason Weston. 2022. BlenderBot 3: A deployed conversational agent that continually learns to responsibly engage. *ArXiv:2208.03188*.

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

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*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language models for dialog applications. *ArXiv:2201.08239*.

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. 2023a. LLaMA: Open and efficient foundation language models. *ArXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv:2307.09288*.

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 *Proceedings of NeurIPS*.

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

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 *Proceedings of NeurIPS*.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

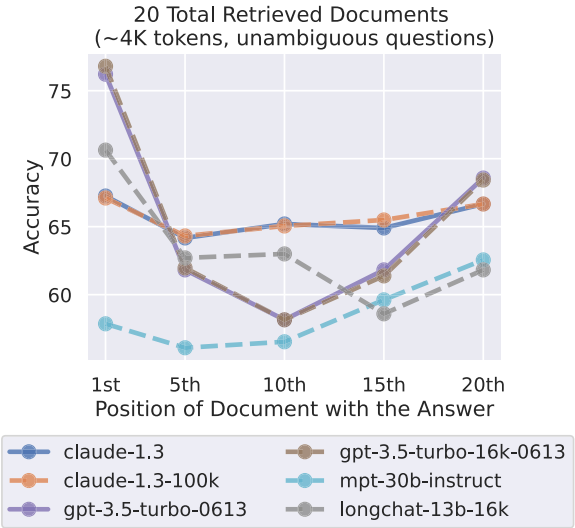

Figure 12: Language model performance on a unambiguous subset of questions.

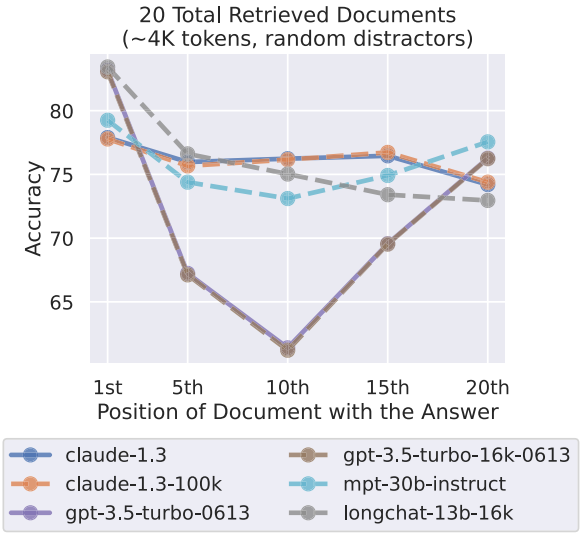

Figure 13: Language model performance on multi-document QA when using random distractors, rather than retrieved distractors.

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 NaturalQuestions annotations.

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 12).

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 document containing the answer can often be identified with simple heuristics (e.g., lexical overlap with the query). Figure 13 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.

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 fine-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 14 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 §2.3 with those when randomizing the distractor order and

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

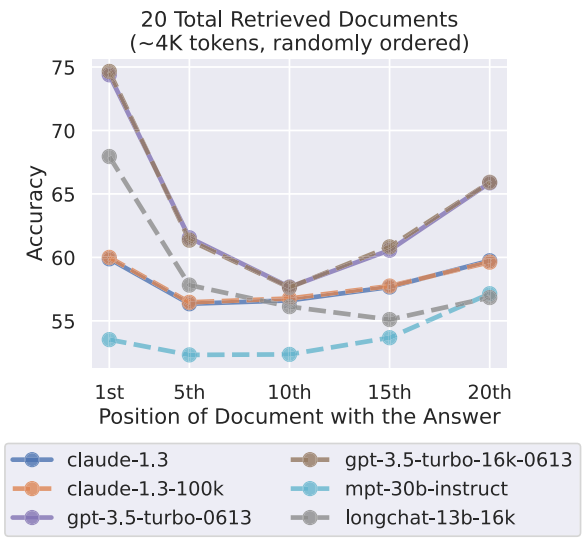

Figure 14: Language model performance when randomizing the order of the distractors (rather than presenting them in order of decreasing relevance) and mentioning as such in the prompt.

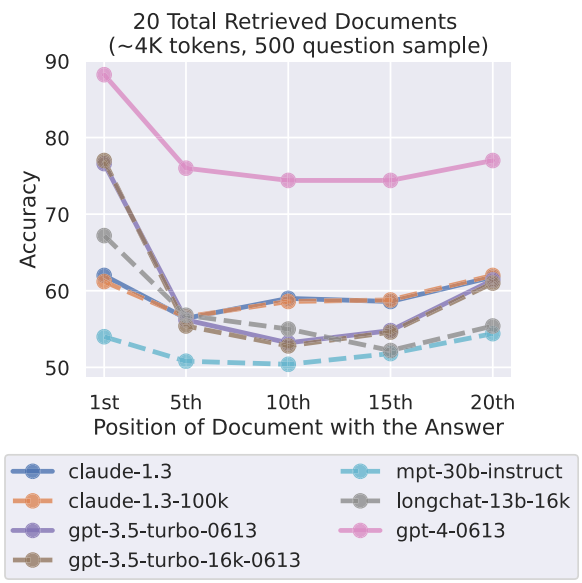

Figure 15: Although GPT-4 has higher absolute performance than other models, its performance still degrades when relevant information occurs in the middle of the input context.

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.

D GPT-4 Performance

We evaluate GPT-4 (8K) on a subset of 500 random multi-document QA examples with 20 total documents in each input context (Figure 15). 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.

E Llama-2 Performance

We evaluate Llama-2 (Touvron et al., 2023b) on multi-document QA with 20 total documents in each input context. The Llama tokenizer produces longer sequences than the tokenizers for our previously-studied models, so we discard 20 examples (out of 2655) that exceed Llama-2's maximum context length of 4096 tokens. We experiment with models of varying sizes (7B, 13B, and 70B parameters), with and without additional supervised fine-tuning and reinforcement learning from human feedback (''`-chat-`'' models). The results are presented in Figure 16.

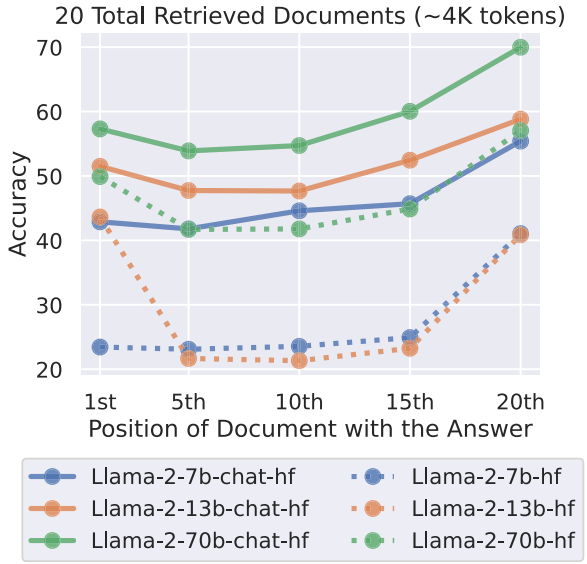

Figure 16: Multi-document QA performance (20 total documents) of Llama-2 models of varying sizes (7B, 13B, 70B parameters), with and without additional supervised fine-tuning and reinforcement learning from human feedback (''`-chat-`'' models).

Comparing Llama-2 models of varying sizes, we find that only the larger models (13B and 70B) exhibit the U-shaped performance curve (i.e., both primacy and recency bias)—the smallest

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026

Llama-2 models (7B) are solely recency-biased. Given these results, we hypothesize that prior work (e.g., Khandelwal et al., 2018; Sun et al., 2021) did not previously observe any primacy bias in language models because the models they studied were too small (less than 1B parameters).

Comparing between Llama-2 models with and without additional supervised fine-tuning and reinforcement learning from human feedback, we see that additional fine-tuning dramatically improves performance on the multi-document QA task. The 7B models with and without additional fine-tuning show minimal primacy bias, and are largely recency-biased. The 13B base model has a dramatic primacy and recency bias—there is a 20-point accuracy disparity between the best- and worst-case performance. Applying additional fine-tuning to the 13B seems to slightly reduce this bias (10-point worst-case degradation), but the bias remains significant. However, the 70B models with and without additional fine-tuning have largely similar trends (showing both primacy and recency bias), and additional fine-tuning minimally changes the positional bias severity.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf by guest on 08 May 2026