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 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.

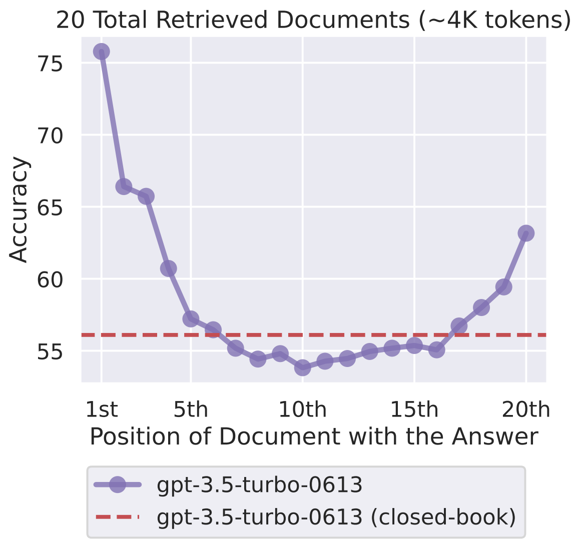

Figure 1: Changing the location of relevant information (in this case, the position of the passage that answers an input question) within the language model's input context results in a U-shaped performance curve—models are better at using relevant information that occurs at the very beginning (primacy 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.

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 (e.g., legal or scientific documents, conversation histories, etc.) or when language models are augmented 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. 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 512-2048 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.,

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

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.

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 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 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.

[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.

[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.

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. Reordering 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.

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 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