A Human-Inspired Reading Agent with Gist Memory of Very Long Contexts

Kuang-Huei Lee[1], **Xinyun Chen**[1], **Hiroki Furuta**[1], **John Canny**[1] **and Ian Fischer**[2]
[1]Google DeepMind, [2]Google Research
Correspond to: {leekh, iansf}@google.com; Author contributions are stated in Appendix J.
Website: read-agent.github.io

Current Large Language Models (LLMs) are not only limited to some maximum context length, but also are not able to robustly consume long inputs. To address these limitations, we propose ReadAgent, an LLM agent system that increases effective context length up to 20× in our experiments. Inspired by how humans interactively read long documents, we implement ReadAgent as a simple prompting system that uses the advanced language capabilities of LLMs to (1) decide what content to store together in a memory episode, (2) compress those memory episodes into short episodic memories called *gist memories*, and (3) take actions to look up passages in the original text if ReadAgent needs to remind itself of relevant details to complete a task. We evaluate ReadAgent against baselines using retrieval methods, using the original long contexts, and using the gist memories. These evaluations are performed on three long-document reading comprehension tasks: QuALITY, NarrativeQA, and QMSum. ReadAgent outperforms the baselines on all three tasks while extending the effective context window by 3 – 20×.

1. Introduction

Transformer-based Large Language Models (LLMs) are highly capable of language understanding, but the amount of text that LLMs are able to read at one time is constrained. Not only is there an explicit context length limitation, but it has also been found that performance of LLMs tends to decline with increasingly long inputs even when they don't actually exceed the explicit context window [25, 37]. In contrast, humans can read, understand, and reason over very long texts, such as a series of interrelated books.

We posit that an underlying reason for this gap is inherent in the differences in reading approaches. Typically, we use LLMs to consume the exact given content word-by-word and the process is relatively passive. On the other hand, humans read and reason over long text differently. First, the exact information tends to be forgotten quickly, whereas the fuzzier gist information, i.e. the substance irrespective of exact words, from past readings lasts much longer [34, 31, 33][1]. Second, human reading is an interactive process. When we need to remind ourselves of relevant details in order to complete a task, such as answering a question, we look them up in the original text.

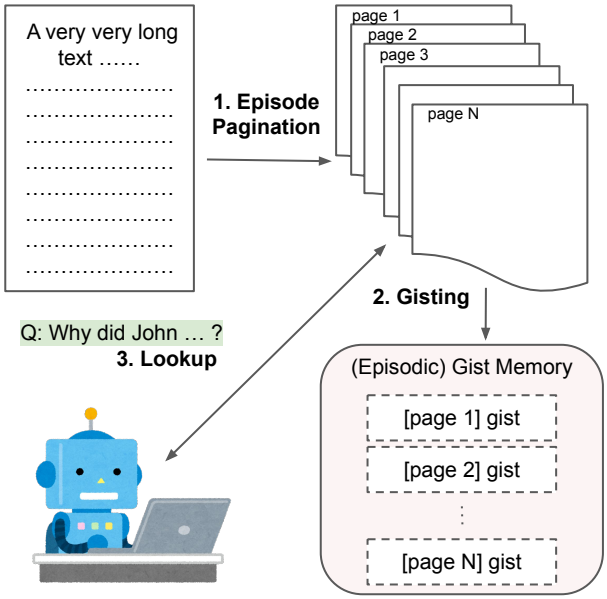


Figure 1 | ReadAgent workflow.

We think that using the fuzzy gist memory to capture global context and attending to local details together enables humans to reason over very long context efficiently, in terms of how much information to process at once, and is also important for comprehension. For example, if we were to infer the intention of a fictional character's specific action described on a page in a novel, besides focusing on the surrounding pages, we likely also need to understand the overall story and

[1]Fuzzy-trace theory [34] posits that people form two types of memory representations about a past event – verbatim and gist memories. Gist memories, often episodic, are fuzzy memories of past events, whereas verbatim memories contain details of past events. People prefer to reason with gists rather than with verbatim memories [32].

the character's personality from reading the whole book (see Appendix C for more analysis).

Motivated by these observations, we propose ReadAgent, an LLM agent system that handles long content inspired by the human approach. ReadAgent is simple to implement and can be built entirely by prompting a previously-trained LLM. As illustrated in Figure 1, it takes three primary steps: **(1) episode pagination**, where we prompt the LLM to decide where to pause in reading contiguous text; the content between pause points becomes an episode, which we refer to as *pages* in this work; **(2) memory gisting**, where we prompt the LLM to compress each page into a shorter *gist* and associate the gist with a corresponding context (e.g. which page the gist was from) – this gives the episodic *gist memory*; **(3) interactive look-up**, where the LLM looks at the given task and the complete set of gists in-context, makes decision on what page(s) to look up, combines the gists with these raw pages, and solves the task.

We evaluate ReadAgent by comparing against using only the gist memory without interactive look-up, using full text for datasets that can fit in the context window, and using retrieval methods to look up pages. ReadAgent outperforms all baselines across three challenging long-document comprehension tasks – QuALITY, NarrativeQA and QMSum – while increasing the effective context length significantly compared to the original LLM with reasonable computation overhead. On NarrativeQA Gutenberg test set, whose average length is 71k words and whose maximum is 343k words, ReadAgent improves the LLM rating (Section 4.1) by 12.97% and ROUGE-L by 31.98% over the best retrieval baseline and increases the effective context length by $\sim 20\times$. On QuALITY, where the articles can fit in an 8K context window, ReadAgent outperforms using full text with a $3\times$ effective context length.

Finally, in Appendix D, we adapt ReadAgent to web navigation, which is a fundamentally very-long context agent setting. We find that ReadAgent is simple to adapt to this setting and shows promising performance.

Our primary contributions are:

- **ReadAgent**, our human-inspired LLM agent that generates gist memories and looks up information as needed for solving tasks on long contexts (Section 3).
- Demonstration of significant performance advantages and scalability through a comprehensive experimental evaluation on challenging long-context benchmarks, comparisons against popular baselines, and analysis (Section 4).

2. Related Work

Long-Context LLMs The most direct way to improve LLM long-context performance is to train or fine-tune LLMs with longer context windows [3, 48, 14, 1, 40, 7]. Another approach is to explore new architectures or efficient implementations of the Transformer [41] attention layers to reduce the need of long-context fine-tuning [6, 30, 45, 19, 16]. However, LLM performance tends to decline with increasingly long inputs even when they don't exceed the specified context length [25]. LLM performance is also shown to be sensitive to distracting information in the context [37]. Thus, the effective context length could be shorter than the explicit limit. Our approach is complimentary to these approaches, scaling the effective context length of the underlying model while reducing the amount of distracting information in context, and requiring neither architectural changes nor training.

Retrieval Retrieval Augmented Generation (RAG) techniques [4, 11, 23, 18, 44, 28, 51] allow an LLM to query task-relevant information from a large database of documents or document pieces. Our work implements a form of retrieval by reasoning over a contextualized gist memory, all with zero-shot LLM prompting. This rethinking of retrieval directly leverages the strength and flexibility of LLM language understanding to reason about which documents to retrieve. Our approach is well-suited to densely-correlated long-document pieces, such as a series of books or a conversation history, but the database cannot scale arbitrarily, since the size of the gist memory is limited by the LLM's context length, and the gist memory's length correlates with the size of the database. In contrast, conventional retrieval approaches can handle larger databsases than our approach. In this work, we compare against retrieval systems that use exactly the same set of documents as our approach.

LLM Agents for Long Texts LLMs can be used as agents to interactively handle very long texts. WebGPT [26] and WebShop [47] learn browsing actions to search for the requested answer on the internet, despite not being designed to understand long documents. The PEARL [39] system proposes action plans for better long-document comprehension through iterative prompting. Self-note [22] amortizes reasoning steps and interleaves intermediate notes with the original documents to improve reasoning. Yang et al. [46] generates long outputs through iterative reasoning. However, these methods cannot address long input texts that exceed the LLM's context length. Similar to this work, MemWalker [5] also reads long documents interactively through iterative prompting. It traverses

a tree of different levels of summaries to search for task-related information. However, the hierarchical summary structure makes it difficult to reason over related but distant information at the same granularity (see Appendix F for more discussion).

3. ReadAgent

Figure 1 shows an overview of ReadAgent, which we describe in detail below.

3.1. Gist Memory

A *gist memory* is an ordered collection of short gists of chunks of text from the original long context. Building a gist memory has two steps: *pagination* and *memory gisting*, described in turn below.

Episode Pagination When ReadAgent reads through a long text, it makes decisions on what content to store together in a memory episode by choosing where to pause reading. At each step, we provide the LLM some text that begins from the previous pause point and ends when it reaches a `max_words` limit. We prompt the LLM to choose which point between paragraphs would be a natural point to pause, and then treat the content between the previous and current pause points as an episode, which we also refer as a *page*. This is *episode pagination*, which we implement with the following prompt.

Example Pagination Prompt

You are given a passage that is taken from a larger text (article, book, ...) and some numbered labels between the paragraphs in the passage.

Numbered labels are in angle brackets. For example, if the label number is 19, it shows as ⟨19⟩ in text.

Please choose a label where it is natural to break reading.

The label can be a scene transition, the end of a dialogue, the end of an argument, a narrative transition, etc.

Please answer with the break point label and explain.

For example, if ⟨57⟩ is a good point to break, answer with "Break point: ⟨57⟩\n Because ..."

Passage:
{...}
{PARAGRAPH 5 TEXT}
⟨5⟩
{PARAGRAPH 6 TEXT}
⟨6⟩
{PARAGRAPH 7 TEXT}
{...}

As shown in the prompt, possible pause points are inserted between paragraphs as numbered tags (e.g. ⟨13⟩), making this a multiple choice question for the LLM. We only start inserting these numbered tags after a `min_words` threshold to make sure that each page has at least `min_words`.

Memory Gisting For each *page*, we prompt the LLM to shorten the exact content into a *gist*, or summary, as follows.

Example Gisting Prompt

Please shorten the following passage.

Just give me a shortened version. DO NOT explain your reason.

Passage:
{PAGE TEXT}

We subsequently prepend a page tag to each gist (e.g. "⟨Page 2⟩\n{GIST CONTENT}") to contextualize it (indicate where the gist was from), and then concatenate all gists. This gives us the gist memory. We use the word "shorten" in the prompt to generate these summarizing gists as it tends to help preserve the narrative flow, making it more natural to concatenate. Using the word "summarize" tended to produce a restructured summary in our experiments.

The original page size is a key factor for how compressed the gist is. Let's say the smallest unit of text that we consider is a paragraph. Intuitively, a paragraph likely has some amount of mutual information with its neighbors. Thus, the larger chunk of text we group together, the more duplicated information we can remove. Empirically, compressing larger chunks of text with LLMs also tends to remove more details, which could affect performance. We control the page size by changing `min_words` and `max_words` in pagination. This trade-off is studied in Appendix A.

3.2. Parallel and Sequential Interactive Look-Up

For a given task about a long document, we want ReadAgent to take actions to look up relevant details in the original text in addition to using its gist memory. As the gist memories are contextualized with page numbers, we simply prompt the LLM to answer which page(s) it would like to look up and read again given the specific task. In the following we discuss two look-up strategies: looking up all pages at once in parallel (**ReadAgent-P**) and sequentially looking up one page at a time (**ReadAgent-S**).

ReadAgent-P As in the following example prompt for question-answering, typically we give it a maximum number of pages that it can look up but also instruct it to use as few pages as possible to avoid unnecessary computational overhead and distracting information. The following prompt shows parallel look-up, where the model requests multiple pages in response to a single prompt.

Example Parallel Lookup Prompt (ReadAgent-P)

The following text is what you remember from reading an article and a multiple choice question related to it.

You may read 1 to 5 page(s) of the article again to refresh your memory to prepare yourself for the question.

Please respond with which page(s) you would like to read.

For example, if you only need to read Page 8, respond with "I want to look up Page [8] to ..."; if you would like to read Page 7 and 12, respond with "I want to look up Page [7, 12] to ..."; if you would like to read Page 2, 3, 7, 15 and 18, respond with "I want to look up Page [2, 3, 7, 15, 18] to ...".

DO NOT select more pages if you don't need to.

You don't need to answer the question yet.

Text:
{GIST MEMORY}

Question:
{QUESTION}

The selected raw pages replace the gist(s) at the corresponding positions in memory, preserving the overall narrative flow. Then we prompt the LLM again with the task and the updated memory and ask it to solve the task.

ReadAgent-S We also study the sequential look-up strategy, where the model requests one page at a time, up to some maximum number of pages. In sequential look-up, the model gets to see the previously expanded pages before deciding which page to expand. This gives the model access to more information than parallel look-up, so we might expect it to perform better in some situations. However, the larger number of interactions with the model increases the computational cost, so sequential look-up should only be used on tasks where it provides clear benefits.

Example Sequential Lookup Prompt (ReadAgent-S)

The following text is what you remember from reading a meeting transcript, followed by a question about the transcript. You may read multiple pages of the transcript again to refresh your memory and prepare to answer the question. Each page that you re-read can significantly improve your chance of answering the question correctly. Please specify a SINGLE page you would like to read again or say "STOP". To read a page again, respond with "Page $PAGE_NUM", replacing $PAGE_NUM with the target page number. You can only specify a SINGLE page in your response at this time. To stop, simply say "STOP". DO NOT answer the question in your response.

Text:

{GISTS WITH IN-LINE EXPANDED PAGES}

Pages re-read already (DO NOT ask to read them again):

{LIST OF PAGE NUMBERS ALREADY READ}

Question:

{QUESTION}

Specify a SINGLE page to read again, or say STOP:

3.3. Computational Overhead and Scalability

Episode pagination, memory gisting and interactive look-ups require iterative inference, which is a possible computational overhead. However, as we show in the following, the overhead is bounded linearly by a small factor, making our approach scale well with input length.

Pagination: In theory, an LLM could read a document and directly provide the pagination in a single pass, so the minimum number of words the LLM must process is the length of the document. Our pagination algorithm splits the document into chunks of at most `max_words`, and then guarantees that at least `min_words` are consumed at each step. Thus, the ratio $\frac{\texttt{max\_words}}{\texttt{min\_words}}$ gives an upper bound on how many times the word length of the document the LLM must process using our algorithm. **Gisting:** Memory gisting is one additional pass of the raw input words, since each page is gisted independently. **Retrieval:** Parallel look-ups are conditioned on gists instead of the full text, and thus will be much shorter than one pass of the raw input words. Each step of a sequential look-up is similar to parallel look-ups and the overall cost is capped with the maximum number of look-ups allowed. **Response:** Finally, answering is also similar to parallel look-ups. There is additional overhead from the prompt templates, of course.

For example, in our QMSum ReadAgent-P 6 page experiments, $\frac{\texttt{max\_words}}{\texttt{min\_words}} \approx 2$, the gist memory is less than 0.2× the original context and the retrieved pages increase that to 0.3×, so the LLM processes ~ 3.5× the original words.

3.4. ReadAgent Variants

In Appendix E, we discuss variants of ReadAgent that can be useful in different problem settings, including when the target task is known prior to reading the long document. In Appendix D, we describe adapting ReadAgent to work in the web navigation setting.

4. Experiments

We evaluate ReadAgent's long-document reading comprehension ability on three long-context question-answering challenges: QuALITY [27], NarrativeQA [21] and QMSum [50]. Although ReadAgent does not require any model training, we develop the proposed method on the training sets and test on the validation, test and/or development sets to avoid any risk of overfitting system hyperparameters.

In this work, we primarily use the instruction-tuned PaLM 2-L [2] for our experiments and evaluation. The

context length of PaLM 2-L is 8K tokens. Details of the model can be found in Anil et al. [2]. Additionally, we provide GPT-3.5[2] results in Appendix B, and experimental results on the web navigation setting in Appendix D.

One important performance measure of the techniques considered here is the **compression rate** (**CR**). We define this as CR $\equiv 100 * (1 - \frac{\text{word-count(in-context text)}}{\text{word-count(full-context text)}})$ at the final query.

4.1. LLM Raters

NarrativeQA and QMSum both have one or more free-form reference responses. They are typically evaluated using syntactic matching metrics such as ROUGE [24] F-Measure. We additionally evaluate these datasets using an automatic LLM Rater as an alternative to human evaluation similar to Peng et al. [29], Chiang et al. [9], Zheng et al. [49], Chiang and Lee [8].

In our implementation, we prompt the LLM to look at the question or instruction and compare the model's answer to the reference answer. The "Strict LLM Rater Prompt" shown below is for judging whether there is an exact match, and the "Permissive LLM Rater Prompt" is for judging whether there is an exact match or a partial match. We apply both prompts to all model responses. If either rater decides there is an exact match, we count it as an exact match. If the strict rater is negative but the permissive rater detects a partial match, we count it as a partial match. Otherwise, it's not a match. In the case that there are multiple reference answers, the response is compared against each reference answer in turn, and the highest rating is returned.

Based on these raters, we define two different scores: **LLM-Rating-1** (**LR-1**) is a strict evaluation score, where we count the percentage of exact matches over all examples; **LLM-Rating-2** (**LR-2**) is permissive, where we count the percentage of exact and partial matches.

Strict LLM Rater Prompt

After reading some text, John was given the following question about the text:

{QUESTION TEXT}

John's answer to the question was:

{MODEL RESONSE TEXT}

The ground truth answer was:

{REFERENCE RESPONSE TEXT}

Does John's answer agree with the ground truth answer? Please answer YES or NO.

Permissive LLM Rater Prompt

After reading some text, John was given the following question about the text:

{QUESTION TEXT}

John's answer to the question was:

{MODEL RESONSE TEXT}

The ground truth answer was:

{REFERENCE RESPONSE TEXT}

Does John's answer agree with the ground truth answer? Please answer "Yes", "Yes, partially", or "No". If John's response has any overlap with the ground truth answer, answer "Yes, partially". If John's response contains the ground truth answer, answer "Yes". If John's response is more specific than the ground truth answer, answer "Yes".

4.2. Baseline Methods

Retrieval-Augmented Generation (RAG) As discussed in Section 2, RAG [23] is a popular approach to extend access to a large amount of text beyond what can fit in the LLM context window. In this paper we compare ReadAgent to RAG baselines using conventional retrieval methods to find relevant "pages" in a long text, where we reuse the pages generated by ReadAgent. We consider two relevance methods: Okapi BM25 [35] and neural retrieval based on the Gemini API embedding model (models/embedding-001)[3]. The neural retrieval relevance score is defined as the dot product between the question embedding vector and each page (or gist memory embedding vector in the case of NarrativeQA, see Section 4.3.2). For reading comprehension tasks, the pages are ranked by relevance to each question, and we prompt the LLM to look at the top-k pages as context for answering the question. In most retrieval settings, the database of documents is quite large, which makes the retrieval task more challenging. In our setting, ReadAgent and retrieval methods all use a per-document database, rather than per-dataset. For example, in QuALITY, there are hundreds of articles, each with multiple questions. The database for retrieval in each question is only the extracted pages from the corresponding article (typically less than 20 pages), rather than the thousands of pages from the entire dataset.

Full or Truncated Text Content The maximum length of QuALITY dev articles is ~6,000 words, which can fit into the PaLM 2-L context window. This allows us to evaluate ReadAgent against directly using the full long document for long-context reading comprehension. The maximum length of QMSum is over 26,000 words. Consequently, we choose to truncate the text to close to the context window limit (6,000 words for PaLM 2-L experiments) to ensure that the truncated text fits in the LLM's context, though this would

[2] http://openai.com/api/

[3] https://ai.google.dev/models/gemini

generally be a weaker baseline. Finally, since the average length of NarrativeQA documents significantly exceeds the context window, it is less meaningful to perform the truncated-context comparison.

Gist Memory We can also attempt to solve the given task by reasoning directly over the gist memory. Doing so helps us understand not only the importance of interactive look-up but also how using the LLM-compressed information alone compares to the full content and retrieval baselines.

4.3. Long-Context Reading Comprehension

4.3.1. QuALITY

QuALITY [27] is a four-way multiple choice question answering challenge with text data from several different sources. QuALITY is evaluated using accuracy, with 25% corresponding to chance performance.

The dev set has an average length of 4,122 words and a maximum of 5,967. The gist memory has an average length of 650 words and a maximum of 1,264. Figure 2 shows the word statistics for the original text and the gists. The compression rate of the gists is 84.24%. See Appendix G for QuALITY pagination hyperparameters.

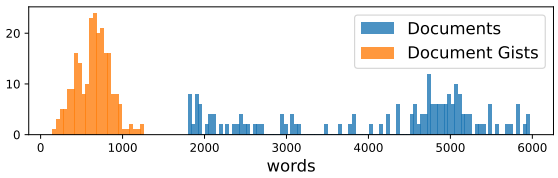


Figure 2 | Histogram of QuALITY word counts for the original text and the gists.

Table 1 shows the experimental results on QuALITY, where ReadAgent (Look up 1-5 pages) gives the best results with a compression rate of 66.97% (meaning that ~ 3× as many tokens can fit in the context window after gisting). The performance increases as we increase the maximum number of pages allowed for look-up, up to 5 pages. At 6 pages, we see that the performance starts to degrade slightly, indicating that allowing 6 pages of context may be increasing the rate of distracting information.

Notably, ReadAgent outperforms using the full original text, which could have been an upper bound on the performance – every other method reduces the amount of text the LLM considers before generating its response. However, this is not a surprising result. Prior work shows that current LLMs are not able to effectively use the full long context window [25], potentially due to training data sparsity, and distracting information can also reduce performance [37, 42].

Method	CR (# LU)	Accuracy
BM25 Retrieval		
Top-1	89.96% (1)	70.55% ± 0.07
Top-2	80.25% (2)	78.38% ± 0.10
Top-3	70.98% (3)	81.59% ± 0.17
Top-4	61.90% (4)	84.28% ± 0.15
Neural Retrieval with Gemini API		
Top-1	90.72% (1)	70.98% ± 0.06
Top-2	81.88% (2)	79.56% ± 0.10
Top-3	73.13% (3)	83.11% ± 0.11
Top-4	64.50% (4)	84.98% ± 0.06
Full Raw Content	0%	85.83% ± 0.19
GistMem	84.24%	77.95% ± 0.08
ReadAgent-P		
Look up 1 pg	76.63% (1.0)	83.80% ± 0.17
Look up 1-2 pgs	72.17% (1.6)	84.95% ± 0.07
Look up 1-3 pgs	69.23% (2.0)	85.46% ± 0.07
Look up 1-4 pgs	67.72% (2.2)	86.31% ± 0.18
Look up 1-5 pgs	66.26% (2.4)	**86.63%** ± 0.10
Look up 1-6 pgs	64.63% (2.7)	86.40% ± 0.07
ReadAgent-S 1-6 pgs	60.27% (3.3)	**86.88%** ± 0.06

Table 1 | QuALITY results on the dev set of 230 docs and 2086 questions using PaLM 2-L. **CR** is the compression rate. **# LU** is the number of lookups. We report means and standard deviations across 3 runs. We omit standard deviations for CR and # LU for presentation purposes; they were all inconsequential.

4.3.2. NarrativeQA

NarrativeQA [21] has the longest context length on average among the three reading comprehension datasets we choose. The dataset is divided into books (Gutenberg) and move scripts. The Gutenberg test set have 70,619 words on average, and the maximum is 343,910 words; the movie scripts test set have 29,963 on average, and the maximum is 63,957 words. As the reference answers are free-form, we evaluate based on ROUGE [24] and the LLM Ratings (Section 4.1). The original main texts are replaced with the HTML-stripped version from SCROLLS [36].

Because of the length of NarrativeQA articles, in order to fit the gists into the context window, we significantly expand the page size, resulting in stronger compression (Section 3.1). For example, the Gutenburg gists from the test set have 2,217 words on average and the maximum is 6,471 words, whereas the movie script gists have 2,155 words on average and the maximum is 4,511 words. Figures 4 and 5 (appendix) show the word statistics for the original text and the gists in Gutenberg and movie scripts respectively. The compression rate of the gists is 96.80% for Gutenberg texts and 91.98% for movie scripts. See Appendices G and H for NarrativeQA pagination hyperparameters and more details.

	Gutenberg Validation (58 docs & 1743 questions)						Gutenberg Test (177 docs & 5207 questions)					
Method	CR (# LU)	LR-1	LR-2	R-1	R-2	R-L	CR (# LU)	LR-1	LR-2	R-1	R-2	R-L
BM25 Retrieval												
Top-1	97.63% (1)	39.01%	50.14%	0.166	0.061	0.156	97.42% (1)	43.5%	55.33%	0.176	0.065	0.165
Top-2	95.24% (2)	49.34%	60.76%	0.203	0.079	0.191	94.80% (2)	51.70%	64.53%	0.206	0.082	0.194
Top-3	93.34% (3)	52.73%	63.68%	0.208	0.080	0.195	93.02% (3)	52.97%	66.03%	0.210	0.083	0.197
Top-4	92.47% (4)	53.59%	64.26%	0.211	0.082	0.197	92.27% (4)	53.60%	66.16%	0.210	0.084	0.197
Neural Retrieval with Gemini API												
Top-1	98.19% (1)	34.25%	46.53%	0.146	0.051	0.134	98.14% (1)	36.47%	47.8%	0.150	0.054	0.140
Top-2	96.30% (2)	44.69%	54.96%	0.180	0.069	0.167	96.15% (2)	44.48%	56.17%	0.182	0.070	0.170
Top-3	94.62% (3)	46.24%	57.31%	0.191	0.077	0.178	94.42% (3)	48.97%	60.73%	0.195	0.076	0.183
Top-4	93.45% (4)	48.59%	59.21%	0.196	0.079	0.184	93.25% (4)	50.62%	62.05%	0.203	0.080	0.191
GistMem	96.89%	55.31%	68.22%	0.233	0.091	0.218	96.80%	55.79%	71.19%	0.231	0.092	0.217
ReadAgent-P												
Look up 1 pg	95.15% (0.94)	58.92%	71.89%	**0.244**	**0.101**	**0.230**	94.84% (0.93)	59.98%	**73.23%**	**0.240**	**0.098**	**0.226**
Look up 1-2 pgs	94.79% (1.23)	**59.84%**	**72.29%**	0.239	0.098	0.224	94.36% (1.34)	59.19%	72.65%	0.231	0.091	0.218
Look up 1-3 pgs	94.39% (1.50)	**59.84%**	71.89%	0.240	0.098	0.226	94.03% (1.61)	59.63%	72.84%	0.230	0.093	0.217
ReadAgent-S 1-2 pgs	94.35% (1.38)	57.89%	71.14%	0.239	0.097	0.225	93.86% (1.46)	60.48%	72.48%	0.232	0.095	0.219
ReadAgent-S 1-3 pgs	94.08% (1.57)	58.52%	71.49%	0.242	0.098	0.229	93.67% (1.57)	**60.55%**	72.79%	0.231	0.095	0.219
	Movie Validation (57 docs & 1699 questions)						Movie Test (172 docs & 5139 questions)					
BM25 Retrieval												
Top-1	97.07% (1)	32.67%	42.61%	0.156	0.058	0.144	96.61% (1)	33.64%	43.34%	0.154	0.054	0.143
Top-2	94.12% (2)	39.97%	50.21%	0.187	0.070	0.174	93.81% (2)	42.50%	53.05%	0.191	0.072	0.178
Top-3	91.18% (3)	43.61%	53.91%	0.198	0.077	0.185	91.00% (3)	46.97%	57.52%	0.207	0.080	0.193
Top-4	88.24% (4)	46.85%	57.62%	0.210	0.084	0.198	88.19% (4)	50.18%	60.13%	0.217	0.085	0.202
Neural Retrieval with Gemini API												
Top-1	97.07% (1)	32.02%	41.44%	0.153	0.053	0.142	96.67% (1)	37.24%	46.22%	0.130	0.043	0.118
Top-2	94.19% (2)	43.20%	51.38%	0.160	0.057	0.148	93.90% (2)	46.49%	54.60%	0.164	0.061	0.151
Top-3	91.29% (3)	47.56%	56.21%	0.176	0.064	0.163	91.14% (3)	50.69%	58.92%	0.186	0.071	0.172
Top-4	88.38% (4)	49.09%	59.33%	0.193	0.075	0.180	88.36% (4)	52.13%	59.41%	0.184	0.072	0.171
GistMem	92.09%	52.56%	64.39%	0.242	0.103	0.227	91.98%	54.68%	64.00%	0.248	0.105	0.234
ReadAgent-P												
Look up 1 pg	89.20% (0.99)	53.38%	65.57%	**0.247**	**0.106**	**0.233**	89.22% (0.98)	57.68%	68.01%	**0.274**	**0.116**	**0.260**
Look up 1-2 pgs	87.68% (1.52)	54.62%	65.63%	0.238	0.098	0.223	88.10% (1.39)	58.24%	68.81%	0.270	0.115	0.255
Look up 1-3 pgs	86.57% (1.91)	54.91%	65.86%	0.241	0.099	0.225	86.73% (1.89)	58.82%	69.12%	0.272	**0.116**	0.257
ReadAgent-S 1-2 pgs	86.36% (1.98)	59.33%	68.28%	0.203	0.082	0.188	85.92% (1.98)	63.33%	72.06%	0.214	0.086	0.199
ReadAgent-S 1-3 pgs	83.56% (2.95)	**59.45%**	**68.81%**	0.210	0.087	0.195	83.18% (2.95)	**64.53%**	**73.06%**	0.217	0.090	0.202

Table 2 | NarrativeQA results (PaLM 2-L). **CR** is the compression rate. **# LU** is the number of lookups. **R-1**, **R-2**, and **R-L** are ROUGE F-Measures. **LR-1**, and **LR-2** are LLM-Ratings.

For the neural retrieval models, we use the gist memory embedding vectors rather than the page embedding vectors because the Gemini API embedding model is limited to 10,000 characters (or less than 2,000 tokens, in expectation), which is too short for embedding full pages in our NarrativeQA experiments. However, using those embedding vectors, we then return the original pages to the LLM context as normal, and use those pages as described in Section 4.2.

Because the Gutenberg texts and the movie scripts have significantly different distributions, we present the results separately. The results in Table 2. ReadAgent again outperforms all the baselines across all subsets of NarrativeQA.

4.3.3. QMSum

QMSum [50] consists of meeting transcripts on various topics and associated questions or instructions. We use the concatenated version of QMSum provided by SCROLLS [36]. The transcripts tend to be quite long, ranging in length from 1,000 to 26,300 words, with an average length of about 10,000 words. Figure 7 shows the histograms of word counts for the QMSum training set. The answers are free form text, so the standard evaluation metric is ROUGE F-Measure. We additionally evaluate using our LLM Ratings (Section 4.1). See Appendices G and I for hyperparameters and additional results.

In Tables 3 and 11, we see that performance improves as the compression rate decreases, so techniques that look up more pages tend to do better than techniques that look up fewer pages. We also see that ReadAgent-S substantially outperforms ReadAgent-P (and all baselines). This performance improvement comes at a cost of up to six times as many requests in the retrieval phase. Since other datasets don't have such a strong performance improvement, we suspect that QMSum is in some sense a more challenging dataset, requiring the model to actively search through the gisted transcript to locate relevant information. This hypothesis seems reasonable, as meeting transcripts are much less structured than the documents, books, and movies found in QuALITY and NarrativeQA.

A large fraction of the tasks in QMSum are a request to provide a summary, rather than a concrete question about some content in the meeting. For many of these, the LLM refuses to look up any pages, instead responding with "I don't need to look up any pages. I can summarize the whole meeting based on what I already remember.", for example. Consequently, the average number of pages looked up for ReadAgent is much lower than the maximum allowed. However, on the tasks that actually involve a question, ReadAgent tends to use most or all of the available lookup pages.

In Tables 3 and 11, the ROUGE scores by themselves don't always show a clear trend. This is because as

Method	CR (# LU)	LLM Rating-1	LLM Rating-2	ROUGE-1	ROUGE-2	ROUGE-L	Resp. Length
BM25 Retrieval							
Top-1	95.69% (1.00)	32.48% ± 1.65	63.85% ± 1.51	27.53 ± 0.23	7.00 ± 0.14	18.45 ± 0.16	48.62 ± 0.28
Top-2	91.48% (2.00)	29.41% ± 0.60	71.57% ± 1.48	28.85 ± 0.17	7.59 ± 0.08	19.34 ± 0.14	52.39 ± 0.49
Top-3	86.93% (3.00)	34.80% ± 1.14	79.53% ± 0.35	30.69 ± 0.17	8.40 ± 0.11	20.64 ± 0.13	53.59 ± 0.35
Top-4	82.55% (4.00)	35.66% ± 0.30	81.13% ± 0.35	31.10 ± 0.10	8.53 ± 0.06	20.36 ± 0.11	54.96 ± 0.42
Top-5	78.13% (5.00)	39.09% ± 0.92	84.44% ± 0.46	31.16 ± 0.14	8.52 ± 0.08	20.69 ± 0.03	54.52 ± 0.13
Top-6	73.97% (6.00)	37.87% ± 0.90	83.70% ± 0.87	31.06 ± 0.04	8.38 ± 0.06	20.43 ± 0.08	56.18 ± 0.44
Neural Retrieval with Gemini API							
Top-1	95.99% (1.00)	34.80% ± 1.39	68.87% ± 0.62	27.86 ± 0.12	7.12 ± 0.04	18.76 ± 0.09	49.46 ± 0.23
Top-2	92.02% (2.00)	40.32% ± 0.92	81.50% ± 0.46	30.17 ± 0.08	8.03 ± 0.03	19.80 ± 0.08	55.48 ± 0.27
Top-3	87.93% (3.00)	40.93% ± 1.35	85.17% ± 1.25	31.36 ± 0.12	8.67 ± 0.10	20.68 ± 0.10	56.71 ± 0.27
Top-4	83.71% (4.00)	40.56% ± 0.62	84.31% ± 0.87	31.52 ± 0.11	8.59 ± 0.10	20.40 ± 0.10	56.47 ± 0.71
Top-5	79.47% (5.00)	40.20% ± 0.76	86.76% ± 0.60	31.32 ± 0.11	8.49 ± 0.11	20.49 ± 0.07	56.73 ± 0.91
Top-6	75.44% (6.00)	40.81% ± 0.52	87.01% ± 0.35	31.92 ± 0.02	8.73 ± 0.09	20.82 ± 0.05	58.39 ± 0.31
Truncated Raw Content							
First 6k words	32.59% (0.00)	14.71% ± 0.79	52.45% ± 0.69	25.42 ± 0.05	4.98 ± 0.09	16.58 ± 0.10	58.42 ± 0.11
Last 6k words	32.38% (0.00)	10.42% ± 0.62	35.66% ± 2.46	20.69 ± 0.19	3.44 ± 0.10	14.13 ± 0.08	44.23 ± 0.11
GistMem	83.13% (0.00)	40.20% ± 0.96	89.83% ± 0.76	31.00 ± 0.09	7.99 ± 0.04	20.15 ± 0.08	65.75 ± 0.20
ReadAgent-P							
Look up 1 pg	80.00% (0.98)	40.56% ± 0.46	89.46% ± 1.48	31.26 ± 0.09	8.22 ± 0.15	20.29 ± 0.07	63.78 ± 1.13
Look up 1-2 pgs	77.38% (1.71)	39.71% ± 1.87	89.71% ± 0.60	31.11 ± 0.04	8.01 ± 0.15	20.21 ± 0.04	64.73 ± 1.02
Look up 1-3 pgs	75.07% (2.53)	38.36% ± 1.21	89.71% ± 0.60	31.50 ± 0.29	8.15 ± 0.15	20.45 ± 0.24	63.91 ± 1.58
Look up 1-4 pgs	73.48% (3.08)	39.95% ± 1.51	90.56% ± 0.35	31.34 ± 0.05	8.08 ± 0.18	20.26 ± 0.07	63.40 ± 0.79
Look up 1-5 pgs	72.29% (3.50)	37.99% ± 0.96	87.75% ± 0.46	31.16 ± 0.10	8.06 ± 0.05	20.35 ± 0.12	65.22 ± 1.40
Look up 1-6 pgs	70.90% (3.97)	39.09% ± 2.04	88.24% ± 0.60	31.50 ± 0.30	8.05 ± 0.13	20.26 ± 0.13	66.70 ± 0.62
ReadAgent-S 1-6 pgs	70.34% (3.55)	**46.57%** ± 0.87	**91.54%** ± 0.30	**32.90** ± 0.17	**8.87** ± 0.23	**21.15** ± 0.14	68.87 ± 0.60

Table 3 | **QMSum validation** results (PaLM 2-L) means and standard deviations across 3 runs. 35 articles and 272 questions. **CR** is the compression rate. **# LU** is the number of lookups. **Resp. Length** is the length in words of the model's final response.

the length of the texts increase (corresponding to the compression rates decreasing), the response lengths increase as well. Longer response lengths result in lower ROUGE precision values, which pushes down the F-Measures. Consequently, for the ROUGE scores to increase as text length increases, the improvement to recall must be more substantial than the reduction to precision. This happens to some extent, but the effect size is small. Furthermore, including gists in the text substantially increases the response length, as is the case for GistMem and all the ReadAgent approaches. This increase is in spite of the fact that all models use the same question-answering prompt, so there is no prompt difference to cause the increased response lengths. This makes it much more challenging for GistMem and ReadAgent to outperform the retrieval methods in ROUGE score. Nevertheless, ReadAgent-S manages to have the highest ROUGE scores as well as the highest LLM ratings. Because of these issues with ROUGE, we consider the LLM ratings to be more informative for comparisons between these runs. However, the LLM ratings do not make it easy to compare with results using a different LLM to rate, such as GPT, and they also do not allow for easy comparisons with other works. The same observation applies to the NarrativeQA results above.

4.4. Ablation Study and Analysis

We provide additional ablation studies in Appendix A.

Retrieval Quality In Table 4, we compare using GistMem with neural retrieval to look up one page with using ReadAgent to look up one page. This is equivalent to replacing ReadAgent's prompt-based retrieval with neural retrieval. ReadAgent's retrieval performs better here.

Method	Accuracy
GistMem + Neural Retrieval Top-1	82.74%
ReadAgent-P (Look up 1 pg)	83.80%

Table 4 | ReadAgent vs. GistMem with neural retrieval.

5. Conclusion

We have presented ReadAgent, a simple interactive prompting system to mitigate the context length and context use limitations of current LLMs. ReadAgent outperforms other strong zero-shot (i.e., not trained or finetuned on the training set) baselines across standard performance metrics of accuracy or ROUGE scores. These results demonstrate that LLMs are capable of generating compressed textual representations of long contexts that are useful for tasks that humans think are important, even without knowing those tasks ahead of time. I.e., the LLM can generate broadly useful gist memories even before knowing what questions are going to be asked about the text that is being gisted. The results also demonstrate that LLMs are capable of reasoning interactively over such compressed representations, using them to decide what information needs to be retrieved to most effectively perform a known task. This method can increase the effective context length by up to 20× while outperforming conventional retrieval techniques. However, this approach

does not give infinite context lengths, nor does it guarantee good performance when the gist memory itself is extremely long. Future work will need to address these fundamental limitations in LLMs.

Acknowledgements

The authors thank Sergey Ioffe, Rif A. Saurous, Yujin Tang, Sergio Guadarrama, Daliang Li, Felix Yu, and Rob Fergus for valuable feedback and discussion.

References

[1] J. Ainslie, T. Lei, M. de Jong, S. Ontañón, S. Brahma, Y. Zemlyanskiy, D. Uthus, M. Guo, J. Lee-Thorp, Y. Tay, Y.-H. Sung, and S. Sanghai. Colt5: Faster long-range transformers with conditional computation, 2023.

[2] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[3] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer, 2020.

[4] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions, 2017.

[5] H. Chen, R. Pasunuru, J. Weston, and A. Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

[6] S. Chen, S. Wong, L. Chen, and Y. Tian. Extending context window of large language models via positional interpolation, 2023.

[7] Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

[8] C.-H. Chiang and H.-y. Lee. Can large language models be an alternative to human evaluations? In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.870.

[9] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[10] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023.

[11] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. Wizard of wikipedia: Knowledge-powered conversational agents, 2019.

[12] H. Furuta, Y. Matsuo, A. Faust, and I. Gur. Language model agents suffer from compositional generalization in web automation, 2023.

[13] H. Furuta, O. Nachum, K.-H. Lee, Y. Matsuo, S. S. Gu, and I. Gur. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arxiv:2305.11854*, 2023.

[14] M. Guo, J. Ainslie, D. Uthus, S. Ontanon, J. Ni, Y.-H. Sung, and Y. Yang. LongT5: Efficient text-to-text transformer for long sequences. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.55.

[15] I. Gur, H. Furuta, A. Huang, M. Safdari, Y. Matsuo, D. Eck, and A. Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arxiv:2307.12856*, 2023.

[16] C. Han, Q. Wang, W. Xiong, Y. Chen, H. Ji, and S. Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

[17] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[18] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering, 2021.

[19] H. Jin, X. Han, J. Yang, Z. Jiang, Z. Liu, C.-Y. Chang, H. Chen, and X. Hu. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*, 2024.

[20] G. Kim, P. Baldi, and S. McAleer. Language models can solve computer tasks. *arXiv preprint arxiv:2303.17491*, 2023.

[21] T. Kočiskỳ, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

[22] J. Lanchantin, S. Toshniwal, J. Weston, A. Szlam, and S. Sukhbaatar. Learning to reason and memorize with self-notes. *arXiv preprint arXiv:2305.00833*, 2023.

[23] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[24] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop: Text Summarization Branches Out 2004*, page 10, 01 2004.

[25] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[26] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[27] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, et al. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, 2022.

[28] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[29] B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

[30] O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

[31] V. Reyna and C. Brainerd. Fuzzy-trace theory: Some foundational issues. *Learning and Individual differences*, 7(2):145–162, 1995.

[32] V. F. Reyna. A theory of medical decision making and health: fuzzy trace theory. *Medical decision making*, 28(6):850–865, 2008.

[33] V. F. Reyna. A new intuitionism: Meaning, memory, and development in fuzzy-trace theory. *Judgment and Decision making*, 7(3):332–359, 2012.

[34] V. F. Reyna and C. J. Brainerd. Fuzzy-trace theory: An interim synthesis. *Learning and individual Differences*, 7(1):1–75, 1995.

[35] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[36] U. Shaham, E. Segal, M. Ivgi, A. Efrat, O. Yoran, A. Haviv, A. Gupta, W. Xiong, M. Geva, J. Berant, et al. Scrolls: Standardized comparison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, 2022.

[37] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[38] T. Shi, A. Karpathy, L. Fan, J. Hernandez, and P. Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, 2017.

[39] S. Sun, Y. Liu, S. Wang, C. Zhu, and M. Iyyer. Pearl: Prompting large language models to plan and execute actions over long documents. *arXiv preprint arXiv:2305.14564*, 2023.

[40] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[42] J. Weston and S. Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023.

[43] J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike, and P. Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

[44] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

[45] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[46] K. Yang, Y. Tian, N. Peng, and D. Klein. Re3: Generating longer stories with recursive reprompting and revision. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4393–4479, 2022.

[47] S. Yao, H. Chen, J. Yang, and K. Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[48] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020.

[49] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[50] M. Zhong, D. Yin, T. Yu, A. Zaidi, M. Mutuma, R. Jha, A. Hassan, A. Celikyilmaz, Y. Liu, X. Qiu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, 2021.

[51] W. Zhong, L. Guo, Q. Gao, and Y. Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

A. Ablation Study and Analysis (Continued)

Episode pagination In this work we ask ReadAgent to decide where to pause reading and what information to store together in memory (Section 3.1), whereas in prior art, rule-based segmentation of text is typically used [5, 43]. We compare the two approaches with similar page length on average in Table 5 to demonstrate that it is indeed beneficial to break at pause points that LLMs consider natural (e.g. scene transitions, ends of dialogue, narrative transitions, etc).

	LLM	Uniform Length
ReadAgent-P (1-5 pgs) Acc.	**86.63%**	85.71%

Table 5 | ReadAgent accuracy on QuALITY with episode pagination based on LLM (PaLM 2-L) vs. uniform length pagination.

The compression trade-off Table 6 presents the empirical results of compression rate increasing as page size increases. As the compression rate decreases, the gists are more useful for answering questions directly. However, for ReadAgent with look-ups, when the compression rate gets too low or too high, accuracy suffers.

	GistMem		ReadAgent-P (1-5 pgs)	
`max_words`	CR	Acc	CR	Acc
400	79.58%	79.91%	66.44%	86.34%
600	84.24%	77.95%	66.26%	86.63%
800	87.61%	76.13%	65.67%	86.15%
1200	90.78%	73.25%	61.94%	85.23%

Table 6 | Compression rate increases as the maximum number of words allowed per page increases on QuALITY. Our default setting of min/max words is 280/600. In the other three experiments, we scale min words proportionally with max words.

B. Evaluation with GPT-3.5

Table 7 shows the results of running experiments using exactly the same setup as described in Section 4.3.1, but using GPT 3.5 Turbo rather than PaLM 2-L. GPT 3.5 Turbo has a context length of over 16,000 tokens, so the QuALITY dataset easily fits into context. We don't specifically tune prompts for GPT 3.5 Turbo, but instead use the same prompts that we use for PaLM 2-L. GPT 3.5 Turbo has a much harder time with this task than PaLM 2-L, but the same general trends hold. Neural Retrieval is weaker than ReadAgent. ReadAgent-S achieves comparable performance to using the full article content. The gap between ReadAgent-P and ReadAgent-S appears to be larger using this model, but we found that ReadAgent-P is very restrictive of how many pages to look up (1.0 in average) even though we allow up to 5. We think that this can likely be remedied if we engineer the prompt for GPT 3.5 Turbo. Nonetheless, comparing to using top 3 from neural retrieval, ReadAgent-P still yields better accuracy and compression rate.

Method	CR (# LU)	Accuracy
Neural Retrieval with Gemini API Top-3	73.13% (3)	69.22%
Full Raw Content	0%	73.30%
GistMem	84.24%	66.06%
ReadAgent-P Look up 1-5 pgs	76.60% (1.0)	69.65%
ReadAgent-S Look up 1-6 pgs	60.43% (3.4)	72.10%

Table 7 | QuALITY results on the dev set of 230 docs and 2086 questions using GPT-3.5-turbo. **CR** is the compression rate. **# LU** is the number of lookups. We report 1 run for each experiment for cost considerations.

C. Case Study

In this section, we analyze reading comprehension examples to demonstrate where the ability to simultaneously think over long-range global context and focus on local information is important. We selected the short story "off course" by Mack Reynolds[4] because it is extremely short (2,712 words) and it is only broken into 8 pages, yet even so, neural retrieval using 4 pages gets three questions wrong that ReadAgent correctly answers. For this story, ReadAgent answers 6 of 8 questions correctly. Neural retrieval answers 3 of 8 correctly, and doesn't get either question correct that ReadAgent misses. Note that in all three examples, ReadAgent only chooses to select two pages, even though it is also permitted to select up to 4. This flexibility is another advantage that ReadAgent has over standard retrieval systems.

"off course" Gist Memory

⟨P0⟩ Patrolmen Dermott and Casey encounter Dameri Tass, an alien who has landed on Earth. Dameri attempts to communicate with them using a device that translates his thoughts into English.
⟨P1⟩ The alien Dameri Tass used a helmet to learn English from Tim Casey, an Irish patrolman. He then became fascinated by a horse and wanted to use the helmet on the animal. Patrolman Dermott felt like he was in a shaggy dog story.
⟨P2⟩ A helicopter arrived, interrupting the horse's inspection. Two Army officers exited and ordered a police cordon around the spacecraft. The alien spoke, surprising the general. More police and military personnel arrived.
⟨P3⟩ Dameri Tass, an alien visitor, was whisked away to Washington and held incommunicado for several days. His arrival caused a global furor. Officials worried about the potential impact of his message on society. Eventually, the UN demanded that he be allowed to speak before the Assembly. The White House agreed and a date was set.
⟨P4⟩ The world eagerly awaited a message from space. Dameri Tass, an envoy from a super-civilization, was expected to guide the world. Most people were ready to be guided, but some were not. The U.N. Secretary-General was nervous about introducing the envoy, as they knew very little about him. He had been asleep for most of his time on Earth and had only recently woken up. He spent his time playing with a dog, cat, and mouse. The Secretary-General was worried about what the envoy would say.
⟨P5⟩ Dameri Tass, an alien, is brought to Earth and mistaken for an envoy from another planet. He reveals he is just a collector for a zoo.
⟨P6⟩ Dameri Tass, an alien, mistakenly landed on Earth. He addressed a large crowd, criticizing their weapons, wars, and lack of a planet-wide government. He then left, refusing to take any Earth creatures with him, but expressing interest in horses.
⟨P7⟩ The others watched as the first visitor from space hurriedly left Earth.

Distracting retrieval The first question gives an example of retrieval of distracting pages and the lack of global context provided by the gist memory causing the LLM to select the incorrect answer when using neural retrieval, even though it had also retrieved the pages that should have led to the correct answer. We provide the gist memory below and the story's pagination in Table 8.

"off course" Question 1

What was Dameri's purpose in landing on earth?
(A) He wanted to witness an uncivilized planet and share knowledge
(B) His spaceship needed to land for repairs
(C) He heard reports that Earth had interesting animal specimens for his collection
(D) He arrived on accident while exploring planets in the Galactic League
The correct answer is (D). ReadAgent chose (D). Neural retrieval chose (C).

Page #	Starting sentence in text
0	Shure and begorra...
1	The alien stooped down...
2	Interest in the horse was ended...
3	"Sure, and it's quite a reception"...
4	Excitement, anticipation...
5	"Here he comes,"...
6	He straightened and started off...
7	The others drew back...

Table 8 | Pagination for "off course".

[4]Available at http://aleph.gutenberg.org/3/0/0/3/30035//30035-h//30035-h.htm.

For the question above, ReadAgent looked up pages 5 and 6. Neural retrieval looked up pages 3, 4, 5, and 6. Pages 4 and 5 both make prominent mention of animals, and Page 5 explicitly mentions that the alien is a collector for a zoo, so answer (C) seems reasonable based on the information on those pages. However, Pages 5 and 6, together with the global context from the gist memory, make it clear that (D) is the correct answer. Since neural retrieval provided both of those pages, the lack of the global context combined with the additional distractor pages led the LLM astray.

"off course" Question 2

What happened to Dameri while he was in custody of the government?
(A) He picked up an accent from the guards
(B) He slept almost the entire time
(C) He learned horses were creatures that could be ridden
(D) He was too shy to speak
The correct answer is (B). ReadAgent chose (B). Neural retrieval chose (A).

Incorrect retrieval The same story provides two examples of the consequences of incorrect retrieval, and the benefits of the gist memory. For the question above, ReadAgent looked up pages 3 and 4. Neural retrieval looked up pages 0, 1, 3, and 6. The correct answer is clearly stated on Page 4, and also clearly stated in the gist of Page 4. If the LLM had access to either of those, it should have been able to answer correctly. Instead, it was undoubtedly confused by Pages 0 and 1, where the alien learns an accent from one of the police officers in the initial encounter.

"off course" Question 3

How did Dameri Tass communicate in English?
(A) He could communicate telepathically
(B) He never was able to communicate in English
(C) He used a handheld translation device
(D) He acquired the knowledge from a human
The correct answer is (D). ReadAgent chose (D). Neural retrieval chose (C).

For the question above, ReadAgent looked up pages 0 and 1. Neural retrieval looked up pages 0, 3, 4, and 6. The critical information was in Page 1, although Page 0 was also relevant. The remaining pages were only relevant in that they demonstrated that (B) was incorrect. Again, the gist memory was sufficient to answer the question correctly, in addition to providing clear signal about what pages are relevant to the question. But neural retrieval's selection of Page 0 without Page 1 made (C) seem plausible, as Page 0 discusses a device that the alien was clearly trying to use for communication.

D. ReadAgent for Web Navigation

We made an attempt to extend ReadAgent to decision making tasks. In particular, we apply ReadAgent for autonomous *web navigation* [38, 20, 13], where the goal is to autonomously control browsers or computer interfaces to complete tasks with natural language instructions provided by users. Such instruction would be something like *Book an appointment for applying new passport for one adult, Ellen Walker, with phone number 123-456-7890 and email address EW@gmail.com on April 4, 2023 at 1 pm in the post office nearest to zip code 60505. Don't send updates via text message*). Example web agent actions include *click*, *type*, and *select* (e.g. *click*, *type nearest post office*, *select April 4, 2023*). Because real-world websites can have very long HTML, LLM web agents often struggle with context length if it operates on raw content [15].

D.1. Implementation

Pagination For HTML, we leverage the explicit HTML DOM tree structure, decomposing the HTML into snippets with elements at a target depth and their descendants. We test the depth from 5 to 7 and choose the best. We use these snippets as the "pages" instead of asking the LLM to paginate.

Memory Gisting Similar to ReadAgent for reading comprehension, we prompt the LLM to summarize snippets into gists zero-shot, and subsequently concatenate the gists. We contextualize the gists with snippet index number in a python dictionary-format (e.g. {"index": ..., "content": ...}).

Interactive Look-up In the interactive look-up step, the LLM looks at a given task instruction, previous action history, and the gists to decide which original HTML snippets it wants to look up. We experimented with parallel look-up (ReadAgent-P) in the web navigation setting for faster experiments. Finally, to predict next-step actions, the LLM reads the retrieved snippets again and predicts the target element id to interact with, the type of action operation (click, type or select), and the input value (if any).

D.2. Mind2Web

We evaluate ReadAgent for Web Navigation on the Mind2Web [10] dataset, a real-world planning and web action prediction benchmark, consisting of 2K instructions and episodes collected from 137 websites. The agent's task is to predict the next-step action (click, type and select) given HTML, task instruction, and previous action history. Mind2Web has three test set splits: cross-task (252 tasks from 69 websites), cross-website (177 tasks from 10 websites), and cross-domain (912 tasks from 73 websites), which was originally designed for different testing different type of generalization. However, since our approach is zero-shot without training, these splits do not serve their original purposes.

Baselines MindAct from the Mind2Web paper [10] first uses a DeBERTa-base [17] model trained for task-relevant element retrieval to get the top 50 relevant elements. Instead of directly predicting target element id (part of an action), it formulates this task as iterative multi-choice question-answering with target element ids sampled from the top 50 and uses the LLM to solve it for performance purpose (see Deng et al. [10] for details). The same LLM also predicts the type of action and an optional value. MindAct (GPT-4) results are the state-of-the-art. We additional generate MindAct results with PaLM 2-L as a reference.

Following the reading comprehension experiments (Section 4), we also compare with using full raw HTML, retrieval with BM25, neural retrieval with Gemini API embedding model (models/embedding-001), and using the gists without look-up, which, like ReadAgent, are not trained for web navigation tasks. We ask the LLM to directly predict that target element id as it is a simpler and more tractable implementation in our setting.

	Cross-Task					Cross-Website					Cross-Domain				
	CR	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**	**CR**	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**	**CR**	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**
Using supervisedly trained RankLM															
MindAct (GPT-3.5 + Rank LM*)	–	20.3	56.6	17.4	0.8	–	19.3	48.8	16.2	0.6	–	21.6	52.8	18.6	1.0
MindAct (GPT-4 + Rank LM*)	–	**41.6**	60.6	**36.2**	2.0	–	35.8	51.1	30.1	2.0	–	37.1	46.5	26.4	2.0
MindAct (PaLM 2-L + Rank LM*)	–	29.8	61.9	24.4	1.2	–	28.8	59.6	21.6	0.6	–	29.9	60.4	24.5	1.3
No training															
(PaLM 2-L)															
+**Raw HTML**	0.0	22.1	<u>76.7</u>	19.2	1.2	0.0	22.2	72.3	18.2	1.7	0.0	23.6	75.6	20.9	1.0
+**BM25 Retrieval** (Top-1)	43.7	16.3	61.7	14.2	0.4	49.7	17.8	60.8	15.2	0.0	51.6	17.3	60.4	15.9	0.0
+**BM25 Retrieval** (Top-5)	19.5	25.9	70.4	22.4	2.0	17.6	29.5	71.8	23.1	1.7	19.2	27.6	71.1	24.4	1.0
+**Neural Retrieval** (Top-1)	74.4	14.6	55.5	11.7	0.4	87.9	18.0	55.8	14.0	0.0	82.8	16.4	60.3	14.2	0.1
+**Neural Retrieval** (Top-5)	32.4	26.4	71.9	22.6	0.8	37.2	26.7	69.1	22.3	2.8	38.1	30.0	72.5	26.9	1.2
GistMem	84.4	11.7	43.1	9.5	0.0	82.5	11.7	43.6	8.4	0.0	83.0	13.4	49.6	11.7	0.5
ReadAgent-P: Lookup 1 snippet	55.1	31.1	70.1	26.8	2.0	54.1	34.5	74.1	28.2	2.3	55.2	36.1	75.6	33.0	2.0
ReadAgent-P: Lookup 1-5 snippets	35.9	<u>33.7</u>	72.5	<u>29.2</u>	<u>**2.8**</u>	35.6	<u>**37.4**</u>	<u>**75.1**</u>	<u>**31.1**</u>	<u>**3.4**</u>	48.2	<u>**37.2**</u>	<u>**76.3**</u>	<u>**33.4**</u>	<u>**2.3**</u>
Δ(Raw – ReadAgent)	–	+11.6	-4.2	+10.0	+1.6	–	+15.2	+2.8	+12.9	+1.7	–	+13.6	+0.7	+12.5	+1.3
Δ(MindAct – ReadAgent)	–	+3.9	+10.6	+4.8	+1.6	–	+8.6	+15.5	+9.5	+2.8	–	+7.3	+15.9	+8.9	+1.0

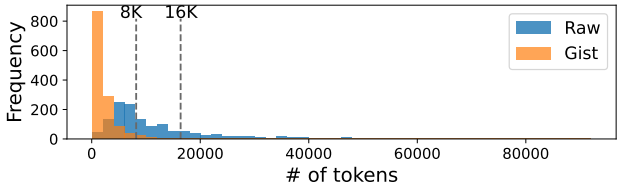
Table 9 | Web navigation performance on Mind2Web [10]. * marks models that are trained supervisedly for the web domain. GistMem and ReadAgent results are all also based on PaLM 2-L. We evaluate the performance in element accuracy (Ele. Acc), operation F1 (Op. F1), step success rate (Step SR), and episode success rate (SR). We also measure the compression rate (CR). The best performance across all the baselines is **bolded**, and the best across the approaches using PaLM 2-L is <u>underlined</u>. ReadAgent achieves consistently better performance than using raw HTML inputs (PaLM 2-L), retrieval methods, and MindAct (PaLM 2-L) with a trained Rank LM for HTML snippet retrieval.

D.3. Results

As shown in Table 9, ReadAgent achieves strong performance compared to the baselines. In particular, the results are even better than MindAct (PaLM 2-L), which uses the supervisedly learned Rank LM, despite ReadAgent not using models trained on the web navigation domain. Prior work shows that state-of-the-art LLMs alone are generally still weaker than the approaches using models specifically trained for the web navigation domain [12].

Figure 3 shows that gisting effectively reduces the number input tokens. Most of the input gists require less than 8K tokens. For example, 97.4% of gisted inputs in cross-website split fits into the 8k context length, while only 51.5% of raw HTML can fit in the context window. The inputs are truncated for the parts that exceed the context length limit, which can significantly impact performance.

The results in Figure 3 and Table 9 indicate that even using the gist memory and ReadAgent retrieval causes truncation on many web pages. This is because the retrieved snippets are quite large, causing the compression rate to drop substantially. In spite of those issues, the ReadAgent results give real gains over using the full context. This indicates that even the truncated gists and retrieved pages are more informative than the truncated raw HTML when using an LLM with a small context length.



Threshold	Raw	Gist
4096 Tokens	14.2%	88.6%
8192 Tokens	51.5%	97.4%
16384 Tokens	79.1%	100%
50th Percentile Tokens	8018	989
90th Percentile Tokens	25337	3596
95th Percentile Tokens	35779	5741
99th Percentile Tokens	55642	12569

Figure 3 | **(Left)** Histogram of raw HTML and gist tokens in the Mind2Web cross-website split. Most of the input gists require fewer than 8K tokens. **(Right)** Statistics of token counts of raw HTML and gists.

E. ReadAgent Variants

E.1. Unconditional and Conditional ReadAgent

When working with a long text, it is possible that the user will know ahead of time what task is to be solved. In that case, conceivably the gisting step could include the task description in the prompt. In so doing, it is easy to imagine that the LLM could do a better job of compressing out information that is irrelevant to the task, thereby improving efficiency and reducing distraction. This approach would be *Conditional ReadAgent*. However, more generally, the task may not be known while preparing the gists, or it may be known that the gists need to be used for multiple different tasks, such as answering many questions about the text. Thus, by excluding the task in the gisting step, the LLM may produce more broadly useful gists, at the cost of reduced compression and increased distracting information. This setting would be *Unconditional ReadAgent*. We only explore the unconditional setting in this work, but we note that the conditional setting may be preferred in some situations.

E.2. Iterative Gisting

For a very long event history, such as a conversation, we may consider further compression of the older memory with iterative gisting to allow having longer contexts, similar to older memories of humans being fuzzier. Though this is not in the scope of this work, it may be useful for applications such as assistant agents.

F. Comparing ReadAgent and MemWalker

As discussed in Section 2, similar to our work, MemWalker [5] also reads long documents interactively like an agent through iterative prompting, instead of forcing LLMs to process everything at once. It first constructs a summary tree where the lowest-level leafs are segments of raw text, the second level nodes are summaries of

text segments, and the higher levels are summaries of summaries. Given a task, it traverses the tree from the root to search for task-related information. We think there are a few reasons to prefer the ReadAgent approach over MemWalker.

First, the reliability is a concern. Having LLMs traverse summary tree may not be a reliable process. In our best-effort re-implementation of MemWalker with PaLM 2-L, it unsatisfyingly achieves 66.73% on QuALITY. To put that into perspective, using full raw content is 85.83%, ReadAgent-P (look up 1-5 pages) is 86.63%, ReadAgent-S (look up 1-6 pages) is 86.88%, and using BM25 Top-1 is 70.55%. Part of the performance difference is caused by a high search failure rate. 11.7% of the searches failed to finish after sufficient retries. This failure rate of our implementation is in a similar range to what the authors reported: 91.4% successes and 8.6% failures[5]. In contrast, the failure rate of ReadAgent is mostly 0%.

Second, the hierarchical summary structure makes it difficult to reason over related but distant information at the same granularity. There isn't much detail preserved at the top levels of the hierarchy. For example, if the two most important text pieces are at the beginning and the end of a very long text, the essential information could be in the first and last leaf. As the agent traverses down to the first leaf, it could be difficult to go back up to the root and down to the last leaf.

The motivations of the two approaches are also different. MemWalker interacts with a summary tree and reasons over traversal trajectories, whereas ReadAgent interacts directly with documents and reasons over gist memories.

G. Pagination Hyperparameters

Pagination Details As described in Section 3.1, `max_words` and `min_words` are two episode pagination hyperparameters. Table 10 gives their values for each of the experiments in Section 4.

Dataset	$\frac{\texttt{max\_words}}{\texttt{min\_words}}$
QuALITY	$\frac{600}{280}$
QMSum	$\frac{600}{280}$
NarrativeQA Gutenberg	$\frac{3000}{500}$
NarrativeQA movie scripts	$\frac{1000}{600}$

Table 10 | Pagination hyperparameters.

H. NarrativeQA Addtional Details

Figures 4 and 5 show histograms of word counts for the two NarrativeQA subsets and their corresponding gist memories.

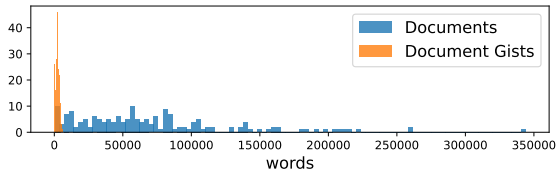


Figure 4 | Histogram of NarrativeQA (Gutenberg) test set word counts for the original text and the gists.

Context Length Control As the NarrativeQA Gutenberg texts can be very long, the corresponding gists can sometimes exceed the context length. For those exceptionally long texts, we ask the LLM to go through the pages and think whether it makes sense to merge pages iteratively with the following prompt, and then re-gist

[5]https://openreview.net/forum?id=H5XZLeXWPS

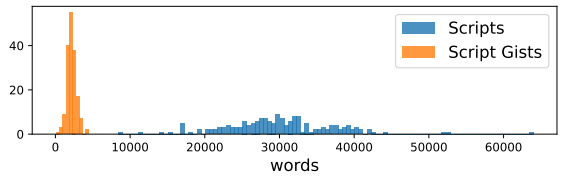


Figure 5 | Histogram of NarrativeQA (movie) test set word counts for the original text and the gists.

the new set of pages. In so doing, we are able to increase the average page size and thus the compression rate (Figure 6).

Example NarrativeQA Gutenberg Page Merging Prompt

Given Page 1 and Page 2, please tell me whether Page 2 starts a new chapter/section/book that is different from what's in Page 1.

Please answer with yes, no, or not sure.

Page 1:
{PREVIOUS PAGE TEXT}

Page 2:
{CURRENT PAGE TEXT}

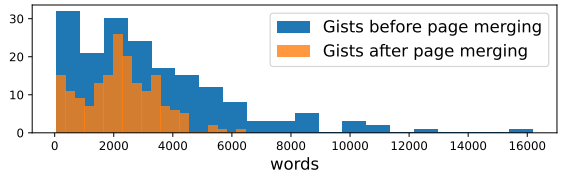


Figure 6 | Histogram of NarrativeQA (Gutenberg) test set gists before and after page merging on the exceptionally long texts.

The gists and pages can both be long for NarrativeQA. Thus, in the interactive look-up step of ReadAgent-P, we prevent the retrieved pages from exceeding the context length by asking the model to sort the pages by importance with the prompt below and iteratively detecting whether adding any pages could go beyond the context window. For ReadAgent-S, we do a similar check to decide whether to early-stop the sequential look-up.

Example Parallel Lookup Prompt (ReadAgent-P) for NarrativeQA

The following text is what you remember from reading an article and a question related to it.

You may read 1, 2 or 3 page(s) of the article again to refresh your memory to prepare yourself for the question.

Please respond with which page(s) you would like to read in the order of importance, beginning with the most important page number.

For example, if you only need to read Page 8, respond with "I want to look up Page [8] to ...".

If you would like to read Page 12 and 7, respond with "I want to look up Page [12, 7] to ...".

If you would like to read Page 15, 2 and 3, respond with "I want to look up Page [15, 2, 3] to ...".

DO NOT select more pages if you don't need to.

You don't need to answer the question yet.

Text:
{GIST MEMORY}

Question:
{QUESTION}

I. Additional QMSum Results

Figure 7 shows the histogram of word counts on the QMSum training set and the corresponding gist memories.

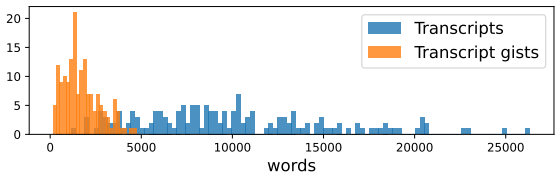


Figure 7 | Histogram of QMSum word counts for the original transcripts and the gisted transcripts. The gisted transcripts are all less than 5,000 words, allowing them to entirely fit into the context windows of PaLM 2-L.

Table 11 shows the same results as Table 3, but on the QMSum test set.

Method	CR (# LU)	LLM Rating-1	LLM Rating-2	ROUGE-1	ROUGE-2	ROUGE-L	Resp. Length
BM25 Retrieval							
Top-1	95.61% (1.00)	24.67% ± 0.44	66.90% ± 0.87	28.81 ± 0.13	8.14 ± 0.15	19.62 ± 0.18	48.15 ± 0.18
Top-2	91.32% (2.00)	31.79% ± 1.31	79.95% ± 0.67	30.89 ± 0.13	9.14 ± 0.05	20.67 ± 0.09	53.91 ± 0.64
Top-3	87.25% (3.00)	33.45% ± 0.00	83.63% ± 1.05	31.39 ± 0.23	9.11 ± 0.05	21.03 ± 0.03	55.15 ± 0.61
Top-4	82.86% (4.00)	37.72% ± 1.05	86.12% ± 0.50	31.71 ± 0.09	9.35 ± 0.13	21.26 ± 0.12	58.21 ± 0.37
Top-5	78.79% (5.00)	39.38% ± 1.02	86.60% ± 0.44	32.66 ± 0.04	***9.98*** ± 0.10	**21.86** ± 0.05	59.20 ± 1.05
Top-6	74.62% (6.00)	40.45% ± 0.89	90.98% ± 0.34	32.56 ± 0.03	9.78 ± 0.03	21.64 ± 0.09	60.40 ± 1.28
Neural Retrieval with Gemini API							
Top-1	95.80% (1.00)	27.05% ± 0.50	67.97% ± 1.74	28.71 ± 0.12	7.98 ± 0.04	19.59 ± 0.04	49.76 ± 0.78
Top-2	91.62% (2.00)	35.35% ± 0.44	80.07% ± 0.00	31.65 ± 0.18	9.59 ± 0.11	21.29 ± 0.11	56.19 ± 0.76
Top-3	87.39% (3.00)	35.71% ± 1.37	88.49% ± 0.34	32.33 ± 0.17	9.84 ± 0.07	21.54 ± 0.13	59.19 ± 0.96
Top-4	83.28% (4.00)	39.62% ± 0.17	90.15% ± 0.34	32.31 ± 0.21	9.69 ± 0.15	21.65 ± 0.15	59.86 ± 0.11
Top-5	79.33% (5.00)	44.01% ± 0.84	91.22% ± 0.34	32.33 ± 0.24	***9.84*** ± 0.21	21.67 ± 0.19	61.53 ± 0.35
Top-6	75.35% (6.00)	44.60% ± 0.89	92.65% ± 0.17	32.55 ± 0.08	9.75 ± 0.21	21.39 ± 0.13	61.29 ± 0.46
Truncated Raw Content							
First 6k words	31.51% (0.00)	13.17% ± 1.05	47.81% ± 5.90	24.15 ± 1.42	4.89 ± 0.57	16.27 ± 0.96	61.43 ± 3.53
Last 6k words	33.80% (0.00)	13.76% ± 0.84	43.42% ± 0.00	22.90 ± 0.10	4.35 ± 0.04	15.69 ± 0.03	52.47 ± 0.39
GistMem	82.81% (0.00)	44.96% ± 0.44	91.93% ± 0.73	31.20 ± 0.17	9.02 ± 0.09	20.60 ± 0.14	65.84 ± 0.87
ReadAgent-P							
Look up 1 pg	79.37% (0.98)	44.84% ± 0.00	92.29% ± 0.34	31.46 ± 0.12	9.09 ± 0.11	20.63 ± 0.05	66.74 ± 0.74
Look up 1-2 pgs	77.00% (1.72)	43.42% ± 1.01	92.88% ± 1.05	31.77 ± 0.16	9.11 ± 0.12	20.70 ± 0.08	65.55 ± 0.28
Look up 1-3 pgs	74.85% (2.46)	44.37% ± 1.21	91.22% ± 0.44	31.89 ± 0.06	8.98 ± 0.13	20.70 ± 0.09	66.06 ± 1.63
Look up 1-4 pgs	73.26% (3.02)	44.13% ± 0.50	90.51% ± 0.44	31.87 ± 0.07	9.12 ± 0.06	20.77 ± 0.01	66.44 ± 0.74
Look up 1-5 pgs	72.01% (3.44)	43.42% ± 1.45	91.22% ± 0.60	31.80 ± 0.16	9.03 ± 0.07	20.64 ± 0.03	66.48 ± 0.39
Look up 1-6 pgs	70.65% (3.89)	42.70% ± 1.54	90.51% ± 0.73	31.74 ± 0.09	8.90 ± 0.09	20.66 ± 0.16	66.24 ± 1.14
ReadAgent-S 1-6 pgs	70.75% (3.42)	**49.58%** ± 0.44	**93.83%** ± 0.34	**32.88** ± 0.15	***9.98*** ± 0.06	21.50 ± 0.04	67.86 ± 0.11

Table 11 | **QMSum test** results (PaLM 2-L) means and standard deviations across 3 runs. 35 articles and 281 questions. **Bold methods** are this work. **Bold** values are the best; ***bold italics*** are ties for best. **CR** is the compression rate. **# LU** is the number of lookups. **Resp. Length** is the length in words of the model's final response. We omit standard deviations for CR and # LU for presentation purposes; they were all inconsequential.

J. Author Contributions

Kuang-Huei Lee developed the initial working prototype, the method and the experiments on QuALITY and NarrativeQA, was a main writer of the manuscript, and led the project overall.

Xinyun Chen developed the method, the LLM rater, and experiments on the NarrativeQA, and significantly contributed to manuscript writing.

Hiroki Furuta developed the web navigation experiments, and significantly contributed to manuscript writing.

John Canny contributed in the initial conceptualization, advised the project, and helped with manuscript editing.

Ian Fischer co-proposed the core idea, developed the method and experiments on QMSum, and was a main writer of the manuscript.