Superposition Prompting: Improving and Accelerating Retrieval-Augmented Generation

Thomas Merth[1] **Qichen Fu**[1] **Mohammad Rastegari***[2] **Mahyar Najibi**[1]

Abstract

Despite the successes of large language models (LLMs), they exhibit significant drawbacks, particularly when processing long contexts. Their inference cost scales quadratically with respect to sequence length, making it expensive for deployment in some real-world text processing applications, such as retrieval-augmented generation (RAG). Additionally, LLMs also exhibit the "distraction phenomenon", where irrelevant context in the prompt degrades output quality. To address these drawbacks, we propose a novel RAG prompting methodology, *superposition prompting*, which can be directly applied to pre-trained transformer-based LLMs *without the need for fine-tuning*. At a high level, superposition prompting allows the LLM to process input documents in parallel *prompt paths*, discarding paths once they are deemed irrelevant. We demonstrate the capability of our method to simultaneously enhance time efficiency across a variety of question-answering benchmarks using multiple pre-trained LLMs. Furthermore, our technique significantly improves accuracy when the retrieved context is large relative the context the model was trained on. For example, our approach facilitates an $93\times$ reduction in compute time while *improving* accuracy by 43% on the NaturalQuestions-Open dataset with the MPT-7B instruction-tuned model over naive RAG.

1. Introduction

Transformer-based autoregressive large language models (LLMs) have led to quantum leaps in text modeling performance over previous methods (Zhao et al., 2023). However, they have massive compute requirements, especially as the context length increases due to the quadratic compute cost of self-attention. Many prior works have explored how to accelerate LLM inference (Huang et al., 2023; Miao et al., 2023). However, such optimizations often require significant architectural or parameter modifications to the pre-trained model, thus mandating expensive re-training or fine-tuning procedures. In addition to causing undesirable compute requirements, long input contexts can also lead to hallucinations and/or divergent responses in model outputs (Liu et al., 2023a; Shi et al., 2023).

Retrieval-augmented generation (RAG) is one alluring application of transformer-based LLMs. In this setting, the LLM can ground its responses in auxiliary, relevant context. Often, the retrieved documents contain long-form text, leading to the aforementioned downsides (Gao et al., 2023). To improve and accelerate RAG, we propose *superposition prompting* [1]. Superposition prompting is demonstrated to

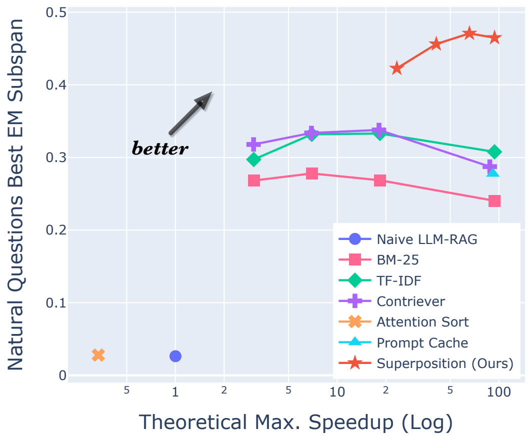


Figure 1. Theoretical Maximum Speedup vs. Accuracy (Best EM Subspan) on NaturalQuestions-Open using the `mpt-7b-instruct` model (Muennighoff et al., 2023). Refer to Section 4.1.1 for experimental details. Plotted values are sourced from Table 1 and Table 5.

[1]Apple, Cupertino, CA, USA [2]Meta, Menlo xPark, CA, USA (*Work done while at Apple). Correspondence to: T. Merth <tmerth@apple.com>, Q. Fu <qfu22@apple.com>, M. Rastegari <mrastegari@meta.com>, M. Najibi <najibi@apple.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]We drew inspiration from the "path integral" formulation of quantum mechanics (Feynman, 1965), where a particle's dynamics

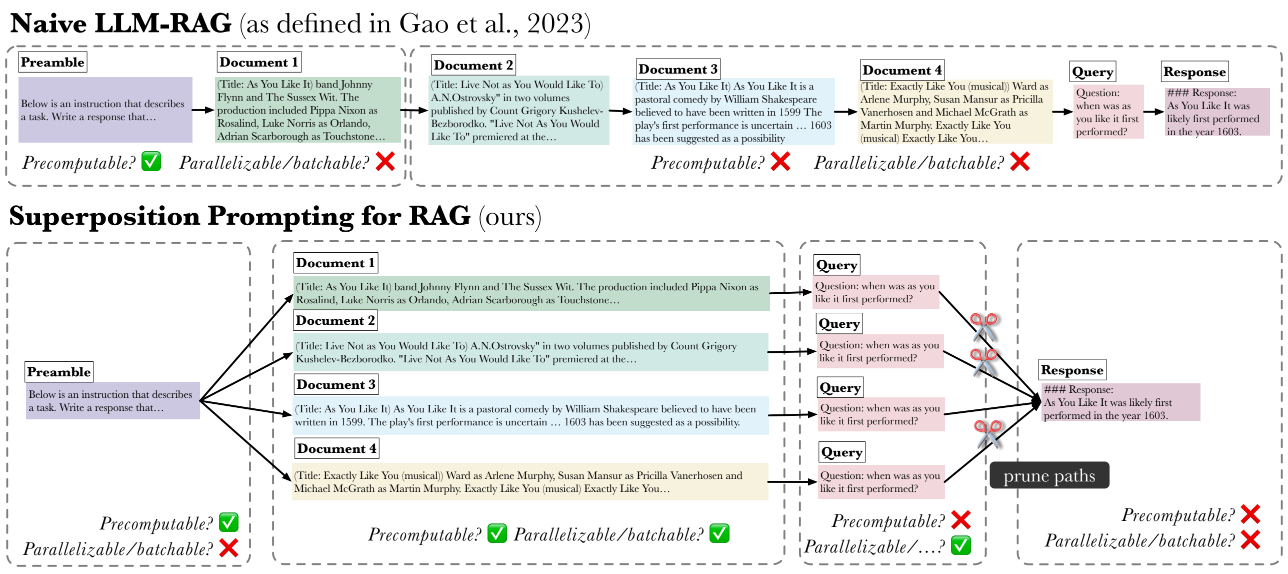

Naive LLM-RAG (as defined in Gao et al., 2023)
Preamble
Below is an instruction that describes a task. Write a response that…
Document 1
(Title: As You Like It) band Johnny Flynn and The Sussex Wit. The production included Pippa Nixon as Rosalind, Luke Norris as Orlando, Adrian Scarborough as Touchstone…
Document 2
(Title: Live Not as You Would Like To) A.N.Ostrovsky" in two volumes published by Count Grigory Kushelev-Bezborodko. "Live Not As You Would Like To" premiered at the…
Document 3
(Title: As You Like It) As You Like It is a pastoral comedy by William Shakespeare believed to have been written in 1599 The play's first performance is uncertain … 1603 has been suggested as a possibility
Document 4
(Title: Exactly Like You (musical)) Ward as Arlene Murphy, Susan Mansur as Pricilla Vanerhosen and Michael McGrath as Martin Murphy. Exactly Like You (musical) Exactly Like You…
Query
Question: when was as you like it first performed?
Response
### Response: As You Like It was likely first performed in the year 1603.
Precomputable? ✅ Parallelizable/batchable? ❌
Precomputable? ❌ Parallelizable/batchable? ❌
Superposition Prompting for RAG (ours)
prune paths
Precomputable? ✅ Parallelizable/batchable? ❌
Precomputable? ✅ Parallelizable/batchable? ✅
Precomputable? ❌ Parallelizable/…? ✅
Precomputable? ❌ Parallelizable/batchable? ❌

Figure 2. Comparison of superposition prompting vs. the "classical" (Naive LLM-RAG) prompting paradigm. Squares represents a token, and arrows depict attention dependencies. Whereas the classical approach is a "linked-list" style DAG, superposition prompting arranges token dependencies such that all documents are processed independently. Due to this dependency structure, we can easily leverage the LLM logits to prune irrelevant context, improving long context reasoning. The dependency structure also allows for faster prompt processing, due to the new opportunities for caching and parallelism of the KV cache and logit computations (each gray box represents, logically, a "batch" that is processed by the LLM, reusing upstream KV caches).

simultaneously improve model accuracy and compute time efficiency for RAG-based question answering tasks *without any additional training or fine-tuning*. To highlight our results, we refer the reader to Figure 1.

In this work, our contributions are as follows; (1) we propose a new generalized framework for prompting LLMs in RAG scenarios, (2) we demonstrate the benefit of our method on question-answering datasets, and (3) we provide extensive experimental evidence and ablation studies to give more confidence to our design decisions. We also propose additional practical optimizations to accelerate inference by pruning, caching, and parallelizing the compute of *prompt paths*. These optimizations are made possible due to the topological structure of our superposition prompts.

For reproducibility, our implementation can be found at `https://github.com/apple/ml-superposition-prompting`.

2. Related Work

Retrieval Augmented Generation. Retrieval-augmented generation (RAG) is a common application of LLMs to generate answers to questions based on a set of retrieved documents (Lewis et al., 2020). Instead of simply prompting a language model with a query, RAG augments the prompt by injecting a set of retrieved documents into the prompt. If done correctly, these documents contain useful knowledge related to the query, which should elicit more accurate and reliable output from the model. Extensive work (Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2021b; Gao et al., 2023; Asai et al., 2023) has shown RAG to be effective for many knowledge-intensive tasks (Petroni et al., 2020). However, incorporating retrieved documents significantly extends the input sequence length and introduces additional computational overhead, raising efficiency concerns. Addressing the challenges of long context processing and efficiency for RAG has become a key focus in recent research (Guu et al., 2020; Beltagy et al., 2020; Ratner et al., 2022).

Efficient Long Context Processing. There have been significant efforts to reduce the memory footprint and computational costs of transformers using techniques such as compression and KV-caching (Sheng et al., 2023; Lin et al., 2023; Xiao et al., 2022). More efficient versions of the transformer architecture have also been explored. For example, Longformer (Beltagy et al., 2020) introduced a drop-in replacement to the standard self-attention, which makes it scale linearly with sequence length. Similarly, Reformer (Kitaev et al., 2020) uses locality-sensitive hashing to reduce the complexity of attention and improve its efficiency for long sequences. In parallel, the SparseTransformer (Child et al., 2019) focuses on the sparsity of the attention layers.

can be represented as a weighted sum over possible trajectories. Analogously, the language dynamics of superposition prompting are modeled by a weighted sum of possible "token trajectories".

While the above innovation addresses the efficiency of long context processing, they often require non-trivial architecture changes and/or re-training. This makes them impractical to use with existing, pre-trained LLMs (Touvron et al., 2023; Zhang et al., 2022). Closer to our work are efficient methods which optimize KV caching and consider token importance (Zhang et al., 2023; Liu et al., 2023c). Other works (orthogonal to ours) investigate how to improve efficiency of LLM output generation (Ning et al., 2023). The above methods differ from ours as they investigate acceleration for LLMs generally, whereas we aim to leverage the specifics of the RAG setting to achieve further improvements.

The closest to our work is the recently proposed Prompt Cache (Gim et al., 2023). This method also leverages the modular structure of the RAG setting to perform local attention on the preamble and documents independently and cache the results. In contrast, our method retains attention dependencies *between* segments in the form of a dependency graph. Also differentiating, we propose pruning and parallelization mechanisms not explored by Gim et al., 2023.

Prompt Engineering. Prompt engineering is the process of deliberately designing and tuning prompts before feeding them to language models to generate text (Liu et al., 2023b). Prior exploration (Bubeck et al., 2023) shows how careful prompt construction can greatly improve the model's responses. Intriguingly, the recent work "Lost in the Middle" (Liu et al., 2023a) has shown that solely the *location* of the "golden document" (the document containing the answer) within a long context significantly affects the performance of language models. Another theme of prompt engineering works has explored how to use graph-like structures while prompting LLMs. Our proposed method might seem, at first glance, identical to other "tree-based" and "graph-based" prompting methods, such as Tree of Thoughts (Yao et al., 2023) and Graph of Thoughts (Besta et al., 2023). However, these methods are proposed in the context of multi-step reasoning, not RAG. Different from the above, Parallel Context Windows (Ratner et al., 2022)—along with other "structured attention" works (Cai et al., 2023; Ye et al., 2023)—aims to build dependencies between prompt text segments. However, these works were generally applied to few-shot learning applications, not retrieval-augmented generation. Our approach also differs from these structured attention papers in that we operate on generalized directed-acyclic graphs, as opposed to just the special case of trees.

3. Proposed Method

The retrieval-augmented generation task is comprised of distinct text segments—the preamble (*a.k.a.* system prompt), a (static) corpus of documents, and a (dynamic) user-supplied query. Instead of concatenating these text segments in textual space, we group them into separate "batches" (the gray boxes in Figure 2), which are passed as calls to the LLM (re-using the KV caches from upstream token segments). With a query as input, superposition prompting processes all choices of documents paired with the query independently (conditioned on the preamble)—in Figure 2, this can be seen as the branching structure. Once the query batch is processed, we then employ *path pruning* (Section 3.2.3) to discard entire attention dependencies based on an importance metric (the scissors in Figure 2). Both of these optimizations improve inference efficiency and enable the model to discard distracting documents unrelated to the query.

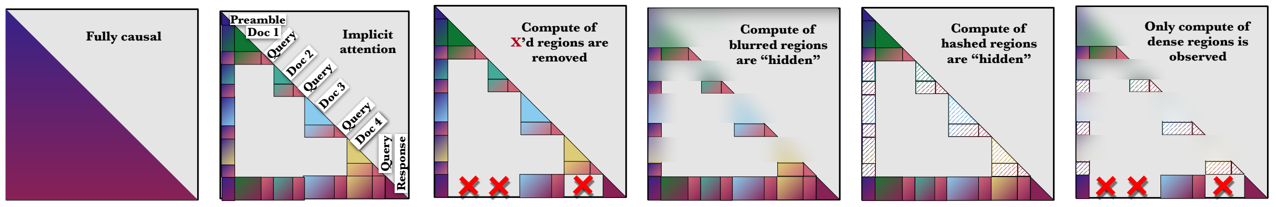
Enabled by the added structure of our superposition prompting approach, we then propose techniques to further accelerate the inference. First, the high-level notion of token sharing across prompts allows us to employ *prompt path caching* (Section 3.3.1). Finally, we describe a *prompt path parallelization* (Section 3.3.2) strategy that leverages independence across segments.

3.1. Retrieval Augmented Generation

We stylize token sequences as bolded vectors and use $\oplus$ to denote concatenation along the sequence dimension. Supposing there are n_d (pre-tokenized) offline documents available for retrieval, we define the set of document token sequences $\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n_d}\}$. We denote the user query as $\boldsymbol{q}$, and our custom preamble sequence as $\boldsymbol{p}$. The goal is to return some response $\boldsymbol{r}$ which answers the query, all while minimizing the latency between the arrival of the query and the serving of the response as observed by the client. The obvious baseline solution (which we refer to as "Naive LLM-RAG") is where one simply concatenates the input sequences as $\boldsymbol{x} = \boldsymbol{p} \oplus \boldsymbol{d}_1 \oplus \cdots \oplus \boldsymbol{d}_{n_d} \oplus \boldsymbol{q}$, then autoregressively generates $\boldsymbol{r}$ using $\boldsymbol{x}$ as the prompt. However, as shown in Section 4, our approach massively outperforms such a baseline both in terms of quality and performance.

3.2. Superposition Prompting

We now detail superposition prompting, a new paradigm for prompting language models. In superposition prompting, prompts are *not* represented as a simple sequence of tokens as they are with "classical" prompting methods (*e.g.* Naive LLM-RAG). Rather, superpositioned prompts are directed acyclic graphs (DAGs) where nodes are token sequences, and edges codify attention dependencies. Plainly put, a particular token sequence, $\boldsymbol{v}$, attends to the tokens in another token sequence, $\boldsymbol{u}$, if and only if there is a path from $\boldsymbol{u}$ to $\boldsymbol{v}$ in the DAG. In this sense, superposition prompting is a generalization of "classical" prompting (since a "classical" prompt is the linked-list special case). Please refer to Algorithm 3 for an algorithmic formalization.


Fully causal
Preamble
Doc 1
Query
Doc 2
Query
Doc 3
Query
Doc 4
Query
Response
Implicit attention
Compute of X'd regions are removed
Compute of blurred regions are "hidden"
Compute of hashed regions are "hidden"
Only compute of dense regions is observed

(a) Baseline (b) Superposition prompting (c) w/ Path Pruning (d) w/ Path Caching (e) w/ Path Parallelization (f) w/ All

Figure 3. Implicit attention dependencies that must be computed during "online serving" (the colors in (b)-(f) correspond to the token segment colors in Figure 2). Note how the various optimizations reduce the computational burden required at online serving-time by pruning, precomputing, and parallelizing the work. It is worth re-emphasizing that in practice, inference is *not* sparse attention on one large sequence, but rather *dense attention* with many different shorter token segments.

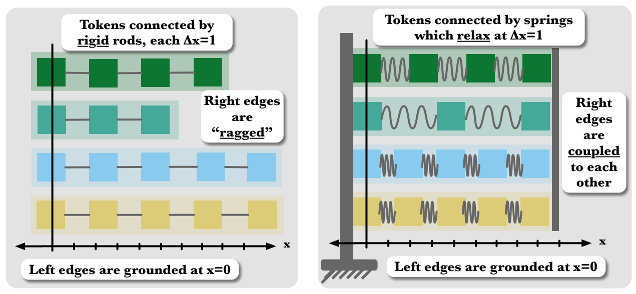
3.2.1. The ForkJoin Prompt Path Topology

In order to leverage superposition prompting for RAG, we must construct a graph topology out of our text segments. We propose the *ForkJoin* graph structure, depicted in Figure 2. Note that each $\boldsymbol{q}_i$ sequence is a duplicate of the original $\boldsymbol{q}$ (Section 3.2.3 will justify this decision). Although this duplication ends up increasing the number of tokens processed, our ablation analysis in Appendix B.1 demonstrates the superiority of this approach in terms of accuracy. Furthermore, Section 3.3.2 describes how the cost of this duplication can be entirely hidden from the user. The ForkJoin topology (implicitly) results in a pseudo-"local attention" structure (Figure 3). We emphasize that this resulting attention pattern is a construct for visualization only—in reality, all calls to the LLM use fully dense attention, although on relatively smaller context lengths. Concretely, each of the dashed boxes in Figure 2 is a separate LLM call.

3.2.2. Token Position Assignment

With classical prompting, tokens are (by default) spaced equally, with a distance of 1. However, with superposition prompting, positioning tokens is not trivial, since paths (of potentially heterogeneous length) run parallel to each other. Thus, we seek to answer the question, "how do we assign meaningful positions to tokens in superposition prompts?"

One simple approach could be to truncate token sequences to a common length to enforce a shared positioning. However, truncating may result in loss of input signal if the trimmed tokens contained valuable information for the query. Another approach could be to *left-align* (or right-pad) sequences to a common length (Gim et al., 2023). While this *left aligned* padding approach is simple, it yields discontinuities in the prompt sequence position assignments (see Appendix E for quantification). With the ALiBi encoding scheme (Press et al., 2021), it can be easily shown that discontinuities unfairly assign attention penalties to the tokens in shorter documents, since the tokens will be concentrated at earlier token positions (and thus larger distances from the current token).[2] Thus, we are motivated to propose a positional assignment strategy that does not result in discontinuities.


Tokens connected by rigid rods, each Δx=1
Right edges are "ragged"
x
Left edges are grounded at x=0
Tokens connected by springs which relax at Δx=1
Right edges are coupled to each other
x
Left edges are grounded at x=0

(a) Left Aligned (b) Equilibrium (Ours)

Figure 4. Visual intuition for our proposed *equilibrium position assignment* vs. left aligned (see Section 3.2.2).

We propose *path equilibrium positioning* as one simple, reasonable strategy. With path equilibrium positioning, we linearly space overlapping paths to fit the harmonic mean, $S(\boldsymbol{D})$, of their collective lengths (for a set of overlapping paths $\boldsymbol{D}$)

$$S(\boldsymbol{D}) = \frac{n_d}{\sum_{\boldsymbol{d}\in\boldsymbol{D}} \frac{1}{\|\boldsymbol{d}\|}} \tag{1}$$

Intuitively, the resulting token positions matches the equilibrium state of coupled masses connected by springs (Figure 4).

Note that the path equilibrium positioning strategy results in real-valued positions. This is a departure from common usage of token position assignments, where integer-valued positions are predominant.[3] We note that the choice of position assignment scheme has no effect on inference effi-

[2]An analogous bias would exist if using a *right aligned* strategy, except shorter documents would be unfairly assigned attention boosts over longer documents.

[3]While this is trivial for the ALiBi positional encoding, it is non-trivial for the Rotary Position Embedding (Su et al., 2021) scheme. To handle this case, we linearly interpolate the *argument* of the sinusoidal functions.

ciency, but can impact model output quality. In Section 5.1, we validate the superiority of path equilibrium positioning.

3.2.3. Path Pruning

Further exploiting the topological structure we've imposed on the prompt, we propose *path pruning* as a mechanism to discard documents it sees as irrelevant to the query. As demonstrated in our experiments, this can benefit both efficiency and accuracy for LLM-based RAG.

In order to prune paths, we must compute a saliency score for each path. Inspired by SGPT (Muennighoff, 2022), we apply Bayes rule to the output of the language modeling head to compute a saliency or entailment score. At a high level, we leverage Bayes' theorem to compute the posterior distribution

$$P(\boldsymbol{d}_i \mid \boldsymbol{q}_i, \boldsymbol{p}) \propto P(\boldsymbol{q} \mid \boldsymbol{d}_i, \boldsymbol{p}) P(\boldsymbol{d}_i \mid \boldsymbol{p})$$

as a saliency metric of document $\boldsymbol{d}_i$'s relevancy to the query.[4] In our experiments, we decide which path indices to prune by greedily selecting the top-k of this categorical distribution (we perform ablations with respect to the choice of k in Section 4.1.2 and Appendix A.1).

To "physically" apply the pruning, we can simply discard the KV caches corresponding to the documents and queries along those paths. Conveniently, all remaining KV caches can be simply concatenated together for use in autoregressive generation of the response.

We provide ablations against other reasonable saliency metrics in Section 5.2. A visual representation of the effect of path pruning on the (implicit) attention patterns can be also seen in Figure 3c.

3.3. Lossless Runtime Optimizations

3.3.1. Path Caching

Assuming auxiliary memory storage is available, we can accelerate inference of superposition prompts by doing work before any query has arrived. This *path caching* technique generalizes the ideas put forth in PagedAttention (Kwon et al., 2023), where we cache the KV embeddings along all path prefixes (not just the "root node", as PagedAttention does). Importantly, our approach also differs from PromptCache (Gim et al., 2023). While their cached "prompt modules" only attend locally to themselves, our path prefix KV caches attend locally to themselves *as well as* to all their ancestors in the graph. Please refer to Algorithm 2 in appendix for formalization.

We now describe our path caching mechanism. Concretely, the preamble KV cache and document KVs are not conditioned on the query, and thus can be precomputed during a "preprocessing" stage. Then, during the "online serving" stage, we retrieve the preamble and document KVs from storage instead of the original input token sequences. Figure 3d shows how, during the online serving stage, much of the attention dependencies have already been computed. Note that the memory requirement for employing path caching is a scalar multiple, c_{model}, of the raw tokenized sequence length. Here, c_{model} is a fixed scalar that depends on the various aspects of the models, such as number of layers, and embedding dimension (*e.g.* $c_{\texttt{bloom-7b1}} = 492$ KB).

3.3.2. Path Parallelization

Since the superpositioned paths of *ForkJoin* are independent of each other (by construction), the corresponding KV caches and logits of the query segments can be computed in parallel. While this does not reduce the "CPU time," it importantly reduces the *wall-clock time* experienced by the user. The parallelization across the duplicated queries can be accomplished either by (1) concatenating sequences along the batch dimension before inference[5] (2) delegating model calls across a distributed cluster of compute nodes (*e.g.* GPUs), or (3) a combination of batching and distributed inference. The most effective strategy will depend on the specifics of the cluster configuration (*e.g.* relative network bandwidth vs. available compute per node).

4. Experimental Results

We perform experiments on three families of large language models, namely OpenELM (Mehta et al., 2024), BLOOMZ (Muennighoff et al., 2023), and MPT (MosaicML NLP Team, 2023). To quantify the effectiveness of superposition prompting when paired with models of different scales, we use various model sizes from these families. For OpenELM, we use the `3B-Instruct` configuration. For BLOOMZ, we instantiate 3B parameter (`bloomz-3b`) and 7.1B parameter (`bloomz-7b1`) models. Finally, for MPT, we use the available instruct fine-tuned 7B parameter model (`mpt-7b-instruct`). This set of models covers different architectures, positional encoding schemes, sizes, and pretraining recipes. We remind the reader we use the publicly released pretrained checkpoints, without employing any additional training, fine-tuning, or task adaptation.

For our experiments, we are primarily interested in the compute time vs. accuracy tradeoff.[6] We use the `fvcore`

[4]This resembles the "principle of least action," which determines the optimal path weighting in the path integral formulation of quantum mechanics.

[5]In general, prompt path lengths will vary, thus requiring padding to a common length. However, a length binning strategy may alleviate most overhead in practice.

[6]In our timing and speedup analysis, we follow previous works (Gim et al., 2023) and do not consider the data retrieval portion of the RAG pipeline, which would require too many assumptions.

(facebookresearch, 2024) package to compute theoretical floating point operation (FLOP) counts for various inference settings. We evaluate the compute cost of each method in units of *compute cycles*—similar to FLOPs, but accounting for parallelism. In practice, to achieve the speedups, extra resources (auxiliary memory and/or auxiliary compute for parallelization) will be required. However, as stated, the goal of this exploration is *acceleration*, not necessarily FLOPs reduction. We refer the reader to Appendix A.2 for detailed breakdowns of the theoretical speedup gains enabled by each of our proposed optimizations.

4.1. Results

We leverage the publicly available NaturalQuestions-Open (Liu et al., 2023a) and MuSiQue (Trivedi et al., 2022) datasets. We do not perform any manual prompt tuning or prompt engineering for any method or baseline, and use the same prompts across all experiments (per dataset) to control for discrepancies that could arise with varying prompt wording. For reproducibility, we present the exact prompt wording used for each dataset in Appendix F. We use greedy autoregressive decoding in all experiments, and randomize the order of documents to prevent any systematic bias possible due to location of the "gold documents" (à la Liu et al., 2023a).

4.1.1. NATURALQUESTIONS-OPEN

NaturalQuestions-Open (Liu et al., 2023a) is an open domain question answering benchmark that is derived from Natural Questions (Kwiatkowski et al., 2019). It contains the historical queries issued to the Google search engine, coupled with answers using the contents of English Wikipedia. We follow the same experimental setup as Liu et al., 2023a, including the same preprocessing and evaluation methodology for the 20 document setting (reporting Best EM Subspan, or "Accuracy" for short).

We present speedup vs. accuracy comparisons in Table 1. For the TF-IDF baseline, we use TF-IDF (from SciPy package Virtanen et al., 2020) to select the top-k documents conditioned on the query, then perform "naive LLM-RAG" (as described in Section 3.1). Our BM-25 baseline is equivalent, except we use Brown, 2020 for the top-k document selection. We also have an equivalent baseline where we use Contriever (Izacard et al., 2021) to select the top-k documents.[7] We compare against the recently proposed Attention Sort method, using their method exactly as described in Peysakhovich & Lerer, 2023. Finally, we compare against Prompt Cache (Gim et al., 2023). Note that Naive LLM-RAG, Prompt Cache, and Attention Sort always attend to all documents.

In addition to Table 1, we present various architectural ablation studies in Section 5 and Appendix A to justify our design decisions.

4.1.2. MUSIQUE

MuSiQue (Trivedi et al., 2022) is a multi-hop reasoning dataset consisting of question answer pairs collected with the goal of making disconnected reasoning harder and consequently adding to the difficulty of the previously introduced multi-hop question answering datasets. We validate our approach on the dev split of MuSiQue-Ans (reporting Answer EM and F1).

A slight modification is made to superposition prompting to handle the multi-hop reasoning setup of MuSiQue. Specifically, we iteratively apply superposition pruning to build a chain of $t \times k$ documents[8], where t and k are hyperparameters. At each time step $\{1, \dots t\}$, we create a superposition with all remaining documents, prune to keep the top k, prepend those (cached) documents to the running prefix, then repeat. A visual depiction of this *iterative superposition* is presented in Figure 6. We hypothesize that iterative superposition can improve performance since we equip the LLM to iteratively solve the multi-hop reasoning challenge.

For our baselines, we compare against Attention Sort, Prompt Cache, and Naive LLM-RAG (all of which always attend to all documents). Our results are summarized in Table 2.

4.2. Analysis

4.2.1. SUPERPOSITION PROMPTING CAN IMPROVE TIME EFFICIENCY

Results on the NaturalQuestions-Open dataset Table 1 shows that superposition prompting is the leading efficient method by an order of magnitude. These gains are mainly due to the path parallelism, and path pruning mechanisms. Table 6 presents a breakdown of the contribution of each of these mechanisms to the speedup. For instance, for `mpt-7b-instruct` (on NaturalQuestions-Open), caching alone yields a $10.2\times$ speedup, whereas parallelism alone yields a $14.8\times$ speedup. These optimizations combined with pruning yield a $93.7\times$ speedup overall.

With MuSiQue, we see observe lower overall speedups for

[7]For a more generous representation of the BM-25, TF-IDF, and Contriever baselines, we compute the speedup metrics assuming document KV caching (although to our knowledge, this has not been previously proposed in literature). Note that caching is not possible with Naive LLM-RAG or Attention Sort since the order of documents is variable, and in general, documents attend to other documents (thus also parallelization is not possible).

[8]Note that only the first k documents chosen are cacheable. Subsequent documents are not cacheable since their KV caches depend on preceding documents (which are dynamically chosen during query serving).

Table 1. Retrieval augmented generation accuracy for various models and methods on the NaturalQuestions-Open dataset. For baselines with hyperparameters—namely the top-k parameter for BM-25, TF-IDF, and Contriever—we present their highest accuracy configuration (see Appendix A.1 for all configurations). We emphasize the superiority of superposition prompting over the considered baselines along the axes of both accuracy and speedup.

Model	Approach	Compute Cycles	Theor. Speedup	Accuracy
OpenELM-3B-In.	Naive LLM-RAG	1.03e+13	1.0	0.001
	BM-25	1.07e+11	96.9	0.166
	TF-IDF	1.07e+11	96.9	0.215
	Contriever	5.62e+11	18.4	0.191
	Attention Sort	3.09e+13	0.3	0.000
	Prompt Cache	1.25e+11	82.4	0.000
	Superposition (Ours)	**1.07e+11**	**96.8**	**0.241**
bloomz-3b	Naive LLM-RAG	9.75e+12	1.0	0.005
	BM-25	1.35e+12	7.2	0.138
	TF-IDF	1.35e+12	7.2	0.188
	Contriever	1.37e+12	7.1	0.168
	Attention Sort	2.93e+13	0.3	0.004
	Prompt Cache	1.29e+11	75.4	0.113
	Superposition (Ours)	**9.92e+10**	**98.3**	**0.223**
bloomz-7b1	Naive LLM-RAG	2.22e+13	1.0	0.022
	BM-25	7.35e+12	3.0	0.150
	TF-IDF	3.21e+12	6.9	0.203
	Contriever	3.23e+12	6.9	0.194
	Attention Sort	6.67e+13	0.3	0.022
	Prompt Cache	2.86e+11	77.7	0.136
	Superposition (Ours)	**2.38e+11**	**93.5**	**0.253**
mpt-7b-instruct	Naive LLM-RAG	2.16e+13	1.0	0.026
	BM-25	3.11e+12	7.0	0.278
	TF-IDF	1.18e+12	18.4	0.333
	Contriever	1.20e+12	18.1	0.338
	Attention Sort	6.49e+13	0.3	0.028
	Prompt Cache	2.36e+11	91.8	0.278
	Superposition (Ours)	**2.31e+11**	**93.7**	**0.465**

the highest performing superposition prompting settings (Table 2). This is due to the employment of *iterative superposition* (Section 4.1.2), which limits caching opportunities to the selection of the first k documents.

4.2.2. Superposition Prompting Can Improve Accuracy

In Table 1 we clearly see that superposition prompting is the dominant method in terms of accuracy on NaturalQuestions-Open, seeing improvements of 12–43% over the naive solution, and up to 15% improvements over the next best competitor. With MuSiQue (Table 2), we note that superposition prompting yields the highest accuracy for each model.

One explanation for the accuracy improvement is how superposition prompting reduces sequence lengths as perceived by the transformer. Recent studies have investigated the apparent lack of "length extrapolation" abilities of transformer-based LLMs (Press et al., 2021; Ruoss et al., 2023; Kazemnejad et al., 2023). One convenient property of superposition prompting is that—from the perspective of the transformer—the maximum sequence length observed is the *longest path through the graph*.[9] For example, with NaturalQuestions-Open, superposition prompting decreases the maximum path (and thus the sequence length) from an average of 2923 tokens to 206 tokens. In this sense, superposition prompting for RAG can enable *non*-long-context transformers to perform well on long sequences. This property could allow model developers to significantly reduce pretraining costs (since training special-purpose "long-context" LLMs leads to increased costs (Press et al., 2021)).

Another explanation for the accuracy improvement is the LLM "distraction" phenomenon. The previous works of Liu et al., 2023a; Borgeaud et al., 2021a; Shi et al., 2023 present arguments for how LLMs can be sensitive to noisy or irrelevant context. With the inclusion of the path pruning mechanism, we equip the model with a structured way to filter out the "noise" (*i.e. irrelevant documents*).

[9]This means the effective (perceived) sequence length is $\mathcal{O}(1)$ instead of $\mathcal{O}(n_d)$, where n_d is the number of offline documents.

Table 2. Retrieval augmented generation accuracy for various models on the MuSiQue dataset. For superposition prompting, t denotes the number of iterations of iterative superposition (described in Section 4.1.2), and k denotes the top-k selected (i.e. not pruned) at each step (see Section 3.2.3).

Model	Approach	Compute Cycles	Theor. Speedup	F1	EM
OpenELM-3B-In.	Naive LLM-RAG	8.81E+12	1.0	0.006	0.000
	Attention Sort	2.64E+13	0.3	0.009	0.001
	Prompt Cache	1.61E+11	54.7	0.000	0.000
	Superposition (X=1, K=4) (Ours)	**1.42E+11**	**62.1**	0.028	0.000
	Superposition (X=2, K=4) (Ours)	5.97E+11	14.8	0.039	0.006
	Superposition (X=4, K=1) (Ours)	1.99E+12	4.4	**0.059**	**0.019**
bloomz-3b	Naive LLM-RAG	8.31E+12	1.0	0.060	0.030
	Attention Sort	2.49E+13	0.3	0.055	0.026
	Prompt Cache	1.64E+11	50.5	0.136	0.081
	Superposition (X=1, K=4) (Ours)	**1.33E+11**	**62.4**	0.173	0.100
	Superposition (X=2, K=4) (Ours)	5.56E+11	14.9	**0.187**	**0.117**
	Superposition (X=4, K=1) (Ours)	1.87E+12	4.4	0.187	0.115
bloomz-7b1	Naive LLM-RAG	1.91E+13	1.0	0.062	0.033
	Attention Sort	5.72E+13	0.3	0.058	0.031
	Prompt Cache	3.67E+11	52.0	0.161	0.108
	Superposition (X=1, K=4) (Ours)	**3.17E+11**	**60.2**	0.159	0.090
	Superposition (X=2, K=4) (Ours)	1.33E+12	14.3	0.171	0.106
	Superposition (X=4, K=1) (Ours)	4.44E+12	4.3	**0.182**	**0.115**
mpt-7b-instruct	Naive LLM-RAG	1.86E+13	1.0	0.064	0.008
	Attention Sort	5.57E+13	0.3	0.062	0.009
	Prompt Cache	3.09E+11	60.1	0.088	0.018
	Superposition (X=1, K=4) (Ours)	**3.04E+11**	**61.1**	0.111	0.029
	Superposition (X=2, K=4) (Ours)	1.29E+12	14.4	0.111	0.031
	Superposition (X=4, K=1) (Ours)	4.26E+12	4.4	**0.120**	**0.040**

4.2.3. Sensitivity to ALiBi vs. RoPE

For reasons outlined in Section 3.2.2, superposition prompting is very naturally suited for transformers which accept continuously-valued token position assignments (*i.e.* they exhibit the *position interpolation* property as defined by Chen et al., 2023). While the ALiBi positional encoding scheme has been shown to posses this property, it has been suggested that fine-tuning may be required to equip Rotary Position Embedding (RoPE)-based models with this property.

Our experiments validate that our proposed equilibrium positional assignment mechanism is compatible even with a *non*-fine-tuned RoPE-based model (*i.e.* the OpenELM family). We leave it to future studies to measure the extent to which fine-tuning may improve accuracy (if at all).

We note that `OpenELM-3B-Instruct` has significantly lower accuracy for many baselines, such as AttentionSort, Naive LLM-RAG and even Prompt Cache. We hypothesize that this is due to the lack of length extrapolation capabilities of RoPE, which would become more pronounced for those baselines.

Table 3. Varying the position assignment function used with superposition prompting on the NaturalQuestions-Open dataset.

Model	Approach	Accuracy
OpenELM-3B-In.	Left Aligned	0.224
	Equilibrium (Ours)	**0.241**
bloomz-3b	Left Aligned	0.208
	Equilibrium (Ours)	**0.223**
bloomz-7b1	Left Aligned	0.245
	Equilibrium (Ours)	**0.253**
mpt-7b-instruct	Left Aligned	0.348
	Equilibrium (Ours)	**0.465**

5. Ablations

5.1. Position Assignment Ablation

In Table 3, we investigate the effect of the position assignment strategy during superposition prompting. We compare our proposed equilibrium path positioning to the left aligned strategy described in Section 3.2.2. Our findings validate our hypothesis outlined in Algorithm 1, where we speculated that left alignment would result in worse performance (due to long sequence attention bias).

5.2. Path Saliency Metric Ablation

In Table 4, we ablate our choice of "path saliency" metric. We compare against two other baselines—*attention* and *none*. With *none*, we simply do not prune. The *attention* baseline consists of using the attention scores for each document (average across tokens, layers and attention heads) as the path score. We highlight that our Bayesian path saliency significantly outperforms attention-based scoring, as well as the control baseline.

Table 4. Retrieval augmented generation accuracy for various path saliency metrics on the NaturalQuestions-Open dataset.

Model	Selection Metric	Accuracy
OpenELM-3B-In.	None	0.005
	Attention	0.163
	Bayesian (Ours)	**0.241**
bloomz-3b	None	0.110
	Attention	0.100
	Bayesian (Ours)	**0.223**
bloomz-7b1	None	0.142
	Attention	0.127
	Bayesian (Ours)	**0.253**
mpt-7b-instruct	None	0.224
	Attention	0.218
	Bayesian (Ours)	**0.465**

5.3. Superposition Factor Ablation

We introduce the hyperparameter *superposition factor* as a parameter to interpolate between a fully superimposed and fully "classical" prompt. Larger superposition factors correspond to "more superimposed" prompts, whereas smaller superposition factors correspond to "less superimposed" prompts (achieved by combining adjacent documents before creating prompt paths).

Formally, we define m as the number of documents considered for a retrieval-augmented generation query (for instance, this is $m = 20$ for common settings of NaturalQuestions-Open (Liu et al., 2023a) and MuSiQue (Trivedi et al., 2022)). By setting a superposition factor $\gamma \in [1, m]$, we compute the "effective documents per path" as $\lfloor \frac{m}{\gamma} \rceil$. Importantly, note that when $\gamma = 1$, we've reduced to the "classical" (Naive LLM-RAG) case. We perform an ablation by sweeping this superposition factor parameter and present results in Figure 5. A visual representation is presented in Figure 7.

The curves generally show improvements along both axes as we increase the superposition quotient. Interestingly, the maximal accuracy may not be fully superimposed, suggesting that this value should be tuned for the given application.

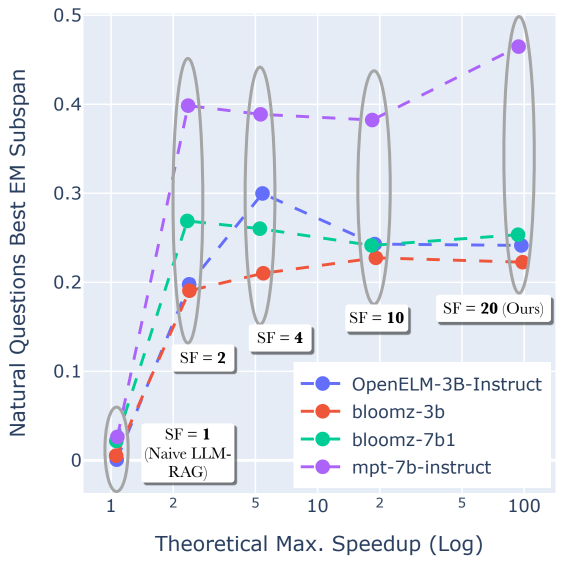


Figure 5. Sweeping values of superposition factor (SF) on the NaturalQuestions-Open dataset with a variety of models.

6. Conclusion and Discussion

In this work, we introduced a novel framework to accelerate and improve retrieval-augmented generation with LLMs. We verified the generalization of our method across various models and datasets and performed extensive ablations.

Our method, *superposition prompting*, was shown to improve long sequence modeling accuracy in single and multi-hop question answering tasks, all while reducing user-observed response latency. Conveniently, this optimization was shown to be effective without any fine-tuning or additional training to the base model. We defer to future work to explore how (if at all) fine-tuning could *further* improve superposition prompting. We also highlight that future work should investigate how to generalize these ideas outside of the RAG setting.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. Specifically, we are proposing a machine learning technique that can be used for improving the quality of generative language modeling while reducing the computational overheads often associated with their deployment. There are many potential consequences of this work, as for the field as a whole, none which we feel must be specifically highlighted here.

Acknowledgements

We would like to extend a thanks to Sachin Mehta, Maxwell Horton, Enrico Fini, and Arsalan Farooq for discussion and feedback on the paper.

References

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *ArXiv*, abs/2310.11511, 2023. URL https://api.semanticscholar.org/CorpusID:264288947.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., and Hoefler, T. Graph of thoughts: Solving elaborate problems with large language models. *ArXiv*, abs/2308.09687, 2023. URL https://api.semanticscholar.org/CorpusID:261030303.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J.-B., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T. W., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J. W., Elsen, E., and Sifre, L. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2021a. URL https://api.semanticscholar.org/CorpusID:244954723.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J.-B., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T. W., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J. W., Elsen, E., and Sifre, L. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2021b. URL https://api.semanticscholar.org/CorpusID:244954723.

Brown, D. Rank-BM25: A Collection of BM25 Algorithms in Python, 2020. URL https://doi.org/10.5281/zenodo.4520057.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

Cai, T., Huang, K., Lee, J., and Wang, M. Scaling in-context demonstrations with structured attention. *ArXiv*, abs/2307.02690, 2023. URL https://api.semanticscholar.org/CorpusID:259360659.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *ArXiv*, abs/2306.15595, 2023. URL https://api.semanticscholar.org/CorpusID:259262376.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and R'e, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *ArXiv*, abs/2205.14135, 2022. URL https://api.semanticscholar.org/CorpusID:249151871.

facebookresearch. fvcore. https://github.com/facebookresearch/fvcore, 2024.

Feynman, R. P. *Quantum Mechanics and Path Integrals*. McGraw-Hill, 1965.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., and Wang, H. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997, 2023. URL https://api.semanticscholar.org/CorpusID:266359151.

Gim, I., Chen, G., seob Lee, S., Sarda, N., Khandelwal, A., and Zhong, L. Prompt cache: Modular attention reuse for low-latency inference. *ArXiv*, abs/2311.04934, 2023. URL https://api.semanticscholar.org/CorpusID:265067391.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-w. Realm: Retrieval-augmented language model pre. *Training*, 2020.

Huang, Y., Xu, J., Jiang, Z., Lai, J., Li, Z., Yao, Y., Chen, T., Yang, L., Xin, Z., and Ma, X. Advancing transformer architecture in long-context large language models: A comprehensive survey. *ArXiv*, abs/2311.12351, 2023. URL https://api.semanticscholar.org/CorpusID:265308945.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022,

2021. URL https://api.semanticscholar.org/CorpusID:249097975.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers. *ArXiv*, abs/2305.19466, 2023. URL https://api.semanticscholar.org/CorpusID:258987259.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. URL https://api.semanticscholar.org/CorpusID:261697361.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401, 2020. URL https://api.semanticscholar.org/CorpusID:218869575.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *ArXiv*, abs/2306.00978, 2023. URL https://api.semanticscholar.org/CorpusID:258999941.

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

Liu, X., Wang, J., Sun, J., Yuan, X., Dong, G., Di, P., Wang, W., and Wang, D. Prompting frameworks for large language models: A survey. *ArXiv*, abs/2311.12785, 2023b. URL https://api.semanticscholar.org/CorpusID:265308881.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *ArXiv*, abs/2305.17118, 2023c. URL https://api.semanticscholar.org/CorpusID:258947558.

Mehta, S., Sekhavat, M. H., Cao, Q., Horton, M., Jin, Y., Sun, C., Mirzadeh, I., Najibi, M., Belenko, D., Zatloukal, P., and Rastegari, M. Openelm: An efficient language model family with open training and inference framework. *ArXiv*, abs/2404.14619, 2024. URL https://api.semanticscholar.org/CorpusID:269302585.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Jin, H., Chen, T., and Jia, Z. Towards efficient generative large language model serving: A survey from algorithms to systems. *ArXiv*, abs/2312.15234, 2023. URL https://api.semanticscholar.org/CorpusID:266551872.

MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL www.mosaicml.com/blog/mpt-7b. Accessed: 2023-05-05.

Muennighoff, N. Sgpt: Gpt sentence embeddings for semantic search. *ArXiv*, abs/2202.08904, 2022. URL https://api.semanticscholar.org/CorpusID:246996947.

Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Scao, T. L., Bari, M. S., Shen, S., Yong, Z.-X., Schoelkopf, H., Tang, X., Radev, D. R., Aji, A. F., Almubarak, K., Albanie, S., Alyafeai, Z., Webson, A., Raff, E., and Raffel, C. Crosslingual generalization through multitask finetuning. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:253264914.

Ning, X., Lin, Z., Zhou, Z., Yang, H., and Wang, Y. Skeleton-of-thought: Large language models can do parallel decoding. *ArXiv*, abs/2307.15337, 2023. URL https://api.semanticscholar.org/CorpusID:260315904.

Petroni, F., Piktus, A., Fan, A., Lewis, P., Yazdani, M., De Cao, N., Thorne, J., Jernite, Y., Karpukhin, V., Maillard, J., et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

Peysakhovich, A. and Lerer, A. Attention sorting combats recency bias in long context language models. *ArXiv*, abs/2310.01427, 2023. URL https://api.semanticscholar.org/CorpusID:263609111.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *CoRR*, abs/2108.12409, 2021. URL https://arxiv.org/abs/2108.12409.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Magar, I., Abend, O., Karpas, E. D., Shashua, A., Leyton-Brown, K., and Shoham, Y. Parallel context windows for large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://api.semanticscholar.org/CorpusID:258686160.

Ruoss, A., Del'etang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M. A., Legg, S., and Veness, J. Randomized positional encodings boost length generalization of transformers. *ArXiv*, abs/2305.16843, 2023. URL https://api.semanticscholar.org/CorpusID:258947457.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Fu, D. Y., Xie, Z., Chen, B., Barrett, C. W., Gonzalez, J., Liang, P., Ré, C., Stoica, I., and Zhang, C. High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:257495837.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., hsin Chi, E. H., Scharli, N., and Zhou, D. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:256459776.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021. URL https://api.semanticscholar.org/CorpusID:233307138.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Xiao, G., Lin, J., Seznec, M., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. *ArXiv*, abs/2211.10438, 2022. URL https://api.semanticscholar.org/CorpusID:253708271.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601, 2023. URL https://api.semanticscholar.org/CorpusID:258762525.

Ye, Q., Beltagy, I., Peters, M. E., Ren, X., and Hajishirzi, H. Fid-icl: A fusion-in-decoder approach for efficient in-context learning. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:259370780.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhang, Z. A., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C. W., Wang, Z., and Chen, B. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *ArXiv*, abs/2306.14048, 2023. URL https://api.semanticscholar.org/CorpusID:259263947.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J., and rong Wen, J. A survey of large language models. *ArXiv*, abs/2303.18223, 2023. URL https://api.semanticscholar.org/CorpusID:257900969.

A. Ablations (cont.)

A.1. Top-k Ablation

Here, we sweep values for top-k for our method, where k are the number of documents retained for generating the answer (full table results are provided in Table 5). For completeness, we also ablate this hyperparameter for all "ranking-based" baselines, namely BM-25, TF-IDF, and Contriever (described in Section 4.1.1). Note that for superposition prompting, this top-k value corresponds the number of paths retained after path pruning. We see that accuracy tends to peak around $k = 2$ to $k = 4$, although it comes at a cost to speedup versus $k = 1$. Interestingly we do see that performance decreases steadily for increasing $k > 4$. This seems to coincide with the arguments put forth in (Shi et al., 2023) and Press et al., 2021 and Liu et al., 2023a of "increased distraction" as the context length increases.

Table 5. Retrieval augmented generation accuracy and speedups for various models on the NaturalQuestions-Open dataset, sweeping top-k. Note that "Comp." is used as an abbreviation for "Compute", and "Sp." is used as an abbreviation for "Speedup".

Model	Approach	Top-k	Comp. Cycles	Theor. Sp.	CUDA Sp.	Accuracy
bloomz-3b	BM-25	8	3.13E+12	3.1	2.40	0.129
		4	1.35E+12	7.2	3.83	0.138
		2	5.09E+11	19.1	5.53	0.106
		1	9.92E+10	98.2	6.51	0.092
	Contriever	8	3.15E+12	3.1	2.26	0.162
		4	1.37E+12	7.1	3.51	0.168
		2	5.25E+11	18.6	4.87	0.153
		1	1.15E+11	84.5	5.61	0.114
	TF-IDF	8	3.13E+12	3.1	2.40	0.164
		4	1.35E+12	7.2	3.81	0.188
		2	5.09E+11	19.1	5.48	0.187
		1	9.92E+10	98.2	6.46	0.155
	Superposition (Ours)	8	4.06E+11	24.0	5.80	0.244
		4	2.30E+11	42.4	5.92	**0.264**
		2	1.43E+11	68.3	5.96	0.252
		1	9.92E+10	98.2	5.64	0.223
bloomz-7b1	BM-25	8	7.35E+12	3.0	2.33	0.150
		4	3.21E+12	6.9	3.99	0.150
		2	1.22E+12	18.2	5.40	0.128
		1	2.38E+11	93.4	8.02	0.098
	Contriever	8	7.37E+12	3.0	2.26	0.185
		4	3.23E+12	6.9	3.81	0.194
		2	1.23E+12	18.0	5.04	0.172
		1	2.54E+11	87.5	7.32	0.125
	TF-IDF	8	7.35E+12	3.0	2.32	0.188
		4	3.21E+12	6.9	4.00	0.203
		2	1.22E+12	18.2	5.39	0.202
		1	2.38E+11	93.4	8.01	0.159
	Superposition (Ours)	8	9.66E+11	23.0	6.52	0.271
		4	5.49E+11	40.5	6.58	**0.301**
		2	3.42E+11	65.0	6.59	0.288
		1	2.38E+11	93.4	6.61	0.253
mpt-7b-instruct	BM-25	8	7.11E+12	3.0	2.29	0.268
		4	3.11E+12	7.0	4.01	0.278
		2	1.18E+12	18.4	5.38	0.269
		1	2.31E+11	94.0	8.12	0.240
	Contriever	8	7.13E+12	3.0	2.22	0.318
		4	3.13E+12	6.9	3.80	0.334
		2	1.20E+12	18.1	5.00	0.338
		1	2.47E+11	87.8	7.28	0.287
	TF-IDF	8	7.11E+12	3.0	2.28	0.297
		4	3.11E+12	7.0	3.99	0.332
		2	1.18E+12	18.4	5.34	0.333
		1	2.31E+11	94.0	8.07	0.308
	Superposition (Ours)	8	9.27E+11	23.4	5.78	0.423
		4	5.28E+11	41.0	6.14	0.456
		2	3.30E+11	65.7	6.32	**0.471**
		1	2.31E+11	94.0	6.46	0.465

A.2. Runtime Optimization Ablation

In Table 6 we present a full breakdown of the incremental effect of the path pruning, path caching, and path parallelism optimizations proposed in Section 3.2.3, Section 3.3.1, and Section 3.3.2, respectively.

Table 6. Ablation of speedup vs. accuracy on the NaturalQuestions-Open dataset by enabling/disabling the optimizations proposed in Section 3.3. Each of Pruning?/Caching?/Parallelism? correspond to path pruning, path caching, and path parallelism enabled.

Model	Pruning?	Caching?	Parallelism?	Compute Cycles	Theor. Speedup	Accuracy
OpenELM-3B-In.	False	False	False	9.81E+12	1.1	0.005
			True	6.82E+11	15.1	0.005
		True	False	9.86E+11	10.5	0.005
			True	1.18E+11	87.2	0.005
	True	False	False	9.80E+12	1.1	0.241
			True	6.71E+11	15.4	0.241
		True	False	9.74E+11	10.6	0.241
			True	1.07E+11	96.9	0.241
bloomz-3b	False	False	False	9.08E+12	1.1	0.114
			True	6.40E+11	15.2	0.114
		True	False	9.25E+11	10.5	0.114
			True	1.18E+11	82.4	0.114
	True	False	False	9.06E+12	1.1	0.223
			True	6.21E+11	15.7	0.223
		True	False	9.06E+11	10.8	0.223
			True	9.92E+10	98.2	0.223
bloomz-7b1	False	False	False	2.18E+13	1.0	0.125
			True	1.52E+12	14.6	0.125
		True	False	2.20E+12	10.1	0.125
			True	2.68E+11	82.8	0.125
	True	False	False	2.18E+13	1.0	0.253
			True	1.49E+12	14.9	0.253
		True	False	2.17E+12	10.2	0.253
			True	2.38E+11	93.4	0.253
mpt-7b-instruct	False	False	False	2.13E+13	1.0	0.287
			True	1.46E+12	14.8	0.287
		True	False	2.11E+12	10.3	0.287
			True	2.34E+11	92.7	0.287
	True	False	False	2.13E+13	1.0	0.465
			True	1.46E+12	14.8	0.465
		True	False	2.11E+12	10.3	0.465
			True	2.31E+11	94.0	0.465

B. Supplementary Algorithm Details

B.1. Bayesian Path Selection

Define $H : (\mathbb{R}^{* \times n_v}, \mathcal{S}^*) \to \mathbb{R}$ as the language modeling cross-entropy. [10] Formally, for our *ForkJoin* prompt topology, we compute the Bayesian path saliency as follows.

As detailed in Appendix B.3, during superposition prompting inference, we compute logits for the preamble $\pi \in \mathbb{R}^{\|\boldsymbol{p}\| \times n_v}$ (where n_v is the vocabulary size), logits for all documents $\delta_i \in \mathbb{R}^{\|\boldsymbol{d}_i\| \times n_v}$, and logits for all queries $\phi_i \in \mathbb{R}^{\|\boldsymbol{q}\| \times n_v}$ for $i \in D$. Then, for each $k \in D$, we can compute:

[10]More specifically, this is the "shifted" cross entropy between a logit tensor and sequence tensor of the same sequence dimension (where we discard element 1 from the input sequence and the last element from the logit tensor).

$$\log P(\boldsymbol{d}_i \mid \boldsymbol{q}_i, \boldsymbol{p}) = \log P(\boldsymbol{q}_i \mid \boldsymbol{d}_i, \boldsymbol{p}) + \log P(\boldsymbol{d}_i \mid \boldsymbol{q}) - C \tag{2a}$$

$$= \frac{\log H(\delta_i, \boldsymbol{d}_i)}{\|\boldsymbol{d}_i\|} + \frac{\log H(\pi, \boldsymbol{p})}{\|\boldsymbol{p}\|} - C \tag{2b}$$

Equation (2a) follows from Bayes Rule (where C is some unspecified constant), while Equation (2b) follows from our definition of language model. The C term is inconsequential for our use, since we eventually softmax before comparing the likelihoods. This justifies why we duplicate queries in the *ForkJoin* topology—without duplicating the query for each path, we only have access to $P(\boldsymbol{q}_i \mid \boldsymbol{d}_1, \ldots \boldsymbol{d}_{n_d}, \boldsymbol{p})$, not the independent terms $\{P(\boldsymbol{q}_i \mid \boldsymbol{d}_i, \boldsymbol{p}) \mid i \in D\}$ We choose to normalize the log likelihood terms by their corresponding sequence lengths ($\|\boldsymbol{d}_i\|$ and $\|\boldsymbol{q}\|$) to effectively achieve a "likelihood density per token," which prevents bias against shorter sequence lengths[11].

B.2. Equilibrium Position Assignment

Section 3.2.2 outlined the intuition behind the equilibrium position assignment algorithm. Here, we algorithmically describe the method.

For convenience, we define the **ArangePositions** function, a_Δ, as

$$a_\Delta(s_0, m) = \langle s_1, s_1 + \Delta, ..., s_1 + (n-1)\Delta \rangle \in \mathcal{P}^m \tag{3}$$

where $\Delta \in \mathbb{R}$ is the step size, $s_0 \in \mathbb{R}$ (the "starting" position) and $m \in \mathbb{Z}^+$ (the extend amount).

Algorithm 1 Equilibrium Positioning of the "*Fork*" portion of the *ForkJoin* Topology

```
Input: Preamble token sequence $\boldsymbol{p}$, set of document sequences $\boldsymbol{D} = \{\boldsymbol{d}_i \mid i \in D\}$
$\check{p} = a_1(0, \|\boldsymbol{p}\|)$ ▷ Using Equation (3)
for $i = 1$ to $n_d$ do
    $s_i = S(\boldsymbol{D})/\|\boldsymbol{d}_i\|$ ▷ Using Equation (1)
    $\check{d}_i = a_{s_i}(\max(\check{p}) + 1, \|\boldsymbol{d}_i\|)$
end for
Output: Preamble token positions $\check{p}$, set of document token positions $\{\check{d}_i \mid i \in D\}$
```

B.3. Full Algorithm Outline

We denote $D = \{1, \ldots, n_d\}$ as our document "indices." Note that we stylize KV cache variables as boxed variables for visual distinctness (for instance $\boxed{a}$). We define $\boxed{\emptyset}$ as an "empty" KV cache (i.e. sequence length of 0). We also define $\mathrm{LM} : (\boxed{x}, \boldsymbol{y}, \check{y}) \mapsto (\boxed{y}, \psi)$ where:

- $\boxed{x} \in \mathcal{M}$ (KV cache used as input)
- $\boldsymbol{y} \in \mathcal{S}$ (new tokens not included in KV cache)
- $\boxed{y} \in \mathcal{M}$ and $\|\boxed{y}\| = \|\boldsymbol{y}\|$ (KV cache computed by LLM)
- $\psi \in \mathbb{R}^{\|\boldsymbol{y}\| \times n_v}$ (logit predictions computed by LLM)

Following the insight of Section 3.3.2, we also define

$$\mathrm{LMP} : (\{\boxed{x}_i \mid i \in X\}, \boldsymbol{y}, \check{y}) \mapsto (\{\boxed{y}_i \mid i \in X\}, \{\psi_i \mid i \in X\})$$

with analogous outputs to LM, although "batched" (note how the call accepts a collection of input KV caches, and outputs a collection of KV caches and a collection of logits). A visual depiction can be seen in Figure 3e.

With these definitions, we present the full formalized preprocessing algorithm Algorithm 2 and online serving Algorithm 3.

[11]Note that SGPT (Muennighoff, 2022) avoided length bias by truncating sequences to a common length. We choose not to follow this design decision to prevent potential loss of vital information in the input data.

Algorithm 2 Offline Preprocessing

```
**Input:** Preamble token sequence $\boldsymbol{p}$
**Input:** Set of document sequences $\boldsymbol{D} = \{\boldsymbol{d}_i \mid i \in D\}$
$\check{p} \oplus \bigoplus_{i \in D} \check{d}_i$ := **Equilibrium**$(\boldsymbol{p}, \boldsymbol{D})$ ▷ *Algorithm 1*
$(\boxed{p}, \pi) := \mathrm{LM}(\boxed{\emptyset}, \boldsymbol{p}, \check{p})$
**for** $i = 1$ **to** $n_d$ **do**
    $(\boxed{d}_i, \delta_i) := \mathrm{LM}(\boxed{p}, \boldsymbol{d}_i, \check{d}_i)$
**end for**
**Output:** Preamble positions $\check{p}$, KV cache $\boxed{p}$, logits $\pi$
**Output:** Set of document KV positions $\{\check{d}_i \mid i \in D\}$, KV caches $\{\boxed{d}_i \mid i \in D\}$ and logits $\{\delta_i \mid i \in D\}$
```

Algorithm 3 Online Serving

```
**Input:** Preamble positions $\check{p}$, KV cache $\boxed{p}$, logits $\pi$
**Input:** Set of document KV positions $\{\check{d}_i \mid i \in D\}$, KV caches $\{\boxed{d}_i \mid i \in D\}$ and logits $\{\delta_i \mid i \in D\}$
**Input:** Query tokens $\boldsymbol{q}$
**Input:** Postamble tokens $\boldsymbol{t}$ ▷ *For instance, "\n### Response\n" if using Alpaca instruct tuning format.*
$\check{q} := a_1(\max(\check{d}_1), \|\boldsymbol{q}\|)$ ▷ *Using Equation (3).*
$\boldsymbol{B} := \{\boxed{p} \oplus \boxed{d}_i \mid i \in D\}$
$\{(\boxed{q}_i, \phi_i) \mid i \in D\} := \mathrm{LMP}(\boldsymbol{B}, \boldsymbol{q}, \check{q})$
▷ *Could use tuned threshold instead of top-k.*
$K := \mathrm{argmin}^K_{i \in D} P(\boldsymbol{d}_i \mid \boldsymbol{q}_i, \boldsymbol{p})$ ▷ *Using Equation (2).*
$\boxed{pdqt} := \boxed{p} \oplus \bigoplus_{k \in K} (\boxed{d}_a \oplus \boxed{q}_a) \oplus \boxed{t}$
$\boxed{t}, \tau := \mathrm{LM}(\boxed{pdqt}, \boldsymbol{t}, \check{t})$
$\lambda := \tau$
▷ *Greedy Decoding shown here, but WLOG another decoding scheme could be used.*
**repeat**
    $\boldsymbol{l}$ := **Sample**$(\lambda) \in \mathcal{S}$ ▷ *Some arbitrary sampling procedure.*
    $(\boxed{l}, \lambda) := \mathrm{LM}(\boxed{pdqt} \oplus \boxed{r}, \boldsymbol{l}, \check{l})$
    $\boxed{r} := \boxed{r} \oplus \boxed{l}$
    $\boldsymbol{r} := \boldsymbol{r} \oplus \boldsymbol{l}$
    $e$ := **Stop?**$(\boldsymbol{r}) \in \{0, 1\}$ ▷ *Say, 1 if EOS-terminated.*
**until** $e = 1$
**Output:** Response token sequence $\boldsymbol{r}$
```

C. CUDA Benchmarks

In Table 5 and Table 7, we present measurements of the compared methods in a realistic server deployment scenario (an NVIDIA A100 80GB). Our CUDA implementation is written in pure PyTorch, and we report the median timing over 30 trials for each method (generating a 5 token response to the prompt). Mirroring our theoretical speedup projections, we report speedups over the Naive LLM-RAG method.

We notice that the actual measured speedups are an order of magnitude smaller than the theoretical maximum speedups calculated. This is expected—as we heavily optimize the FLOPs (up to $100\times$), the memory bottlenecks begin to dominate the runtime. We expect that a fused CUDA kernel implementation could bridge the order of magnitude gap, similar to how Dao et al., 2022 achieves an order of magnitude improvement over the naive PyTorch implementation by mitigating memory transfer bottlenecks.

D. Additional Visualizations

D.1. Iterative Superposition

We present a visual depiction of iterative superposition (from Section 4.1.2) in Figure 6.

Table 7. CUDA speedup measurements for remaining baselines methods not enumerated in Table 5 (this table is meant to be directly comparable to Table 5).

MODEL	APPROACH	CUDA SPEEDUP	ACCURACY
BLOOMZ-3B	NAIVE LLM-RAG	1.000	0.005
	ATTENTION SORT	0.337	0.004
	PROMPT CACHE	6.317	0.113
BLOOMZ-7B1	NAIVE LLM-RAG	1.000	0.022
	ATTENTION SORT	0.335	0.022
	PROMPT CACHE	8.643	0.136
MPT-7B-INSTRUCT	NAIVE LLM-RAG	1.000	0.026
	ATTENTION SORT	0.332	0.028
	PROMPT CACHE	8.113	0.278

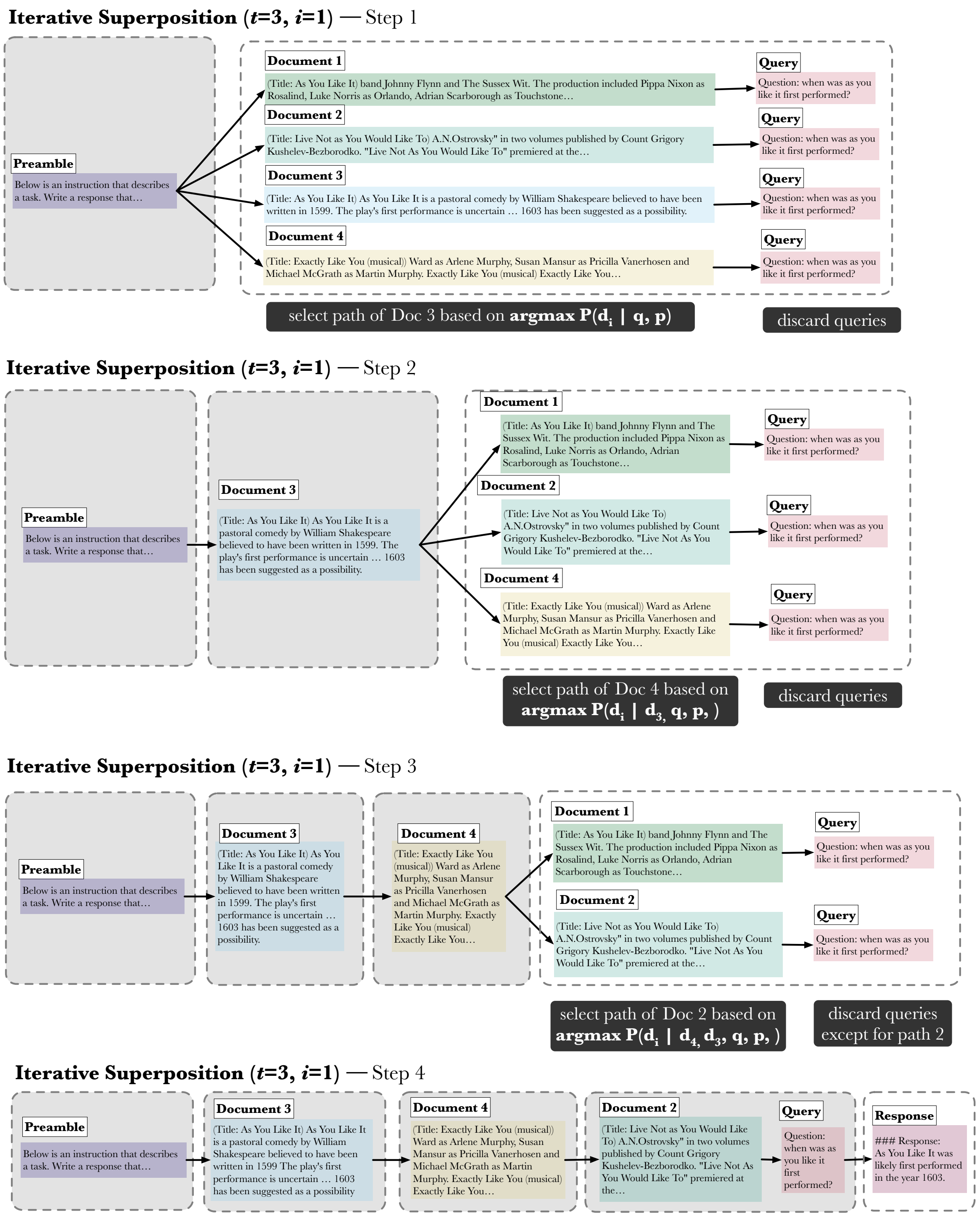
D.2. Superposition Factor

We present a visual depiction of superposition factor (from Section 5.3) in Figure 7.

E. Dataset Statistics

To get a better sense of the effect of the position encoding schemes, we present token count statistics for the document lengths of each evaluation dataset (using a BPE-based tokenizer). Define n_d to be the number of offline documents per example, and define M to be the overall number of examples. Use $\boldsymbol{d}_i^j$ to correspond to the ith document within the jth example.

- Average document length ($\frac{1}{Mn_d}\sum_i\sum_j \|\boldsymbol{d}_i^j\|$)
  - Natural Questions: 142.6
  - MuSiQue: 121.2
- Average length of longest document ($\frac{1}{M}\sum_j \max_i \|\boldsymbol{d}_i^j\|$)
  - Natural Questions: 170.2
  - MuSiQue: 222.8
- Average Left Align token padding gap size ($\frac{1}{Mn_d}\sum_j\sum_i(\max_k \|\boldsymbol{d}_k^j\| - \|\boldsymbol{d}_i^j\|)$)
  - Natural Questions: 27.6
  - MuSiQue: 149.6


Iterative Superposition (*t*=3, *i*=1) — Step 1
Preamble
Document 1
Document 2
Document 3
Document 4
Query
select path of Doc 3 based on **argmax P(d_i | q, p)**
discard queries
Iterative Superposition (*t*=3, *i*=1) — Step 2
select path of Doc 4 based on **argmax P(d_i | d_3, q, p,)**
discard queries
Iterative Superposition (*t*=3, *i*=1) — Step 3
select path of Doc 2 based on **argmax P(d_i | d_4, d_3, q, p,)**
discard queries except for path 2
Iterative Superposition (*t*=3, *i*=1) — Step 4
Response

Below is an instruction that describes a task. Write a response that…

(Title: As You Like It) band Johnny Flynn and The Sussex Wit. The production included Pippa Nixon as Rosalind, Luke Norris as Orlando, Adrian Scarborough as Touchstone…

(Title: Live Not as You Would Like To) A.N.Ostrovsky" in two volumes published by Count Grigory Kushelev-Bezborodko. "Live Not As You Would Like To" premiered at the…

(Title: As You Like It) As You Like It is a pastoral comedy by William Shakespeare believed to have been written in 1599. The play's first performance is uncertain … 1603 has been suggested as a possibility.

(Title: Exactly Like You (musical)) Ward as Arlene Murphy, Susan Mansur as Pricilla Vanerhosen and Michael McGrath as Martin Murphy. Exactly Like You (musical) Exactly Like You…

Question: when was as you like it first performed?

Response:
As You Like It was likely first performed in the year 1603.

Figure 6. Iterative superposition illustrated example with 3 iterations. Steps 1-3 are the "superposition" steps, where we evaluate multiple documents. Step 4 is where we generate the response from the selected (*i.e.* unpruned) paths.

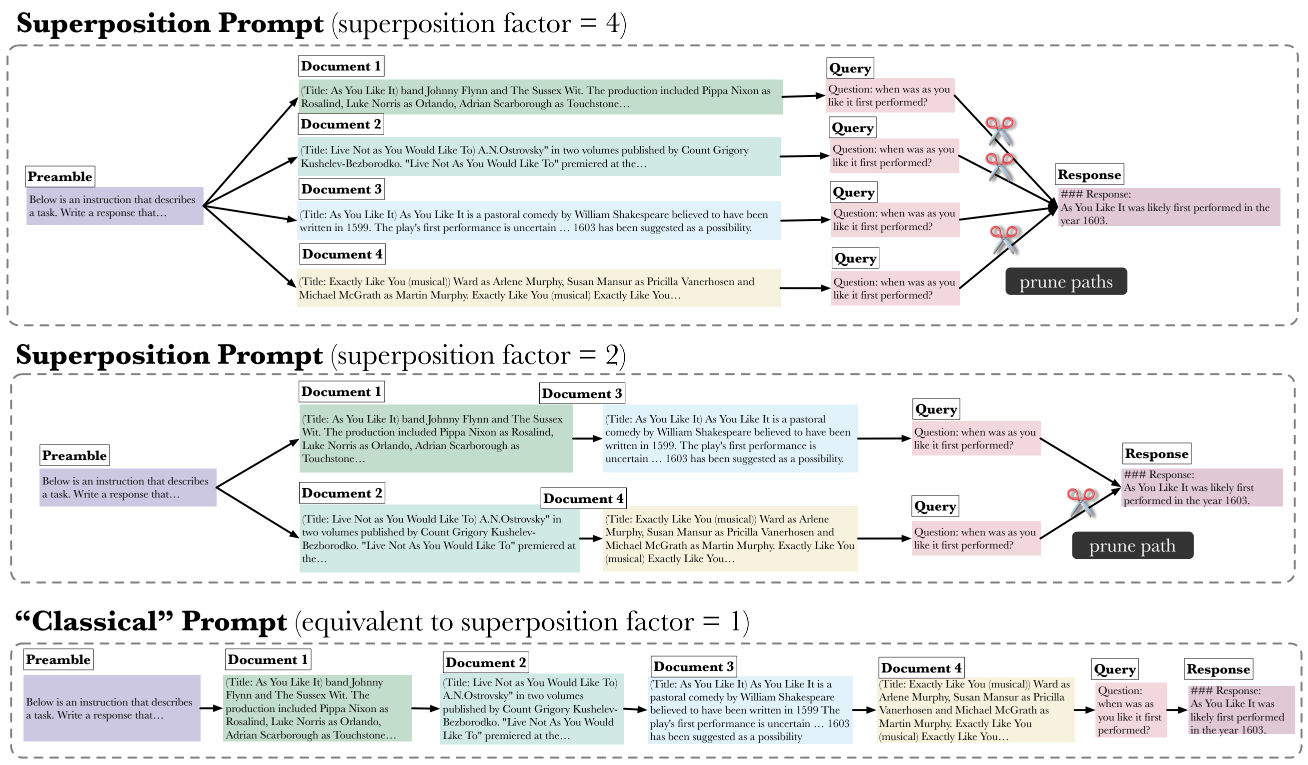

Superposition Prompt (superposition factor = 4)
Preamble
Below is an instruction that describes a task. Write a response that…
Document 1
(Title: As You Like It) band Johnny Flynn and The Sussex Wit. The production included Pippa Nixon as Rosalind, Luke Norris as Orlando, Adrian Scarborough as Touchstone…
Document 2
(Title: Live Not as You Would Like To) A.N.Ostrovsky" in two volumes published by Count Grigory Kushelev-Bezborodko. "Live Not As You Would Like To" premiered at the…
Document 3
(Title: As You Like It) As You Like It is a pastoral comedy by William Shakespeare believed to have been written in 1599. The play's first performance is uncertain … 1603 has been suggested as a possibility.
Document 4
(Title: Exactly Like You (musical)) Ward as Arlene Murphy, Susan Mansur as Pricilla Vanerhosen and Michael McGrath as Martin Murphy. Exactly Like You (musical) Exactly Like You…
Query
Question: when was as you like it first performed?
Response
### Response:
As You Like It was likely first performed in the year 1603.
prune paths
Superposition Prompt (superposition factor = 2)
Preamble
Below is an instruction that describes a task. Write a response that…
Document 1
(Title: As You Like It) band Johnny Flynn and The Sussex Wit. The production included Pippa Nixon as Rosalind, Luke Norris as Orlando, Adrian Scarborough as Touchstone…
Document 3
(Title: As You Like It) As You Like It is a pastoral comedy by William Shakespeare believed to have been written in 1599. The play's first performance is uncertain … 1603 has been suggested as a possibility.
Document 2
(Title: Live Not as You Would Like To) A.N.Ostrovsky" in two volumes published by Count Grigory Kushelev-Bezborodko. "Live Not As You Would Like To" premiered at the…
Document 4
(Title: Exactly Like You (musical)) Ward as Arlene Murphy, Susan Mansur as Pricilla Vanerhosen and Michael McGrath as Martin Murphy. Exactly Like You (musical) Exactly Like You…
Query
Question: when was as you like it first performed?
Response
### Response:
As You Like It was likely first performed in the year 1603.
prune path
"Classical" Prompt (equivalent to superposition factor = 1)
Preamble
Below is an instruction that describes a task. Write a response that…
Document 1
(Title: As You Like It) band Johnny Flynn and The Sussex Wit. The production included Pippa Nixon as Rosalind, Luke Norris as Orlando, Adrian Scarborough as Touchstone…
Document 2
(Title: Live Not as You Would Like To) A.N.Ostrovsky" in two volumes published by Count Grigory Kushelev-Bezborodko. "Live Not As You Would Like To" premiered at the…
Document 3
(Title: As You Like It) As You Like It is a pastoral comedy by William Shakespeare believed to have been written in 1599 The play's first performance is uncertain … 1603 has been suggested as a possibility
Document 4
(Title: Exactly Like You (musical)) Ward as Arlene Murphy, Susan Mansur as Pricilla Vanerhosen and Michael McGrath as Martin Murphy. Exactly Like You (musical) Exactly Like You…
Query
Question: when was as you like it first performed?
Response
### Response:
As You Like It was likely first performed in the year 1603.

Figure 7. Visual depiction of the superposition factor, which is measure of how "superpositioned" a prompt is. A rigorous definition is provided in Section 5.3.

F. Prompt Example on NaturalQuestions-Open

As LLMs are sensitive to the specific wording of prompts, we present a prompt example in full for reproducibility.

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:

Write a high-quality answer for the given question using only the following relevant search results.

[Document](Title: The Crossing (play)) The Crossing (play) The Crossing is a 2006 South African one-man play by Jonathan Nkala.[...]

[Document](Title: The Crossing (TV series)) The Crossing is an American science fiction thriller series that airs on ABC and CTV. The series debuted on April 2, 2018. On March 20, 2018, ABC released the pilot episode on their website. The series is filmed in British Columbia, Canada.

[Document](Title: Crossing South) Crossing South Crossing South is a travel show, television production that was created[...]

[Document](Title: Crossing (2007 film)) He received a Gemini nomination for his work on the show. Crossing (2007 film) Crossing is a 2007[...]

[Document](Title: The Crossing (TV series)) British Columbia and in New Westminster. The first camp footage was filmed at Camp McLean. Filming in Vancouver[...]

[Document](Title: Crossing East) honored by winning a Peabody Award. Crossing East Crossing East is an American documentary series for public radio produced by Dmae Roberts[...]

[Document](Title: Crossings (TV series)) Crossings (TV series) Crossings is a Malaysia dark comedy television drama that consisted of 13 episodes. Bob works as a copywriter[...]

[Document](Title: The Mexican Dream) of the border crossing action takes place was not that difficult. The bar and carwash were probably[...]

[Document](Title: Southern Crossing (film)) it was filmed was ”so wonderful” they had to demolish it in references to the theater’s heritage factor.[...]

[Document](Title: Crossing East) Crossing East Crossing East is an American documentary series for public radio produced by Dmae Roberts and MediaRites and hosted by George Takei and Margaret Cho. Covering Asian immigration to the[...]

[Document](Title: Crossing Lines) series, having previously produced miniseries, as well as its first project since being acquired by StudioCanal in 2012.[...]

[Document](Title: The Crossing (TV series)) a threat. Set in the fictional town of Port Canaan, Oregon and in Seattle, the series was filmed in coastal areas of British Columbia[...]

.

.

.

[Document](Title: The Crossing Hero) Fridays, at 12nn. Beginning 5 April 2015, ”The Crossing Hero” airs on Taiwan’s China Television (CTV),[...]

Question: where did they film the show the crossing?

Response:

<model continues from here...>

G. Prompt Example on MuSiQue

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
You are a question-answering assistant, who is careful to reference source material. Use the source(s) below to answer the user question.

[Document](Title: National Workers Memorial (Australia)) The National Workers Memorial in the national capital, Canberra, Australian Capital[. . .]
[Document](Title: Braddon, Australian Capital Territory) Braddon (postcode: 2612) is an inner north suburb of Canberra, Australian Capital[. . .]
[Document](Title: WKDM) WKDM 1380 is a United States ethnic brokered radio station licensed to New York City. The station is owned by Multicultural Broadcasting and airs programming in Mandarin Chinese, 24 hours a day from Monday[. . .]
[Document](Title: York, Upper Canada) The Town of York was the second capital of the district of Upper Canada and the predecessor to Toronto (1834). It was established in 1793 by Lieutenant - Governor John Graves Simcoe as a "temporary" location for the capital of Upper Canada, while he made plans to build a capital near today's[. . .]
[Document](Title: KLIF-FM) KLIF-FM (93.3 FM, branded as "Hot 93.3") is a radio station licensed to serve Haltom City, Texas, United States. The station is owned by Cumulus Media, and the broadcast license is held by Radio License[. . .]
[Document](Title: WRGV) WRGV (107.3 FM) is a radio station licensed to serve the community of Pensacola, Florida, United States. The station is currently owned by iHeartMedia, Inc. and the broadcast license is held by Clear Channel Broadcasting Licenses, Inc. WRGV broadcasts an urban contemporary music format to the greater[. . .]
[Document](Title: WWRU) WWRU is a Korean language AM radio station licensed to Jersey City, New Jersey, broadcasting to the New York[. . .]
[Document](Title: KDBS) KDBS (1410 AM, ESPN Alexandria) is an American radio station broadcasting a sports talk format. The station is licensed by the Federal Communications Commission (FCC) to serve the community of Alexandria, Louisiana. The[. . .]
[Document](Title: Brantley York) Richard Brantley York (January 3, 1805 – October 7, 1891) was a Methodist minister and educator best known for founding and serving as president of the institution that would become Duke[. . .]
.
.
.
[Document](Title: Randolph County, Illinois) Owing to its role in the state's history, the county motto is "Where Illinois Began." It contains the historically[. . .]
[Document](Title: Minsk Region) Minsk Region or Minsk Voblasć or Minsk Oblast (, "Minskaja vobłasć" ;, "Minskaja oblastj") is one of the regions of Belarus. Its administrative center is Minsk, although it is a separate administrative[. . .]
[Document](Title: Mount Franklin (Australian Capital Territory)) Mount Franklin is a mountain with an elevation of in the Brindabella Ranges that is located on the border[. . .]

Question: When did the town WIZE is licensed in become capitol of the state where Brantley York was born?

Response:
<model continues from here...>