Improving LLM Long Context Understanding via Synthetic Data and Adaptive Compression

by

Jerry Li

B.S. Computer Science and Engineering, Massachusetts Institute of Technology, 2024

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by:	Jerry Li Department of Electrical Engineering and Computer Science May 17, 2024
Certified by:	Rogerio Feris MIT-IBM Watson AI Lab, Thesis Supervisor
Certified by:	Leonid Karlinsky MIT-IBM Watson AI Lab, Thesis Supervisor
Certified by:	Aude Oliva MIT-IBM Watson AI Lab, CSAIL, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Improving LLM Long Context Understanding via Synthetic Data and Adaptive Compression

by

Jerry Li

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Recent innovations in large language models (LLMs) have led to their widespread use, but the long context problem remains a fundamental challenge. Transformer-based LLMs are constrained by the quadratic scaling of the self-attention mechanism, which restricts most popular LLMs to a context length of several thousand tokens. Many methods have been introduced to extend the context of LLMs, including the Activation Beacon approach. In this work, we propose two key advancements to the existing methodology. First, we generate long context synthetic data across a variety of tasks for training context-extended models, which can supplement or even replace expensive human-annotated data. Second, we introduce a novel two-pass, adaptive compression technique for more intelligent compression of long contexts. We find that the two strategies lead to orthogonal performance improvements on real-world long context tasks, resulting in an overall 4.2% increase in accuracy compared to the previous benchmark.

Thesis supervisor: Rogerio Feris
Title: MIT-IBM Watson AI Lab

Thesis supervisor: Leonid Karlinsky
Title: MIT-IBM Watson AI Lab

Thesis supervisor: Aude Oliva
Title: MIT-IBM Watson AI Lab, CSAIL

Acknowledgments

I would like to thank:

Rogerio Feris, Leonid Karlinsky, and Aude Oliva for being the main supervisors of this project, without whom this project would not have been possible. They provided extensive support, feedback, and inspiration throughout the year, despite having to put up with my nonlinear progress.

Dima Krotov, who also provided great suggestions and feedback.

And of course, my amazing friends and family for their unwavering support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, rapid innovations in large language models (LLMs) have caused them to become ubiquitous due to their impressive comprehension and contextual understanding abilities. OpenAI's ChatGPT has quickly become a popular option for anything from writing essays to solving math problems to developing apps. However, one fundamental challenge that remains for Transformer-based LLMs is the long context problem. Specifically, due to the Transformer [1] self-attention mechanism scaling quadratically with the input, the context length for many popular LLMs remains restricted to several thousand tokens. Meta's recently released Llama 3, for example, is still constrained to a context length of just over 8,000 tokens (about 6,000 words), which is only sufficient for processing a few articles, a few code files, or a few chapters of a book.

A small context length is a key limitation for many potential applications. For example, the original motivating use case for this work was to build a storytelling AI for sports matches, utilizing an LLM to automatically write sports articles based on match information, player interviews, and previous knowledge. However, without resolving the long context problem, such a system would be difficult to build given that a single player interview alone may take up most of the context length - restricting the additional input required for the AI to write a cohesive, descriptive story. In summary, a significantly extended context length would open up many new possibilities such as feeding entire books or codebases into the LLM, potentially replacing the need for methods such as Retrieval-Augmented Generation (RAG) altogether.

Given the importance of context extension, it is no surprise that substantial research has been focused in this direction. Previous works have attempted to resolve the long context problem in a variety of ways, such as through modifying the LLM architecture, attention mechanism, and/or positional embeddings, or through learning a compressed version of the context. While some of these methods have found success and recent claims of $>$ 1,000,000 token context lengths have come forth, several challenges nonetheless face the majority of these methods. First, many of the methods rely on finetuning with human-annotated/-generated long context data. Due to the nature of the task, obtaining quality long context data in this manner is both labor-intensive and expensive. Second, the resulting model

accuracy is often weaker on long context tasks - such as failing trivial needle-in-a-haystack tests - even when a "successful" context extension is claimed. Evaluations are often done using only language modeling loss, and performance on real-world long context tasks is regularly omitted or poor.

1.2 Contribution

Hence, in this work, we explore orthogonal methods of addressing the two aforementioned challenges. We build off the previous Activation Beacon [2] work, which processes long contexts by using a sliding window to process one short interval of the context at a time. It is trained to compress each window into fewer tokens, thereby allowing for an extended context. Unfortunately, Activation Beacon suffers from both of the previous shortcomings - it is finetuned on human-annotated data, and its accuracy suffers in long context tasks due to compression losses.

To address these shortcomings, we finetune Activation Beacon on synthetic (LLM-generated) long context data. The synthetic data is constructed in a novel self-extension manner: rather than utilizing a stronger teacher model such as GPT-4 for generation, the target LLM (the LLM for which the context length is being extended via finetuning) is itself used to generate the dataset. The data is inexpensive, easy, and quick to obtain. Using the same teacher model as the student ensures that gains in long context understanding cannot be the result of knowledge distillation. Furthermore, self-extension is particularly useful when working with a state-of-the-art model or a highly specialized one, for which an appropriate teacher model is difficult to obtain. To address the second concern, we devise a novel two-pass, adaptive compression methodology to improve accuracy on long context tasks. On the first pass, the relevancy of sections of the context are identified using attention scores. The second pass re-processes the long context with an emphasis on the relevant sections, applying little to no compression on the important parts while heavily compressing the irrelevant portions. We evaluate the effect of both of these changes on a standard long context understanding benchmark and demonstrate that they lead to orthogonal improvements in accuracy.

To summarize, our main contributions are as follows:

- We generate **long context synthetic data** for training context-extended models and demonstrate its effectiveness as a supplement - or even replacement - for human-annotated data when finetuning. Notably, we introduce a novel **self-extension** methodology where the LLM to be finetuned is itself used for data generation.
- We implement a novel **two-pass, adaptive compression** methodology for improving performance on long context tasks. We demonstrate performance gains orthogonal to the synthetic data improvements.

1.3 Related Work

1.3.1 Long Context Methods

In this section, we briefly explore previous works related to LLM context extension, with a focus on methods most similar to ours.

Memory and Compressed Memory. The earliest and most naive approach to handling long contexts was to divide it into segments and pass each through the Transformer. However, this is ineffective as it loses all context between segments. Transformer-XL [3] was designed to learn longer-range dependencies - beyond the fixed-length context - by keeping a cache of the previous segment's activations (termed the "memory") at each Transformer layer. The Transformer architecture was then modified to allow attending to the previous activations, creating a recurrence mechanism and thereby enabling the learning of longer-range dependencies. Later, the Compressive Transformer [4] was introduced as an improvement upon Transformer-XL: instead of caching the previous activations directly, they are first compressed using a learned convolutional operator. This further extended the maximum learnable context length. More recently, the Memorizing Transformer [5] replaced the compression mechanism with a much larger cache and a k-nearest-neighbor (kNN) attention mechanism. Instead of dealing with compression losses, the most relevant raw activations are queried from the cache using kNN to serve as the "memory." This tradeoff between giving up all information contained in the less-relevant activations in the cache and being able to use the most relevant activations in raw, uncompressed form was demonstrated to be effective.

Using Attention as a Compressor. Mu et al. [6] realized that, instead of learning a separate compression operator as in the Compressive Transformer, the LLM's own self-attention mechanism itself could be utilized as the compressor. They introduced Gisting, which learns to compress arbitrary prompts into much fewer gist tokens. Training could be done by simply making minimal modification to the attention mask. However, Gisting was focused on the prompt compression task rather than long context modeling. It was Landmark Attention [7] that, in simultaneous work, used a related idea for context extension. Landmark Attention trains a special landmark token to represent each context window and uses attention scores on the landmark tokens to retrieve relevant windows from memory. This - in a similar way to the Memorizing Transformer - allows for inference with arbitrary context length. Finally, Activation Beacon [2] combined ideas from Gisting and the Compressive Transformer by compressing each window into fewer beacon tokens and accumulating those as memory for the following windows. Its mechanism is described in significantly more detail in Chapter 2, as we build directly off this work.

Other Notable Methods. Aside from memory-based methods, there are numerous other strategies for adapting LLMs for long context. Perhaps the most well known are methods such as Position Interpolation [8] and NTK-Aware scaling, which manipulate the LLM's RoPE [9] positional encoding to allow the LLM to handle unseen context lengths at inference time. These methods are notable because they are orthogonal to ours. In a separate vein, StreamingLLM [10] achieved reasonable perplexity on context lengths of millions of tokens by simply discarding all tokens from the cache except for the first and most recent ones. They uncovered the attention sink phenomenon: LLMs place a disproportionate amount

of attention on the first token, which makes standard sliding window attention methods ineffective. By simply keeping around the first token, perplexity was largely recovered. StreamingLLM is mentioned because the attention sink phenomenon plays an eventual role in our methodology.

Retrieval-Augmented Generation. A related field of LLM research worth mentioning is Retrieval-Augmented Generation (RAG), which involves retrieving additional documents to aid LLMs in producing accurate, relevant output - thereby no longer requiring all of the knowledge to be stored in the model parameters. RAG can be thought of as a way to deal with long context at the input level, but the top-k documents retrieved by RAG may still not fit into the LLM context window. Therefore, context compression is useful as a complementary technique to RAG. If sufficiently long context lengths are achieved, it may even serve as a replacement.

1.3.2 Synthetic Data

In this section, we discuss previous works related to training LLMs with synthetic data.

Synthetic Data for Instruction Tuning. Initially, much of the work in synthetic data was directed toward instruction tuning. Standard instruction tuning for an LLM requires collecting thousands of quality human-annotated samples for supervised fine tuning (SFT) and reinforcement learning with human feedback (RLHF). To make data collection cheaper and faster, Self-Instruct [11] was introduced, which leverages an LLM's capabilities to generate numerous samples from a small "seed" set of human examples. Several extensions and improvements to Self-Instruct followed, a notable example being the training method for Taori et al.'s [12] Alpaca model, which used a stronger teacher model as the synthetic data generator to train the student. More recently, IBM's LAB [13] methodology introduced a data curation taxonomy along with a multi-phased instruction-tuning strategy with replay buffers to train a model which obtained competitive performance to those trained on human-annotated data.

Synthetic Data for Long Context. Besides instruction tuning, a few recent works have found success in using synthetic data specifically for long context LLM training. An et al.'s [14] INformation-INtensive (IN2) training methodology leverages a strong LLM to generate question-answer pairs about a piece of text, which is then combined with distracting text to form a long context. They showed that training with this data specifically helps address the "lost-in-the-middle" problem, where LLMs tend to focus on the beginning and end of a long context, forgetting information in between. More recently, Zhang et al. [15] finetuned the new Llama 3 model using long context samples generated with GPT-4, extending the original 8K context length to 80K tokens. In addition to synthetic QA pairs, they also included tasks such as summarization in the synthetic data. Finally, Xiong et al. [16] proposed a method most similar to the self-extension method we employ: leveraging the aforementioned Self-Instruct to generate synthetic data to extend Llama-2-Chat's context length, using Llama-2-Chat itself. However, our approach remains distinct in that we do not require an initial set of example samples nor do we require an existing corpus from which to extract long contexts.

Chapter 2

Preliminaries

This section is intended as an abbreviated, high-level introduction into the Activation Beacon [2] method, for those unfamiliar, as a strong conceptual understanding is needed for this work. Please refer to the original work for a more complete explanation.

2.1 Activation Beacon

2.1.1 Overview

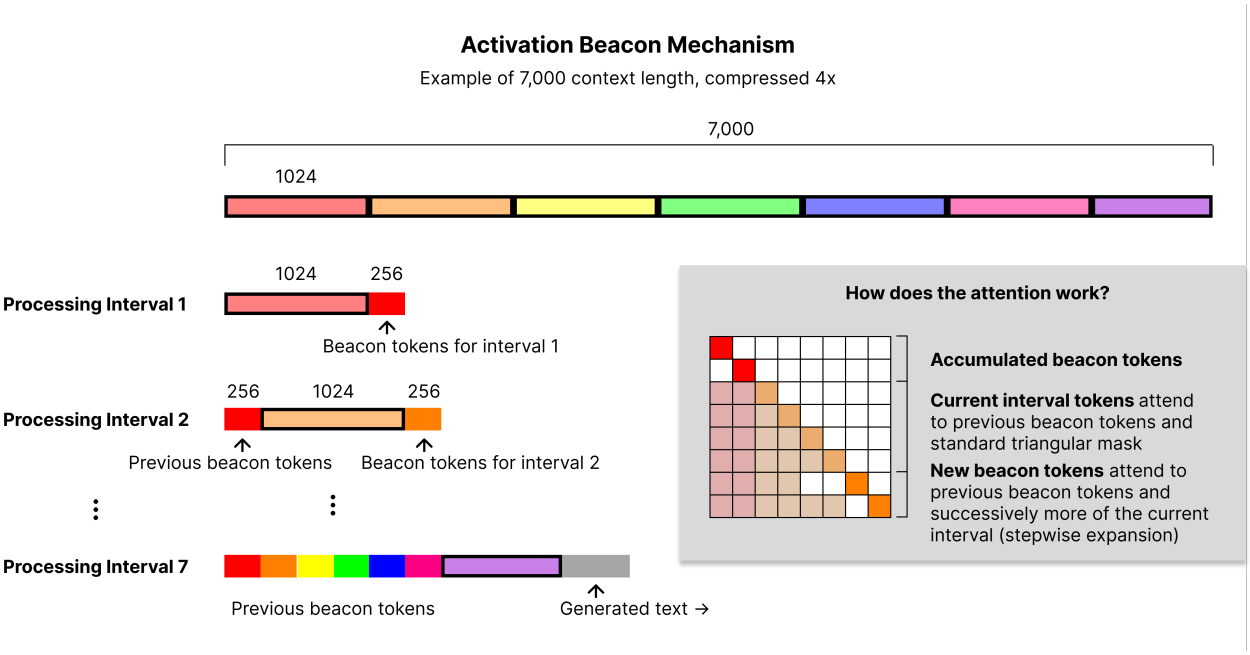

Figure 2.1: Full diagram of Activation Beacon.

At a high level, Activation Beacon processes a long context by partitioning it into shorter intervals and processing those sequentially. When processing each interval, Activation Beacon leverages the LLM's own self-attention mechanism to compress the interval into fewer

"beacon" tokens. These tokens are accumulated and prefixed when processing future intervals, serving as a compressed memory of the past context. Let's examine how this process works in more detail.

First, note that we introduce two primary modifications to the LLM: a special beacon token, as well as a copy of the LLM's multi-head attention (MHA) parameters which is exclusively used for processing beacon tokens, which we denote MHA^b. Also note that throughout this work, we use a context interval of 1,024 tokens. Let's look at how Activation Beacon condenses a single interval. First, $\frac{1024}{r}$ beacon tokens are appended to the end of the sequence, where r represents the compression ratio. Then, the sequence is processed via auto-regression, just as in a standard LLM. When processing the beacon tokens, however, the MHA^b parameters are used. The overall effect is to condense the raw key and value activations of the context into the condensed activations of the beacon tokens. Note that Activation Beacon employs a specific attention mask for the beacon tokens, termed stepwise expansion: each beacon token attends to one more span than the previous. This is demonstrated in Figure 2.1.

When processing the next interval, the condensed activations of the beacon tokens from the previous interval are prepended to the 1,024 context (while the raw activations are discarded). Then, we process the second interval just as before, with the distinction being that all tokens in the sequence can attend to the previous beacon tokens. We continue processing the remaining intervals in this manner, accumulating the condensed activations in the prefix. Consequently, assuming the LLM has a context length of 4,096, Activation Beacon is able to process contexts of up to length $3072 \times r + 1024$, where r is the compression ratio, since 1,024 is allocated to the raw context interval and $4096 - 1024 = 3072$ remains for the accumulated beacon tokens.

2.1.2 Training

For training, only the beacon token embedding and the MHA^b parameters are tuned; the rest of the LLM is frozen. This is because we only need the LLM to learn to use beacon tokens to compress activations; we don't want to affect the normal token inference. Training is performed using standard auto-regression.

Activation Beacon is trained on a mixture of data from RedPajama (pretraining) and LongAlpaca (instruction tuning). It trains only on short samples of between 1,200 and 8,192 tokens long for efficiency, as the recursive nature of the process means that it can be applied at inference to much longer context lengths than those seen in training. Furthermore, during training, the compression ratio for each interval is sampled at random from $\{2, 4, 8, 16, 32, 64, 128\}$. This ensures that the model sees a large range in the number of prepended beacon tokens during training, which helps the emergent behavior on unseen longer context lengths. Additionally, it teaches the model to condense contexts at a variety of compression ratios.

During inference, the compression ratio is chosen to be the minimum which can still process the entire context, and is the same for all intervals.

Chapter 3

Methods

3.1 Synthetic Data for Self-Extension

3.1.1 Overview

We generate long context synthetic data for training context-extended LLMs, as an inexpensive supplement or replacement for human-annotated data. Specifically, we generate data covering four unique long context tasks: question-answering (QA), summarization, few-shot QA, and few-shot summarization. We employ a self-extension methodology: the LLM which is being trained for context extension is the same as the one used for generating the synthetic data. This is useful for two reasons: it ensures that performance gains cannot be the result of mere knowledge distillation from a stronger teacher model, and it's necessary when working with a state-of-the-art or highly specialized model. A note on terminology: through the remainder of this section, "target LLM" will be used to refer to the LLM which is used for both data generation and training.

3.1.2 Data Generation

To construct our synthetic samples, we begin with the "great noun list," a list of the 6,801 most frequently used common nouns in the English language. The target LLM is then prompted to generate a long story about the noun; the noun essentially serves as a random seed to produce a large variety of different stories. Next, for each story, we prompt the target LLM to generate a question-answer pair about the story, as well as a brief summary. During story generation, we use sampling with high temperature (0.7) and high top-k (50) parameters for the LLM to encourage creativity and diversity in the produced stories. In contrast, we use low temperature (0.1) and low top-k (10) parameters when generating the question-answer pairs and summaries, so as to ensure factuality and relevancy. The exact prompts used for these generations are provided in the Appendix.

With the question-answer pair and summary for each story, we can now construct the samples for each of the four long context tasks. Regardless of task, we begin by selecting $N \in [3, 14]$ random stories/nouns. The bounds are chosen to ensure that the generated example has a context length approximately between 1,200 and 8,192 tokens. The remaining

steps are task specific, and the training example templates for each task are presented in thorough detail in Figures 3.1 and 3.2.

- **QA**: We construct the long context by concatenating the N stories. Then, we randomly select one of the N stories and its corresponding question-answer pair. The long context training example is constructed by joining the long context and question as the input, and the answer is the desired output. Notably, the story for which the question is asked may be present anywhere (beginning, middle, or end) in the context, encouraging the model to remember information from the entire context.

- **Summarization**: We construct the long context in an identical manner to QA. Then, instead of only one story, we randomly select $n \in [1, \min(5, N)]$ stories and their corresponding summaries. The training example's input is formed from the long context followed by a prompt to summarize the n chosen stories (identified by number and noun), while the output is composed of the n summaries concatenated together. Again, this task requires the model to memorize all of the stories in the long context to succeed.

- **Few-shot QA**: By definition, few-shot examples are very different from the previous ones. The input is composed of $N-1$ (story, question, answer) examples concatenated together, the "shots," followed by the story and question for the Nth noun. The output is then the answer for the Nth example. Few-shot training tasks are included to improve the model's in-context learning abilities over long contexts.

- **Few-shot Summarization**: The training examples for few-shot summarization are constructed nearly identically to few-shot QA, with the only distinction being summaries replacing the question-answer pairs.

In this manner, we generate 4,000 training examples for QA, 2,000 training examples for summarization, and 1,000 training examples each for the few-shot tasks. Note that our methodology is very efficient - as creating the training examples doesn't require further LLM prompting beyond the initial story, QA, and summary generation - and can easily be scaled to much longer context lengths simply by choosing larger N.

3.1.3 Training

For all of our experiments, the target LLM is Llama-2-7B-chat, which has a context length of 4,096. The training data totals 160,000 samples and is comprised of 143,000 examples sampled from RedPajama, 9,000 examples from LongAlpaca, and the 8,000 synthetic samples as previously described. The RedPajama samples represent standard long context "pretraining" data, while LongAlpaca is a human-annotated instruction-following dataset. Each sample has a context length between 1,200 and 8,192.

We train for one epoch of the whole dataset, or 160,000 steps, on a 8xA100 GPU node. Training takes around 11 hours. All remaining training parameters follow the original Activation Beacon paper, including a context interval of 1,024, a batch size of 8, and a linearly decaying learning rate starting at 5e-5.

Question-Answering

Input

Memorize the following stories and then answer the question based on the given stories.

Story 1 begins.
{Story about noun 1}
Story 1 ends.
⋮
Story N begins.
{Story about noun N}
Story N ends.

Answer the question based on the given stories.
Question: *{Question for story i}*

Output

Answer: *{Answer for story i}*

Summarization

Input

Memorize the following stories and then write a summary for each of the specifically requested stories.

Story 1 begins.
{Story about noun 1}
Story 1 ends.
⋮
Story N begins.
{Story about noun N}
Story N ends.

Write a summary for Story *{i_1}*, the story about *{noun i_1}*, a summary for Story *{i_2}*, the story about *{noun i_2}*..., and a summary for Story *{i_n}*, the story about *{noun i_n}*.

Output

Summary:
{Summary for story i_1}
⋮
{Summary for story i_n}

Figure 3.1: Synthetic sample template for question-answering and summarization.

Few-shot QA

Input

Answer the question based on the given story. The following are some examples.

Story:
{Story 1}
Question: *{Question for story 1}*
Answer: *{Answer for story 1}*
⋮
Story:
{Story N-1}
Question: *{Question for story N-1}*
Answer: *{Answer for story N-1}*

Story:
{Story N}
Question: *{Question for story N}*

Output

Answer: *{Answer for story N}*

Few-shot Summarization

Input

Write a summary for the given story. The following are some examples.

Story:
{Story 1}
Summary: *{Summary for story 1}*
⋮
Story:
{Story N-1}
Summary: *{Summary for story N-1}*

Story:
{Story N}

Output

Summary: *{Summary for story N}*

Figure 3.2: Synthetic sample template for few-shot QA and few-shot summarization.

3.2 Two-Pass Adaptive Compression

3.2.1 Overview

In addition to synthetic data, we implement an orthogonal method for improving long context understanding at *inference time*. The motivation is that, when performing most real-world long context tasks, not all parts of the context are equally useful. This is especially the case for tasks like question-answering, where only a short section of the context is needed to correctly answer the question. However, the original Activation Beacon method does not take this into account; the same compression ratio is applied throughout. This is a major problem as Activation Beacon cannot even reliably complete the trivial passkey retrieval task (retrieve a passkey placed at random within a large body of distracting text) at any context lengths above around 8,000, since the interval containing the passkey is compressed. We improve upon this by heavily compressing intervals of the context which we deem less relevant to the task, while keeping important intervals uncompressed or lightly compressed. This ensures that - as long as the relevant intervals are correctly identified - the LLM has a better memory of the important parts to complete the task.

3.2.2 Two-Pass Mechanism

Adaptively compressing intervals based on their importance requires two passes over the long context: the first pass to compute relevancy (and by implication, a compression ratio for each interval), and the second pass to actually generate the output with the appropriate compression ratios. In general, the first pass can be any retrieval-esque method which can handle long context, such as BM25. However, instead of using a separate method for the first pass, our inspiration is to simply perform an initial pass with Activation Beacon itself, then use attention scores on the beacon tokens of each interval as a measure of importance. A diagram of this method is presented in Figure 3.3 and described in more detail below.

For the first pass of Activation Beacon, we process the long context with a compression ratio of 8. 8 is chosen because it is the lowest compression ratio which can handle a context length up to 15,500, which is what we require (compression ratio of 4 can only handle a maximum context of $4 \times 3072 + 1024 = 13312$). This means that each context interval of 1,024 is compressed into 128 beacon tokens. After processing the entire context, we do not generate output. Instead, we look at the attention scores on the beacon tokens of each interval. Specifically, we average the attention scores at the final index across all layers, all heads, and all 128 beacon tokens to produce a single attention score for each context interval. We normalize them to sum to 1. Now, we have a measure of relevancy for each interval: if the model attends strongly to a certain interval when about to generate the answer, it is likely that that interval contains important information.

For the sake of clarity, let's follow a real example throughout the next few steps. After normalizing to sum to 1, the attention scores for this sample (which happens to have a context length filling 7 intervals) were

$$(0.2178, 0.1436, 0.2178, 0.0757, 0.1079, 0.1299, 0.1118)$$

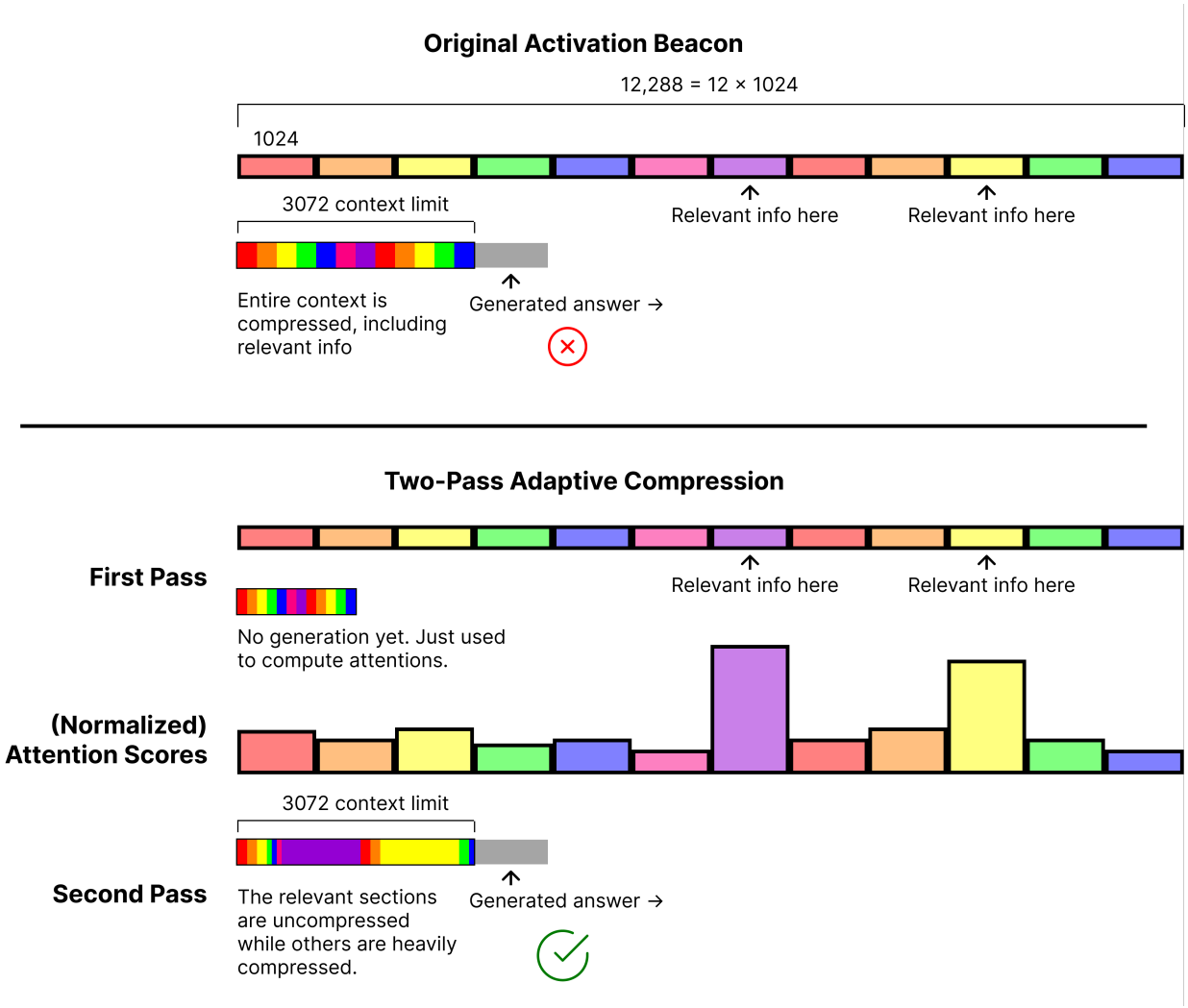

Figure 3.3: Full diagram of the two-pass, adaptive compression method.

3.2.3 Normalizing Attention Scores

To go from a distribution of attention scores over intervals to a compression ratio for each interval requires a few additional steps. The first issue is the attention sink phenomenon. Previous works [10] have observed that LLMs place disproportionate amounts of attention on the first few tokens, regardless of whether those tokens actually contain important information. More generally, the LLM attention distribution has biases that exist regardless of the context - not just on the attention sinks but also on recent tokens, while middle tokens tend to go under-attended - which make a uniform prior inadequate. Therefore, while it seems from the example that the first interval is very relevant, this may not actually be the case.

We handle this by using a calibration dataset to obtain the "natural" attention distribution over beacon tokens. The distribution, of course, depends on how many total context intervals there are. Therefore, we sample 50 calibration contexts for each number of intervals between 2 and 15 from a dataset containing 10,000 RedPajama samples, 10,000 LongAlpaca

samples, and 4,000 synthetic QA samples. Then, we process each sample with Activation Beacon and a compression ratio of 8 (to align with the first pass). Finally, for each number of context intervals (2-15), we compute the mean and standard deviation for the attention score at each interval.

So, we take our attention distribution and normalize it using a z-score calculation with the calibrated mean and standard deviation. We clamp the z-score between -3 and 3 and then exponentiate the result with a base of 2. This transformation gives us only positive attention scores where a scale factor of 2 correlates to a standard deviation difference.

Let's go back to our example. For 7 context intervals,

- Calibration means: $(0.2002, 0.0913, 0.0967, 0.1152, 0.1289, 0.1562, 0.2119)$
- Calibration stds: $(0.0786, 0.0271, 0.0280, 0.0354, 0.0396, 0.0432, 0.0742)$

Computing z-scores and exponentiating yields

$$(1.1641, 3.8125, 8.0000, 0.4609, 0.6914, 0.6562, 0.3926)$$

Seems like the answer may be hidden in the third interval!

3.2.4 Computing Compression Ratios

We are ready to compute compression ratios, or equivalently, the number of beacon tokens to allocate to each interval. The second perspective is the one we take. We first allow for a temperature parameter α, which is applied as an exponent to each of the attention scores. α controls the extent to which we compress or don't compress intervals; a higher α means that even a small difference in attention scores could lead to a large difference in compression. Intuitively, the more sure we are that our attention scores are choosing the right intervals, the higher the α we should use.

Next, we normalize the attention scores to sum to the maximum context length available for beacon tokens, 3072, which gives us the appropriate number of beacon tokens to allocate to each interval. We round the numbers to the closest valid beacon token amount (with some additional nuance to prevent any overflows of the 3072 max length), finally yielding the desired compression ratio for each interval.

In our example, using $\alpha = 1$ for simplicity, we get $(235, 772, 1616, 93, 140, 133, 79)$ for the allocations, which translates to compression ratios of

$$(4, 2, 0, 4, 2, 4, 8)$$

The last step, of course, is to perform the second pass, using Activation Beacon to process the long context and applying the calculated compression ratio for each interval. Note that the original Activation Beacon method, by comparison, would've applied a compression ratio of 2 for all intervals, potentially losing valuable information in the third interval.

Chapter 4

Results

4.1 Experiment Setup

We evaluate both of our modifications on the five real-world tasks in the LongBench [17] dataset: single-document QA, multi-document QA, summarization, few-shot learning, and code completion, totalling 3,350 test samples. Following the original Activation Beacon paper, and because most of the LongBench samples have context length less than 16,000, we truncate all contexts to 15,500 tokens for evaluation. For the Llama-2-7B-chat baseline, contexts are truncated to 3,500 tokens to fit within the 4K context length.

Note that the Llama-2-7B-chat and Activation Beacon (RP+LA) baselines are run ourselves and differ from those presented in the Activation Beacon paper.[1]

4.2 Synthetic Data Results

Table 4.1: Results on LongBench. RP = RedPajama, LA = LongAlpaca, S = Synthetic

Method (Data)	SingleDoc	MultiDoc	Summary	FewShot	Code	Avg
Llama-2-7B-chat	24.12	23.3	24.45	**63.11**	55.33	36.83
Activation Beacon (RP+LA)	**25.71**	28.39	23.31	58.1	55.66	36.99
Activation Beacon (RP+LA+S)	25.15	**29.89**	**24.59**	58.74	**58.22**	**37.97**
Activation Beacon (RP+S)	25.48	29.14	23.58	55.93	58.12	37.05

We present the results for our synthetic data experiments in Table 4.1. Note that in this section, we do not use the two-pass technique as we are focusing on the independent effects of the synthetic data. We find strong evidence that training on our synthetic data leads to improvements on long context understanding tasks. The original Activation Beacon - trained on RedPajama and LongAlpaca data - is not much stronger than Llama-2-7B-chat, with an average accuracy of 36.99 on the LongBench dataset. Training on an additional 8,000

[1]We were not able to replicate the exact results in the paper. Communication with the authors revealed that they used additional training tricks not specified in their paper nor code, so we use our own baselines.

samples of synthetic data increases the accuracy by 2.6% to 37.97. Notably, the increase is observed across all tasks except for single-doc QA, suggesting that, perhaps due to the diversity in our synthetic tasks, the model improves across a diverse range of evaluation tasks as well. A possible reason for the lack of improvement in single-doc QA is that the long contexts in our synthetic data do not perfectly align with the task, since they contain a variety of shorter stories concatenated together rather than a single long story.

Furthermore, we perform an ablation study by removing the human-annotated LongAlpaca data entirely, replacing it only with synthetic data for finetuning (recall that RedPajama is just a large pretraining corpus and can be kept as is). Results show that Activation Beacon finetuned only on synthetic data achieves comparable performance to the model finetuned on human-annotated instruction tuning data, with an accuracy of 37.05. Overall, our findings suggest that synthetic long context data can serve as a useful supplement, or even replacement, for far more expensive human-annotated finetuning data.

4.3 Two-Pass Results

Table 4.2: Results on LongBench with various α.

Method (Data)	SingleDoc	MultiDoc	Summary	FewShot	Code	Avg
Llama-2-7B-chat	24.12	23.3	24.45	**63.11**	55.33	36.83
Activation Beacon (RP+LA)	**25.71**	28.39	23.31	58.1	55.66	36.99
Activation Beacon (RP+LA+S)	25.15	29.89	**24.59**	58.74	**58.22**	37.97
Two Pass (RP+LA, $\alpha = 3$)	25.44	27.27	23.58	60.48	55.45	37.23
Two Pass (RP+LA+S, $\alpha = 1.5$)	25.47	28.27	24.53	60.6	57.27	37.94
Two Pass (RP+LA+S, $\alpha = 3$)	25.06	29.96	24.36	61.89	57.52	38.49
Two Pass (RP+LA+S, $\alpha = 4$)	24.52	**30.57**	24.39	62.15	57.43	**38.55**

We present the results for our two-pass adaptive compression methodology in Table 4.2. We analyze how two-pass impacts performance, with and without synthetic data. When two-pass is applied on the baseline Activation Beacon model (no synthetic data), we observe a modest improvement in accuracy of 0.6% to 37.23 on the LongBench dataset. However, when applied to the Activation Beacon model trained with synthetic data, we see a larger performance increase of 1.5% to 38.55, with an appropriately chosen α. The increase is driven primarily by improvements on the multi-doc QA and few-shot learning tasks.

Overall, we find that the two-pass methodology leads to orthogonal improvements in accuracy, albeit smaller compared to synthetic data. By utilizing both improvements, our key finding is that LongBench accuracy can be increased by 4.2% compared to the previous Activation Beacon benchmark.

4.3.1 Two-Pass Ablations

Only investigating the overall two-pass results doesn't reveal the full story for why it works or where the benefits are coming from. To better understand the mechanism, we performed

ablation studies on using different compression ratios and different retrieval methods with Activation Beacon.

Table 4.3: Results for different compression ratios. Evaluated on 10 samples per dataset.

Compression Ratio	SingleDoc	MultiDoc	Summary	FewShot	Code	Avg
128	14.29	10.62	21.09	56.43	54.85	29.79
64	15.26	**14.67**	22.31	50.64	57.2	30.22
32	17.07	9.67	22.53	52.32	**60.45**	30.41
16	21.24	14.61	23.42	58.28	60.4	33.82
Min (≤ 8)	**28.24**	13.33	**24.94**	**66.43**	56.95	**36.62**

In Table 4.3, we present the accuracies using different compression ratios, with the same compression ratio being used across all intervals.[2] Our key finding here is that our intuition for maintaining low compression ratios to improve accuracy is a correct assertion: average accuracies increase consistently as the compression ratio is decreased. Therefore, the motivation behind the two-pass methodology is valid.

Table 4.4: Results for different retrieval methods.

Retrieval Method	NarrativeQA	MultiFieldQA	HotpotQA	2WikiMQA	MuSiQue	Avg
Random	21.7	34.81	34.85	30.12	**18.03**	27.9
BM25	23.05	**38.69**	**38.81**	31.08	16.44	29.61
Attention	**25.49**	35.76	38.8	**34.23**	17.04	**30.26**

In Table 4.4, we verify the assertion that attention scores is a good proxy for identifying relevant and irrelevant context intervals. To do so, we evaluate 3 different methods of retrieving the relevant intervals: selecting at random, using BM25 (a simple bag-of-words retrieval algorithm), and using attention scores, just as in the two-pass mechanism. In all cases, we retrieve the top 5 most relevant intervals as selected by the tested method and compress those intervals minimally (compression ratio 2) while applying higher compression to the remaining intervals. We evaluate on the specific datasets in LongBench for which we expect retrieval to be most relevant, primarily those in the single-doc QA and multi-doc QA categories.

We can draw two key conclusions from our findings: 1) attention is the strongest of the three methods as a relevancy measure, and 2) adaptively compressing intervals based on their relevancy - another fundamental idea behind the two-pass method - is also valid.

4.3.2 Passkey Retrieval

Finally, one of the most egregious shortcomings of the original Activation Beacon model was that it failed one of the most naive long context evaluation tasks, passkey retrieval.

[2]Results are presented on a smaller subset of the LongBench dataset for efficiency; these results are not directly comparable to those found in the previous tables.

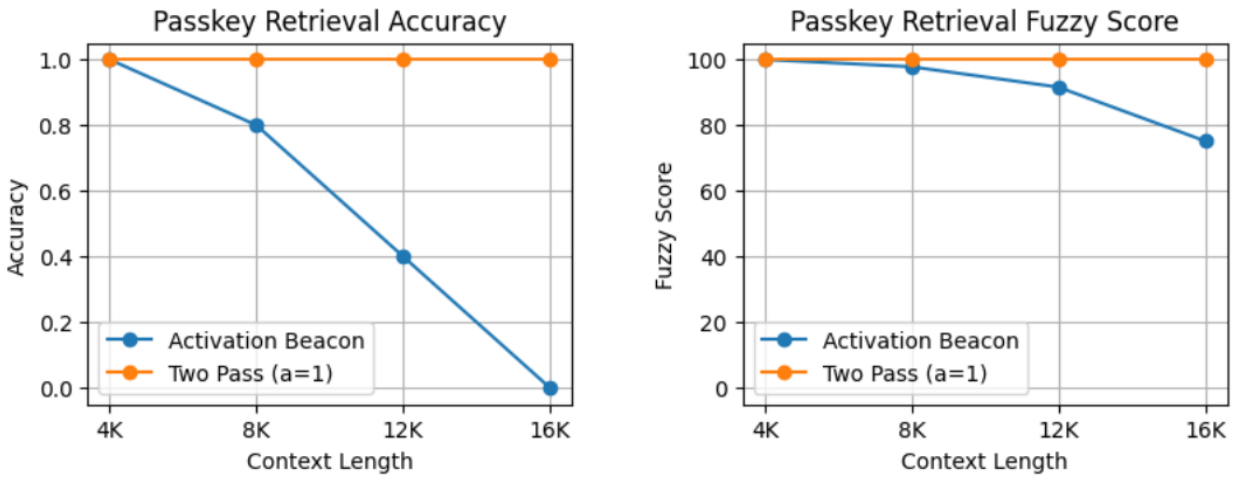

Figure 4.1: Passkey retrieval accuracy and fuzzy scores.

In passkey retrieval, the phrase "The passkey is {passkey, some 5-digit number}" is hidden among a distracting long context, and the model is prompted to recite the passkey. As seen in Figure 4.1, Activation Beacon cannot remember the passkey fully accurately at a context length of 8,000 and beyond, though the fuzzy scores (which measure overlap in the digits) reveal that some of the passkey is correctly maintained. This, of course, is due to the compression being applied to the passkey.

By applying the two-pass method directly with $\alpha = 1$, we see that the passkey is now retrieved with perfect accuracy up to 16,000 context length (it is likely to work beyond 16,000, but this is not tested as it requires calibration of additional compression ratios for the first pass). Manually examining the compression ratios being applied in the second pass reveals that the mechanism is correctly leaving the interval containing the passkey uncompressed, allowing for the perfect retrieval.

Chapter 5

Conclusion and Future Work

We identify two shortcomings of current long context methods, specifically the need for expensive, human-annotated long context data and the poor performance of long context models on real-world long context tasks, and introduce two novel solutions. First, we create a synthetic dataset spanning four long context tasks in a self-extension manner: we leverage the LLM which is being trained itself to generate the synthetic data. Second, we implement a two-pass, adaptive compression methodology for more intelligent compression of long contexts compared to the existing Activation Beacon method. Our experiments demonstrate that these two strategies provide orthogonal improvements on Activation Beacon's performance on real-world long context tasks. We hope that this work will contribute to the continued advancement of long context understanding in LLMs, enabling additional practical applications.

5.1 Efficiency

One major drawback worth noting about the two-pass methodology is that, well, it requires two passes. This means (nearly) twice the compute and (nearly) twice the inference time compared to the original method. While trading off efficiency for accuracy could be reasonable, improving the efficiency of the two-pass method is clearly a major future objective. There are several potential methods to do so, such as using a smaller draft model to perform the first pass more efficiently, or to utilize some or all of the beacon tokens calculated in the first pass in the second pass without re-processing them.

5.2 Multi-LoRA

Speaking of efficiency, introducing around 2B new parameters in the form of MHA^b is also inefficient. One strategy to reduce the number of new parameters to be introduced is with LoRA [18]. To counteract potential accuracy losses due to the reduction in parameters, it is logical to introduce a unique LoRA adapter for each compression ratio, which we term Multi-LoRA. Similarly, we can also introduce unique beacon tokens for each compression ratio, rather than using a single beacon token (a single embedding is very cheap in terms

of parameters). Both of these modifications mean that the model can train a mixture of "experts" specialized for each compression ratio, rather than making a single set of parameters responsible for all ratios. Initial experiments show that this method achieves comparable performance to MHA^b, while reducing the number of added parameters significantly, so additional work in this direction is desired.

5.3 Synthetic Data

Another exciting future direction is in improving the synthetic data generation. Models tend to benefit from diverse training data, and we are interested in devising additional tasks for the synthetic dataset, such as generating multi-hop question/answer pairs. Also, the way we concatenate stories to form the long context is a bit naive; there is no continuity or similarity between neighboring stories, which rarely reflects real-world use cases and may be the reason for the poor performance on single-doc QA. Clustering stories by semantic similarity and/or using prompts to interlink stories would be a logical next step toward generating higher quality training examples.

Appendix A

Synthetic Data Prompts

Story Prompt

Write a long story with long paragraphs about *{noun}*. Do not ask questions and do not add itemized items.

Story:

QA Prompt

You are given a story. Create only one specific question and only one answer based on the story. The question should be specific to the story, not a general question that could be asked of any story, and the answer should be based only on information given in the story.

Story:
{story}

Create only one specific question and only one answer based on the story. The question should be specific to the story, not a general question that could be asked of any story, and the answer should be based only on information given in the story.

Question:

Summary Prompt

You are given a story. Write a summary for the story.

Story:
{story}

Write a summary for the story.

Summary:

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[2] P. Zhang, Z. Liu, S. Xiao, N. Shao, Q. Ye, and Z. Dou, *Soaring from 4k to 400k: Extending llm's context with activation beacon*, 2024. arXiv: 2401.03462 [cs.CL].

[3] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, *Transformer-xl: Attentive language models beyond a fixed-length context*, 2019. arXiv: 1901.02860 [cs.LG].

[4] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, *Compressive transformers for long-range sequence modelling*, 2019. arXiv: 1911.05507 [cs.LG].

[5] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy, *Memorizing transformers*, 2022. arXiv: 2203.08913 [cs.LG].

[6] J. Mu, X. L. Li, and N. Goodman, *Learning to compress prompts with gist tokens*, 2023. arXiv: 2304.08467 [cs.CL].

[7] A. Mohtashami and M. Jaggi, *Landmark attention: Random-access infinite context length for transformers*, 2023. arXiv: 2305.16300 [cs.CL].

[8] S. Chen, S. Wong, L. Chen, and Y. Tian, *Extending context window of large language models via positional interpolation*, 2023. arXiv: 2306.15595 [cs.CL].

[9] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, *Roformer: Enhanced transformer with rotary position embedding*, 2023. arXiv: 2104.09864 [cs.CL].

[10] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, *Efficient streaming language models with attention sinks*, 2024. arXiv: 2309.17453 [cs.CL].

[11] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, *Self-instruct: Aligning language models with self-generated instructions*, 2023. arXiv: 2212.10560 [cs.CL].

[12] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, *Alpaca: A strong, replicable instruction-following model*, 2023.

[13] S. Sudalairaj, A. Bhandwaldar, A. Pareja, K. Xu, D. D. Cox, and A. Srivastava, *Lab: Large-scale alignment for chatbots*, 2024. arXiv: 2403.01081 [cs.CL].

[14] S. An, Z. Ma, Z. Lin, N. Zheng, and J.-G. Lou, *Make your llm fully utilize the context*, 2024. arXiv: 2404.16811 [cs.CL].

[15] P. Zhang, N. Shao, Z. Liu, S. Xiao, H. Qian, Q. Ye, and Z. Dou, *Extending llama-3's context ten-fold overnight*, 2024. arXiv: 2404.19553 [cs.CL].

[16] W. Xiong, J. Liu, I. Molybog, *et al.*, *Effective long-context scaling of foundation models*, 2023. arXiv: 2309.16039 [cs.CL].

[17] Y. Bai, X. Lv, J. Zhang, *et al.*, *Longbench: A bilingual, multitask benchmark for long context understanding*, 2023. arXiv: 2308.14508 [cs.CL].

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].