TOWARDS EFFECTIVE MODELING OF LONG-RANGE CONTEXT

A Dissertation Presented

by

SIMENG SUN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2024

Robert and Donna Manning College of
Information and Computer Sciences

TOWARDS EFFECTIVE MODELING OF LONG-RANGE CONTEXT

A Dissertation Presented

by

SIMENG SUN

Approved as to style and content by:

Mohit Iyyer, Chair

Andrew McCallum, Member

Brendan O'Connor, Member

Brian Dillon, Member

Ani Nenkova, Member

Ramesh K. Sitaraman, Associate Dean for
Educational Programs and Teaching
Robert and Donna Manning College of
Information and Computer Sciences

ACKNOWLEDGMENTS

First and foremost, I would like to extend my deepest gratitude to my advisor, Mohit Iyyer, for his invaluable guidance and mentorship over the years. I am thankful for the support he provided in helping me develop my own research taste, encouraging me when I struggled with negative results, and offering me insightful feedback on many things, including the very details of my works and presentations. I am greatly indebted to him for empowering me to grow as a researcher. I feel fortunate to have learned from him, and am grateful for being a part of the UMass NLP lab.

I would like to thank my committee members Andrew McCallum, Brendan O'Connor, Brian Dillon, and Ani Nenkova. Their refreshing views, thoughtful comments and constructive advice helped shape this thesis into its final form. I would also like to extend special thanks to Ani for her guidance during my master's study at UPenn and for sparking my passion for research. Her mentorship has led me to pursue a field I find deeply rewarding.

This thesis would not have been possible without the support of my friends in Amherst. I am grateful for the warm companionship and friendship with Xiang Lorraine Li, Weiqiu You, Yixiao Song, Mengxue Zhang and her feline friends. The hiking trips, interesting conversations, and movie and game nights are my precious memories. I appreciate the helpful discussions with Nader Akoury, Yapei Chang, Andrew Drozdov, Dhawal Gupta, Marzena Karpinska, Kalpesh Krishna, Tu Vu, and Shufan Wang. Special thanks to Nader for teaching me how to parallel park, and to Marzena for bringing so much joy to the lab. Studying and living in Amherst has also been enjoyable because of Yutao Zhou, Lijun Zhang, Xiao Liu, Chau Pham, Yekyung Kim, Katherine Thai, Katie Keith, Wenlong Zhao, Chenyun Wu, Zhan Xu, Zhipeng Tang, Erica Cai, Marisa Hudspeth, and Ankita Gupta. I would like to thank David Jensen for teaching my favorite seminar at UMass, Georgia Stuart and other Unity staff for allowing smooth usage of the cluster, and Eileen Hamel, Elizabeth Parolski and Kyle Skemer for their administrative support.

I am also grateful to people I met outside of Amherst and friends living far away. I thank Varun Manjunatha for his guidance during my internship at Adobe, James Cross, Angela Fan, and Maha Elbayad for their mentorship during my internships at Meta, and Yang Liu for his guidance during my internship at Microsoft. I would like to express my gratitude to Steven Tan, Haoran Xu, and Prashant Rajput for their companionship during the in-person internship in 2022 summer. Additionally, I would like to thank Zixin Zhang for introducing me to the OTW Open Doors project and volunteer activities. I also want to thank the friends I made many orbits ago – Yaohua Ma and Jiahua Kang, for the regular online movie parties and their ongoing support. I also appreciate the many authors whose writings impacted me profoundly in multiple ways.

Finally, I extend my heartfelt gratitude to my family, especially my parents, for their unwavering care and constant encouragement throughout this journey.

ABSTRACT

TOWARDS EFFECTIVE MODELING OF LONG-RANGE CONTEXT

MAY 2024

SIMENG SUN

B.Sc., BEIHANG UNIVERSITY

M.Sc., UNIVERSITY OF PENNSYLVANIA

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Mohit Iyyer

At the core of recent advancements in natural language processing are language models, which are trained to predict the next token given the preceding context. Recent developments in deep learning has led to the efficient scaling of context window in Transformer-based language models. Despite the progress, these models still exhibit severe limitations when tackling long-context tasks, such as book-level summarization and long-document question answering.

While the context window size has been continuously increasing, there is a lack of understanding on how these models utilize long-range context, or context that spans at least several thousand tokens. As such, we first provide an analysis of long-range context modeling with both perplexity and segment-level task evaluations. Our results show that perplexity, the most commonly used intrinsic metric for language model evaluation, may obscure the evaluation of long-range context modeling. In contrast, segment-level evaluation, which involves computing the probability of a sequence of tokens rather than a single token as done in perplexity, proves to be a more suitable method for evaluating long-range context modeling. Based on this finding, we enhance the segment-level evaluation by proposing a challenge dataset ChapterBreak, and demonstrate that SuffixLM, a model trained with segment-level signals, outperforms the standard token-level language model in this task.

The limited context modeling capability prompts us to investigate new ways to improve recent large language models. To this end, we first develop a prompting framework, PEARL, by leveraging large instruction fine-tuned language models to decompose complex reasoning into executable plans. We demonstrate the efficacy of PEARL on a subset of the long-document QA dataset, where the correct answer depends on the long-range context instead of a short excerpt. Our second approach builds on the benefits of modeling context at the segment level. Concretely, we propose a new training method, SuffixRL, by fine-tuning a token-level language model directly using segment-level signals. We show that training models with SuffixRL leads to more natural and coherent continuations in an open-ended generation setting.

Finally, we conclude this thesis by proposing three general directions and identifying seven concrete topics that hold promise for future exploration. We hope this thesis can spur more principled research in long-context modeling.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Language modeling is the task of assigning probabilities to sequences. At its core, language modeling involves predicting the probability of the next token given its preceding context, applying the chain rule of probability. J.R. Firth emphasizes the inseparability of context and the process of deriving meanings, describing the latter as *"a serial contextualization of our facts, context within context, each one being a function, an organ of the bigger context and all contexts finding a place in what might be called the context of culture." [58]* While it remains an ongoing discussion [14] as to whether language models can derive "meaning" from context, the role of context is unmistakably crucial—it forms the basis for predicting the next token in language modeling. Earlier approaches relied on short preceding contexts, such as n-grams [18], to estimate sequence probability. Recent advancements in hardware, optimization, and architectural design, exemplified by Transformer [187], now enable pre-training of language models on thousands of tokens from scratch [74].

Significance of Modeling Long-range Context. Due to the lack of consensus on the definition of *long-range context*, here we loosely define it as spanning at least several thousand tokens. Recent advances in scaling language model pre-training [130, 6], and successful application of reinforcement learning from human feedback (RLHF) [132] have led to the development of many general AI assistant systems such as ChatGPT [131]. When provided with appropriate prompts, these systems exhibit promising performance on many short-context tasks [215, 201], such as generating code snippet [185, 223, 77], paraphrasing [222, 93], and answering questions [130, 217]. Despite the promising results, recent works show that, when dealing with long-context tasks, these models still face challenges, such as inadequacy in resolving contextual nuances in document-level machine translation [81], incoherence and content omission in book-level summarization [23], and ineffective context utilization in multi-document QA [109].

One straightforward method to address long-context tasks is to augment language models that have a limited context window with retrieval techniques [205, 211]. For example, when answering a question about a long narrative, such as "*How is Oscar related to Dana?*" regarding the movie *Ghostbusters II*, a relevant excerpt implying their relationship can be extracted from the long movie script and inserted into the limited context window for the language model to produce the final answer. This retrieval-augmented approach is effective in dealing with simple queries and is also widely employed to answer open-domain questions by retrieving information from sources like Wikipedia [104, 164].

While retrieval-based methods ease the challenge of processing long context in one shot, they may not be suitable for cases where maintaining *continuity* in the context is crucial. For instance, answering a question like "*How does the tone change throughout the story?*" requires properly segmenting the narrative into salient plots, summarizing various sections, and analyzing the tone transitions. Simple retrieval-based methods, therefore, may not be immediately useful for processing the global dependencies mentioned previously. In addition to *continuity*, long context also gives rise to various other complex global dependencies, such as referencing previous examples/statements/lemmas in lengthy mathematical proofs and later chapters of a textbook, processing code repositories of millions of lines of code, and analyzing patterns and norms in accumulated log files. Processing all of these dependencies requires effective modeling of the long-range context and is not immediately resolved through retrieval-based methods.

Research questions. Having established the necessity of long-range context modeling, we delve into two basic research questions that form the basis of this thesis:

- **RQ #1:** How do language models utilize long-range context?

- **RQ #2:** How can we improve the modeling of long-range context?

We provide an overview about **RQ #1** in section 1.2, with more detailed evaluations and analyses presented in Chapter 2 and Chapter 3. For **RQ #2**, we offer an overview in section 1.3, and attempt to answer the question in Chapter 4 and Chapter 5.

Scope of this thesis. While various modalities of context exist (e.g., image, speech), this thesis focuses on natural language context in textual form. We emphasize that mixing multiple modalities of context is an exciting future direction in Chapter 6. Our results and discussions primarily center around Transformer-based LMs given their superior performance compared to other methods [18, 15, 120]. Recent years, new training objectives have emerged, including masked loss [37] and denoising objective [103]. However, in this thesis, our focus is solely on decoder-only causal language modeling [142].

1.2 Evaluation of Long-range Context Modeling

Since the introduction of Transformer [187], efforts to improve the efficiency of self-attention [28, 35] have led to a proliferation of long-range Transformer language models. While the initial models were trained mostly on short sequences such as 512 tokens [187, 38, 143], context window of more recent models can now reach up to the level of 100k tokens by applying various acceleration techniques [25, 148, 74].

Limitation of Perplexity Evaluation. Despite the progress, the ways in which these models take advantage of the long-range context remain unclear. In chapter 2, we perform a fine-grained analysis of two long-range Transformer language models that accept input sequences of up to 8K tokens, one of which achieves the best performance on PG-19 long-sequence LM benchmark in 2021. Our results reveal that providing distant long-range context to these models only improves their predictions on a small set of tokens (e.g., those that can be copied from the distant context). This finding is directly manifested in the marginal gains in perplexity when incorporating long-range context beyond the local 2K tokens. One implication of this finding is that perplexity may not be well-suited for evaluating long-range context modeling. Perplexity, which involves averaging over all tokens in a validation corpus, is dominated by the majority of tokens that can be well-predicted with short-range local context. This can result in misleading model comparison on long sequences, as the difference in perplexity can be largely explained by the effectiveness of modeling local-context rather than long-range context. As such, we pivot to segment-level evaluation, where the model needs to make predictions about a sequence of tokens rather than a single next token.

Segment-level Evaluation of Long-context Modeling. In the latter part of Chapter 2 and throughout Chapter 3, we delve into a novel task: *in-book suffix identification*. This segment-level evaluation requires the model to differentiate the gold suffix segment from multiple negatives extracted from the same book. Unlike single next-token prediction, each segment comprises multiple tokens, carries more information, thereby making it more challenging to predict compared to a single next token. We conjecture that, similar to next-token prediction, accurate prediction of a future segment relies on the access to extended preceding segments. In Chapter 3, we empirically verify this hypothesis with SuffixLM, a segment-level language model specifically trained to predict gold suffixes. Our results indicate that incorporating more preceding segments allows SuffixLM to achieve better performance in identifying gold future segments. Moreover, on a challange dataset CHAPTERBREAK, which we propose to evaluate long-range context modeling, SuffixLM also outperforms other (large) language models trained with next-token prediction objective, including recent LLaMA-2 [183], Mistral-7B [76] and state space model Mamba-2.8B [56].

1.3 Towards Effective Modeling of Long-range Context

In addition to the analyses in Chapter 2 and Chapter 3, several studies also demonstrate the sub-optimal performance of long-context modeling in recent large language models [23, 109, 78]. This raises a natural question: how can we improve the modeling of long-range context in language models? To answer this question, we propose methods from two distinct directions – one focusing on a novel prompting framework, while the other modifies the training objective of language models.

Prompting Large Language Models to Reason Over Long Context. The first direction we explore depends on large pre-trained language models. These models demonstrate promising reasoning capability on short-context tasks through the use of chain-of-thought prompting [200, 127], which decomposes a complex task into intermediate steps. However, it remains unclear how to apply such methods to reason over long documents, in which both the decomposition and the output of each intermediate step are non-trivial to obtain. In Chapter 4, we propose PEARL, a prompting framework to improve reasoning over long documents. PEARL consists of three stages: action mining, plan formulation, and plan execution. More specifically, given a question about a long document, PEARL decomposes the question into a sequence of actions (e.g., SUMMARIZE, FIND_EVENT, FIND_RELATION) and then executes them over the document to obtain the answer. Each stage of PEARL is implemented via zero-shot or few-shot prompting of LLMs with minimal human input. We evaluate PEARL on a challenging subset of the QuALITY dataset, which contains questions that require complex reasoning over long narrative texts. Our result indicates that PEARL is a promising first step towards leveraging LLMs to reason over long documents.

Training Language Model with Segment-level Signals. In Chapter 3, we demonstrate the benefits of modeling context at the segment level with SuffixLM. Building on this insight, in Chapter 5, we introduce a new training method, SuffixRL, which involves training a token-level language model with segment-level signals. This method draws inspiration from the recent RLHF framework [132], which helps align large language models with human intents, and involves training models with proximal policy optimization (PPO) [157]. Given the resource-demanding nature of PPO training process, in the first part of Chapter 5, we propose two experimental changes that permit us to train large language models with limited resources. We validate our new experimental setup on AlpacaFarm [43] benchmark, and show that applying low-rank adaptation [66] and an alternative regularization method leads to better performance compared to strong baselines. With the validated setup, we apply SuffixRL to LLaMA-1-7B models [182] and demonstrate its efficacy in enhancing coherence in open-ended generation setting.

1.4 Future directions

Finally, in retrospect of the limitations of this thesis and other recent research, we conclude by proposing three general future directions regarding long-range context modeling that can be fruitful to explore in the future. These directions include (1) immediate follow-up directions of SuffixRL, (2) exploration of alternative evaluation methods for long-context tasks in the era of large language models, and (3) alternative architectural designs and training practices for long-range foundation models. For each of the general directions, we provide two to three concrete topics, detailing the current research status, as well as potential obstacles and solutions.

1.5 Declaration of Collaborations

This thesis covers works from multiple papers, which were in collaboration with researchers specified as below:

- *Do Long-Range Language Models Actually Use Long-Range Context?*
  Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. Accepted to EMNLP 2021.

- *ChapterBreak: A Challenge Dataset for Long-Range Language Models*
  Simeng Sun, Katherine Thai, Mohit Iyyer. Accepted to NAACL 2022.

- *PEARL: Prompting Large Language Models to Plan and Execute Actions Over Long Documents*
  Simeng Sun, Yang Liu, Shuohang Wang, Chenguang Zhu, Mohit Iyyer. Accepted to EACL 2024.

- *Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of RLHF*

  Simeng Sun, Dhawal Gupta, Mohit Iyyer. Preprint 2023 September.

CHAPTER 2

DO LONG-RANGE LANGUAGE MODELS ACTUALLY USE LONG-RANGE CONTEXT?

2.1 Introduction

Understanding long documents requires modeling various discourse-level phenomena, including anaphora [60, 55], argument structure [54], narrative scripts and trajectories [152, 98], and causal links between concepts [123]. Unfortunately, most language models (LMs) are trained to predict the next word given only a small window of local context, which prevents them from using long-range discourse structure to improve their predictions. Many research efforts over the years have attempted to address this issue: for example, [147] incorporated statistics from distant tokens to improve n-gram models, while [75] added document-level context to neural LMs.

More recently, the Transformer LM [187], which forms the backbone of state-of-the-art NLP systems [38, 20], has become the focus of numerous efforts to process longer input sequences. Transformer LMs are constrained by the inefficiency of the self-attention mechanism, whose complexity scales quadratically with the input sequence length. As such, more efficient methods based on sparse attention [32] and cached memory [145] have been proposed to increase the maximum sequence length, which has progressed from GPT-2's 1024 tokens [143] to 4096 tokens [218] and finally 8192 tokens [149]. When evaluated on the PG-19 benchmark dataset [145], which contains long documents in the public domain, these "long-range" Transformer LMs also reach lower perplexities than baseline models on held-out data.

How do "long-range" Transformer LMs make use of the long-range context?[1] Do they actually encode important discourse information to improve their predictions? In this chapter, we conduct a series of fine-grained analysis experiments to answer these questions, several of which are inspired by the context analysis of LSTM LMs conducted by [84]. We focus specifically on analyzing the behavior of the state-of-the-art Routing Transformer and a simpler baseline model in the presence of various perturbations (e.g., word shuffling, random document replacement), and look closely at how different types of tokens are affected. Our results show that:

- Providing long-range context (i.e., further than 2K tokens away) to these models has *negligible* impact on the perplexity of tokens near the end of a sequence in aggregate. However, a fine-grained analysis reveals that it does help a small set of tokens (tokens within subword clusters and those that can only be copied from the distant context), as well as particular types of books (fictional and continuous).

- Despite the aforementioned improvements on a fraction of tokens, significantly perturbing the long-term context with word shuffling and random replacement has no notable impact on perplexity overall, suggesting that the evaluated models encode long-range context superficially at best.

- Long-range context is not used for sequence-level prediction tasks that move outside the teacher-forced setting of the previous experiments.

While modern LMs can process much longer input sequences than those of the past, we conclude that they do not exploit the information available in the long-range context. We recommend that future research on long-range LMs includes analysis experiments such as those in our work to shed light on how and when they are using the distant context.

[1] In this chapter, we adopt a slightly constrained definition of long-range context, which denotes the context start from the beginning of a long sequence and spanning up to several thousand tokens.

2.2 Background & Setup

In this section, we first provide an overview of the long-range language models analyzed in this work (Local Transformer, Routing Transformer). Next, we describe the experimental setup used in the remainder of this chapter to measure the impact of the long-term context.

2.2.1 Long-range Language Modeling

Given preceding context tokens $w_{<i}$ (the *prefix*), a language model (LM) computes the probability distribution of the next token $p(w_i \mid w_{<i})$. LMs are commonly evaluated using perplexity, which is the exponentiated negative log likelihood of a held-out corpus:

$$ppl = \exp\Big(-\frac{1}{N}\sum_{i=1}^{N} \log p(w_i \mid w_{<i})\Big)$$

Modern LMs are most often implemented with *Transformers* [187], which compute vector representations for every token in the prefix at multiple layers and combine them together using self-attention. Since self-attention requires scoring every token in the sequence against every other token, it scales quadratically in both compute time and memory usage, which limits its application to very long sequences.

Local Transformer A simple way to improve the efficiency of self-attention blocks is to constrain the attention at each layer to a local window of the previous k tokens. Such Transformers, which we refer to as *Local Transformers*, can be feasibly scaled up to large input sequence lengths. The receptive field of a Local Transformer scales linearly with the number of layers, as the l^{th} layer of this model can access the previous $k \times l$ tokens [116, 28, 173].

Routing Transformer The Routing Transformer [149] (RT) takes a more intelligent approach to scaling self-attention. Specifically, the RT assigns keys and queries in self-attention to k clusters, the centroids of which are learned during training. A routing attention strategy computes attention A only over the queries Q_i and keys K_j that belong to the same cluster $\mu(Q_i)$ (i.e., those whose centroid is closest to query Q_i).

$$X_i = \sum_{\substack{j:K_j \in \mu(Q_i), \\ j<i}} A_{ij} V_j$$

In contrast to the position-based local attention, this clustering-based attention takes the *content* of the token representations into account. This sparse self-attention mechanism reduces the complexity from $O(N^2)$ to $O(N^{1.5})$ and has led to state-of-the-art results on tasks such as long-form question answering [91].

2.2.2 Experimental setup

While the previously-described models can be trained with longer inputs than standard Transformers, it remains unclear how they make use of the additional context tokens to improve their predictions of the next word. To shed light on the behavior of long-range Transformer LMs, we perform a series of experiments in which we manipulate the input sequence (both length and content). For token-level experiments (§ 2.3, § 2.4), we only evaluate the perplexity of k tokens near the end[2] of an N token-long input sequence ($k \ll N$) to focus on the effect of long-range context.[3]

2.2.3 Dataset & Models

Dataset We conduct all of our analyses on the validation set of the PG-19 dataset [145]. This dataset contains ~29K books from Project Gutenberg repository published before 1919 and was constructed specifically to evaluate long-range LMs (average document length ~69K tokens). The validation set contains 50 books[4] and ~3 million tokens in total.

[2]An artifact exhibited by RT causes tokens at the very end of a sequence to have much higher losses than the others; this phenomena does not exist with LT. After correspondence with the RT authors, we decide to exclude the very last 40 tokens (whose losses are affected) from all of our evaluations. More details about this artifact can be found in the Appendix.

[3]Language models are normally evaluated on non-overlapping sequences, with the begin and end sequence tokens receiving different amount of context. In our setting, all target tokens have roughly the same amount of context.

[4]We observe a significant gap of ~10 perplexity between the PG-19 test and validation sets and discover that this gap is largely due to the presence of an annotated edition of *The Canterbury Tales and Other Poems*.

Evaluating every token in the validation set with large prefix sizes (e.g. 8K tokens) is computationally infeasible. On one RTX8000 GPU, it takes around 104h to evaluate the entire PG-19 validation set with sequence length 8K and target sequence 10. Thus, we set the number of target tokens per context $k = 10$ and sample a subset of 220K validation tokens for our experiments, which is the same data size used in the LM analysis experiments of [84]. Evaluating RT in this way yields slightly better perplexity on the validation set of PG-19 than using the evaluation setting in the original RT paper (35.2 vs. 36.3). We ensure that the number of tokens sampled from each validation book is proportional to the length of that book.

Models: As training long-range Transformer LMs is also infeasible without immense computational resources, we use publicly-available pretrained checkpoints for all of our experiments. The Routing Transformer (RT) checkpoint contains 490M parameters and processes sequences up to 8192 tokens long, achieving 33.2 perplexity on the PG-19 test set. The released checkpoint, which was trained on 128 TPUv3 cores for several weeks, has a subword vocabulary of $\sim$ 98K types, along with 8 attention heads in each of its 22 layers. We follow the RT paper [149] by scaling the loss by 1.248 before computing the perplexity in order to match the word-level perplexity reported by [145]. The top two RT layers include content-based clustering attention while the remaining are composed entirely of local attention.

Our Local Transformer (LT) is derived from the same checkpoint as RT (and thus has identical model size), except that all clustering heads are replaced with local attention heads. It achieves slightly better perplexity on the PG-19 test set (38.3 vs. 39.3) compared to the LT model trained from scratch by [149], possibly because the local attention heads learn a better representation of the weight matrices by using the information from the clustering

This book intertwines line-by-line annotations with the main text, which causes the preprocessed version in the dataset to be unreadable. We remove this book in all of our experiments, which decreases the test/validation perplexity gap to $\sim$3.

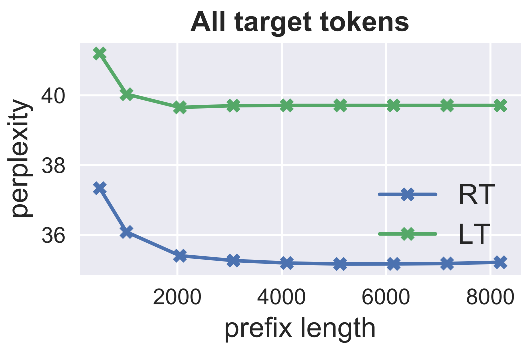


Figure 2.1: The perplexity of all target tokens plateaus after 2K prefix tokens for both Routing Transformer (**RT**) and Local Transformer (**LT**), showing the negligible overall impact of the long-range context.

heads. Note that the original fully trained LT checkpoint was not made publicly available before the EMNLP submission deadline. [5] The attention heads in this model attend to the previous 256 tokens, which results in an effective receptive field of $\sim$ 5K tokens. A preliminary experiment verified that the clustering heads in the RT do attend to the long-range context, beyond 5K tokens, demonstrating that it is at least theoretically incorporating more context than the LT. While we would have also liked to analyze other long-range LMs such as the Compressive Transformer [145] and Longformer [13], these models do not have publicly-available PG-19 checkpoints; additionally, they differ from RT and LT in model size, which makes it hard to perform controlled experiments.

2.3 The effect of longer context

How does the size of the prefix affect the perplexity of long-range Transformer LMs? In this section, we evaluate our RT and LT checkpoints on the PG-19 validation set with varied prefix length. We discover that although these models are theoretically able to encode

[5]The RT authors released a new LT checkpoint during the review period of this chapter. We also evaluate this newly-released LT checkpoint and include the results in the Appendix A.6. Both the RT and the LT checkpoints can be found at `https://github.com/google-research/google-research/tree/master/routing_transformer`

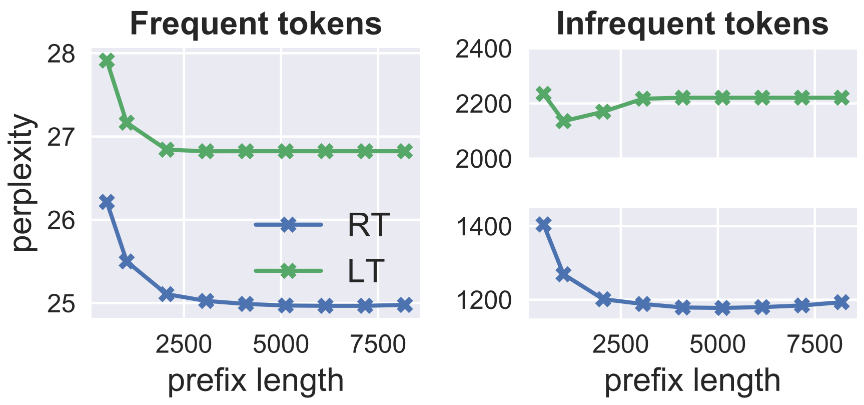


Figure 2.2: RT perplexity of infrequent tokens continues to decrease until prefix length is 5K.

long sequences, increasing the prefix length beyond 2K tokens does not bring discernible improvements in aggregate. However, we do identify small subsets of tokens that benefit from long-range context. Additionally, we find that these models take advantage of long-range context to different degrees on different types of books (e.g., continuous fictional narratives vs. discontinuous magazine articles).

Validation perplexity does not improve when the prefix length grows beyond 2K tokens: As shown in Figure 2.1, RT perplexity plateaus when evaluated with prefixes longer than 2K.[6] In contrast, relative to RT, the perplexity curve for the more primitive LT starts flattening earlier at around 1K tokens (note that its effective context size is only 5K). We conclude that while RT's clustering heads take better advantage of global context than LT, the long-range context beyond 2K tokens is not helping overall. Surprisingly, the perplexity gap between RT and LT is relatively consistent regardless of the prefix length, which indicates that much of RT's gains do not come from its increased ability to leverage long-range context but rather from better modeling of *local context*.

[6]As a point of comparison, the perplexity of the much smaller LSTM language models evaluated by [84] plateaus after 200 words. Additionally, [139] discover that the perplexity flattens after 1K for a smaller standard Transformer LM.

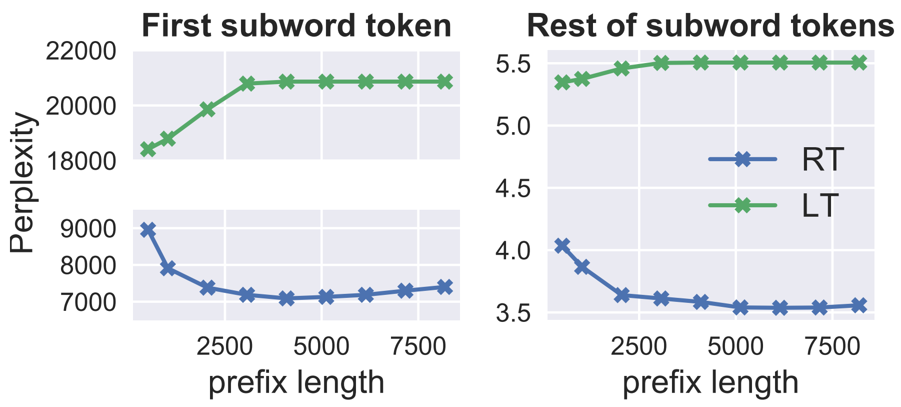


Figure 2.3: RT perplexity of tokens inside subword clusters continues to decrease until a prefix length of 5K.

Infrequent tokens can benefit from increased prefix length: While the overall perplexity does not improve with increasing prefix length, we do observe different behavior when filtering the target tokens by frequency, as shown in Figure 2.2. We define *frequent* tokens to be the top 10% more frequently-occurring tokens in the subword vocabulary of PG-19 while the rest of tokens are classified as *infrequent*.[7] While adding long-range context does not improve either model's predictions of frequent tokens, the RT's perplexity of infrequent tokens decreases from ~ 1200 with a prefix length of 2K to 1180 with prefix length of 5K. However, we do observe that infrequent token perplexity increases back to 1200 as the input is further extended, suggesting that the additional context perhaps confounds the model. Meanwhile, the LT is significantly worse than RT on infrequent token prediction, and its perplexity increases as the prefix length is increased.[8]

Tokens inside a subword cluster benefit from longer contexts: One issue with the previous experiment is that the frequency categorization was computed at a subword level

[7]Around 20K tokens in our target token set are classified as infrequent, which amounts to 9% of all target tokens.

[8]This is likely an artifact due to the elimination of routing attention heads from the RT checkpoint, since the LT trained from scratch does not exhibit such behavior. More details are included in Appendix A.6.

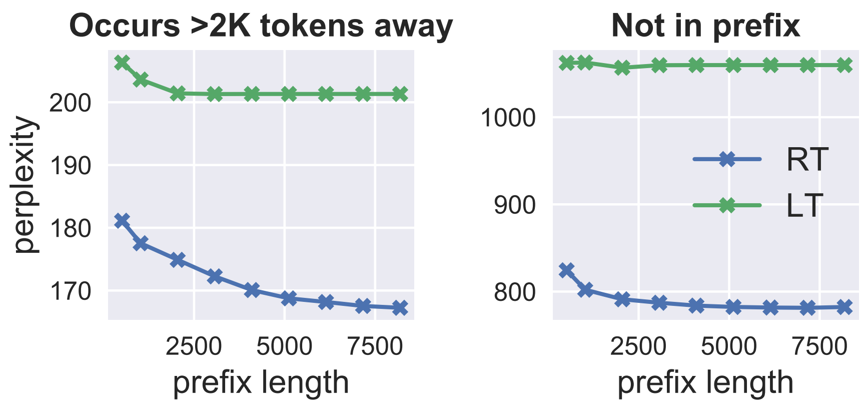


Figure 2.4: RT perplexity of target tokens whose last appearance is more than 2K tokens away in the prefix keeps decreasing.

and so may not exactly correspond to word frequency, especially for infrequent words (e.g., entities) that are split into multiple subwords. We therefore do a follow-up experiment by isolating all words that are split into multiple subwords, and then examining perplexity of these tokens as a function of their position in the subword cluster. For example, the word "Trocadero" is separated into three subword tokens "Tro", "cade", and "ro". We specifically distinguish between the *first* subword in the cluster ("Tro") from the *rest* of the subwords ("cade" and "ro") in the plots shown in Figure 2.3. The perplexities are computed over 4.1K *first* and 5.1K *rest* subword tokens. The *first* subword category exhibits the same curve shape as those for infrequent tokens for both models, although the magnitude of the perplexities is far higher. The rest of the subwords are far easier for both models to predict, but the RT perplexity curve shows some positive impact from the long-range context until a prefix size of 5K tokens.

Routing Transformers are able to copy tokens that occur in the long-range context: Target subword tokens that can be copied from somewhere in the prefix form another inter-

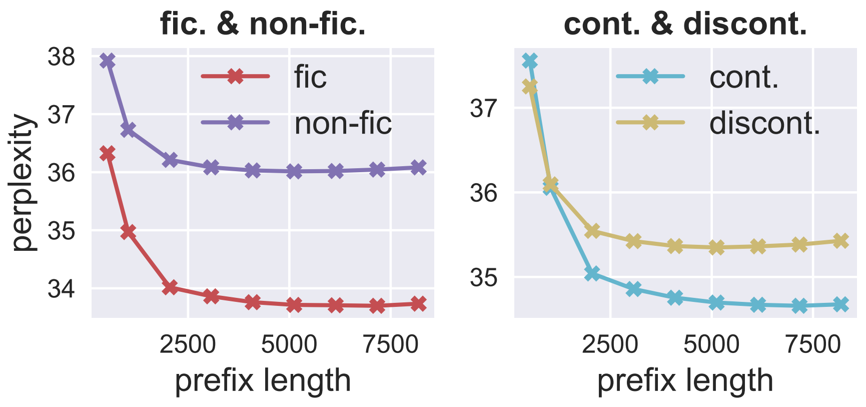


Figure 2.5: RT takes better advantage of context beyond 2K tokens for fictional and continuous books than non-fictional and discontinuous books, respectively.

esting group to analyze.[9] While this is commonplace for frequent words (e.g., determiners, pronouns), it also occurs for entities and rare words (e.g., character names in a novel); sometimes, a word can occur several thousand tokens after its last occurrence. We focus on the latter category of tokens, specifically using a prefix length of 2K tokens as a cutoff to distinguish local and long-range context. Perplexities are computed over 22k tokens which occur last time more than 2K tokens away, and 36K tokens that never appear in the prefix. In particular, the left plot in Figure 2.4 shows the perplexity of tokens that cannot be found in the previous 2K tokens, but occur somewhere in the long-range context (2K to 8K tokens away). While the LT curve for such tokens plateaus after 2K tokens, indicating that LT cannot take advantage of repeated words in the distant context, the RT curve steadily decreases until 8K tokens. The right plot, which shows the subset of target tokens which do not occur anywhere in the short or long-term context, decreases until about 5K tokens and then plateaus. Overall, these results show that long-range context is helpful for tokens that appear even several thousands tokens away.

[9]Note that there is some overlap between the token categories we have analyzed so far. We verify in the Appendix A.3 Table A.1 that the overlap between these categories is not significant enough to confound the results.

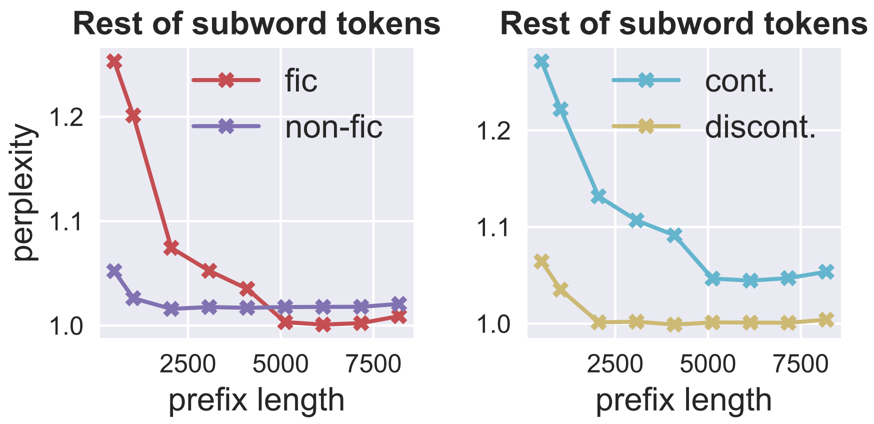


Figure 2.6: The majority of improvements on tokens inside subword clusters (i.e., excluding the first token in subword clusters) from prefixes longer than 2K tokens comes from fictional and continuous books.

Following patterns in the long-range context: Besides the token categories examined above, we also qualitatively look at some examples that are too infrequent to analyze at scale. Interestingly, we observe some simple patterns (slightly more complex than copying) that the RT model picks up on. Specifically, it learns to increment chapter numbers even if the previous chapter title appears more than 2K tokens away: for example, when predicting "Chapter V" in the validation book *Keith of the Border*, modifying the previous chapter title "Chapter IV", which occurs 2300 tokens away, to "Chapter V" causes the loss of the predicted token "V" to increase by over 10.

The impact of book type on the benefits of long-range context: PG-19 contains a diverse array of topics, genres, and formats, not all of which equally benefit from long-range context modeling. For example, while continuous narratives (e.g., novels) certainly build up many high-level discourse structures over a long sequence of tokens, discontinuous text like magazines, textbooks, or short story collections may require primarily local modeling. To better understand the effect of the type of book on long-range LM perplexity, we annotate

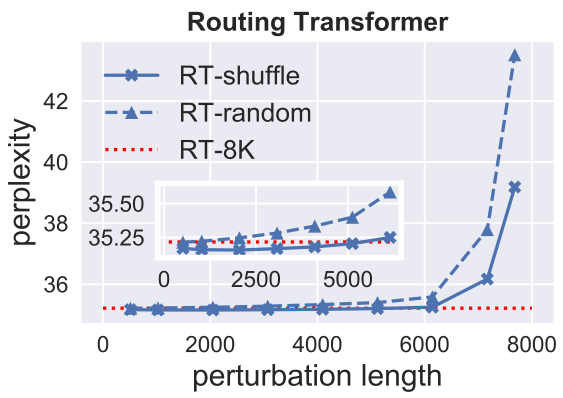


Figure 2.7: Perturbing up to 6K prefix tokens does not notably affect RT's overall perplexity. The corresponding plot for LT is included in Appendix A.4.

every book in PG-19's validation set as either *fiction* or *non-fiction* and *continuous*[10] or *discontinuous*. We also annotate whether the work has been written by the same author or various authors, which is presented in the Appendix. Out of 49 books we annotated, 30 are non-fiction, 31 are discontinuous, and 25 books are both non-fiction and discontinuous. Note that some magazines contain short stories or poems interspersed with news articles and essays; we count these as non-fiction in our analysis.

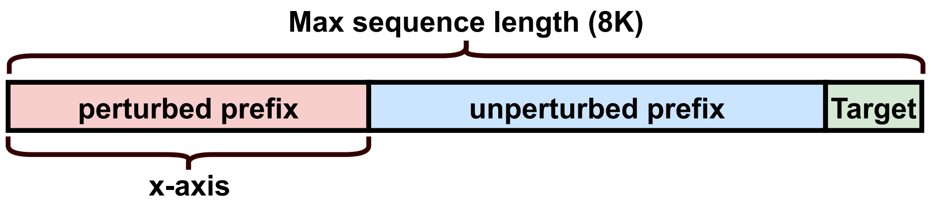


Figure 2.8: Generic illustration of the x-axis in all the perturbation analysis.

We observe in Figure 2.5 that the RT model takes better advantage of long-range context for fictional and continuous books, as the perplexity for these books plateaus at around 5K tokens. Figure 2.6 shows fictional and continuous books exploit better the long-range context

[10]We consider books with related but distinct sections (such as textbooks) to be discontinuous in our annotation.

while predicting tokens within subword clusters. Overall, we find the improvement stems largely from continuous and fictional books; more details are included in Appendix A.2.

2.4 The perturbation of long-range context

The experiments in the previous section show that incorporating long-range context (further than 2K tokens away from the target) yields only marginal improvements to the overall perplexities of RT and LT. However, the long-range context does have a notable positive impact on a subset of tokens and book types. Do these improvements persist in the presence of severe perturbations to the distant context? If so, this would indicate that they are not encoding any complex discourse structure in the context but rather relying on surface information (e.g., token presence) to make better predictions. In this section, we perform a perturbation analysis to quantitatively measure the robustness of the state-of-the-art RT model. Figure A.7 in the Appendix shows that the Local Transformer never uses context beyond 3K. Due to this limitation, we only present results on RT for in this section.

Formally, assume we are given a prefix sequence $P = (w_0, w_1, \ldots, w_n)$ with which we want to predict target sequence $(w_{n+1}, w_{n+2}, \ldots, w_{n+k})$. We apply a perturbation ρ to the first m tokens of the prefix ($w_{0:m}$) to obtain the perturbed prefix

$$\tilde{P} = (\rho(w_0, \ldots, w_m), w_{m+1}, \ldots, w_n).$$

We define the following three perturbation operations for ρ and report results averaged over five runs for each of them.

- **Sequence shuffling**: Tokens within the perturbed window $w_{0:m}$ are shuffled across the entire window (i.e., sentence boundaries are not respected).
- **Random sequence replacement**: $w_{0:m}$ is replaced with a random sequence from another validation book that is m tokens long.

- **Specific token drop**: Specific tokens within $w_{0:m}$ (e.g., those that occur in the target) are dropped and replaced with the padding token.

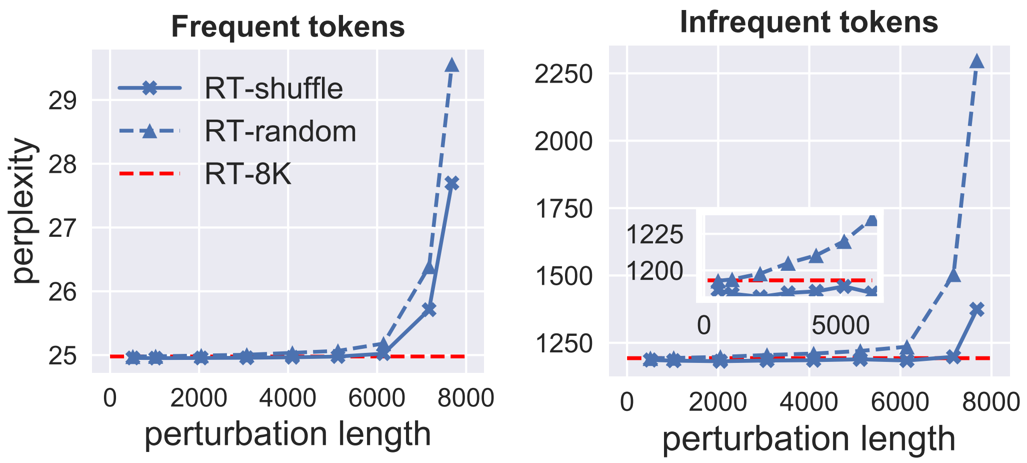


Figure 2.9: While perturbing up to 6K prefix tokens has little impact on frequent tokens, it increases the perplexity of infrequent tokens.

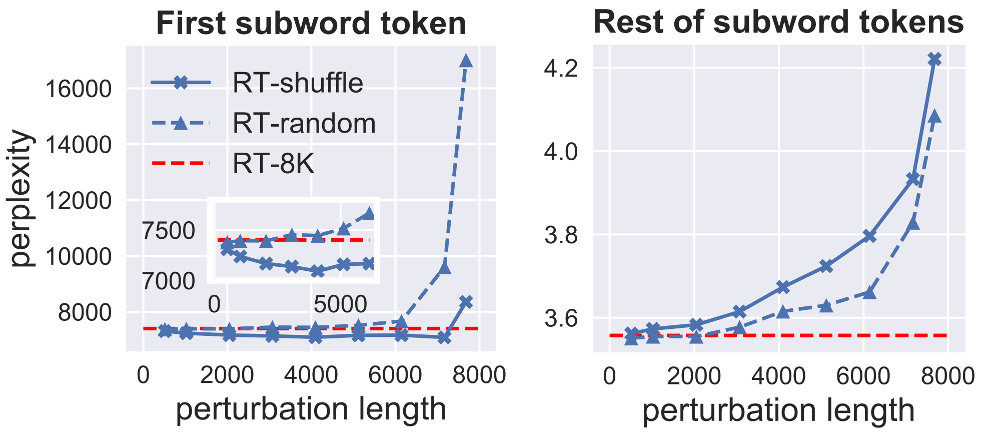


Figure 2.10: Both shuffling and random replacement increase the perplexity of tokens inside subword clusters, with the former having more negative impact.

Sequence-level perturbations further than 2K tokens from the target have minimal impact on perplexity: We first apply sequence-level shuffling and random replacement to the distant context. Both operations have minimal impact on the perplexity of all tokens (Figure 2.7) as well as frequent/infrequent tokens (Figure 2.9) provided at least 2K tokens are left unperturbed. However, these perturbations do have increasing impact as the perturbations come closer to the target, especially for infrequent tokens. Zooming in on the long-range context, we find that random replacement consistently results in higher perplexity than shuffling, but also that shuffling distant context actually achieves slightly *lower* perplexity than when the model is given completely unperturbed prefixes. Overall, these results demonstrate that RT is insensitive to the word order of the long-range context.

Tokens inside subword clusters and tokens repeated in the distant context depend on word order: Similar to the analysis in Section 2.3, the experiments above may hide impacts on small subsets of tokens, which motivates us to do a more fine-grained analysis. We find that tokens inside subword clusters (Figure 2.10) and those that can only be copied from long-range context (Figure 2.11) are sensitive to both the order and the content of the remote context. While random replacement is more harmful than shuffling for tokens that can be copied in the distant context (172 shuffled vs 174 random replacement perplexity when perturbing 6K tokens), shuffling is more detrimental for tokens inside subword clusters (3.8 vs 3.7 perplexity when perturbing 6K tokens).

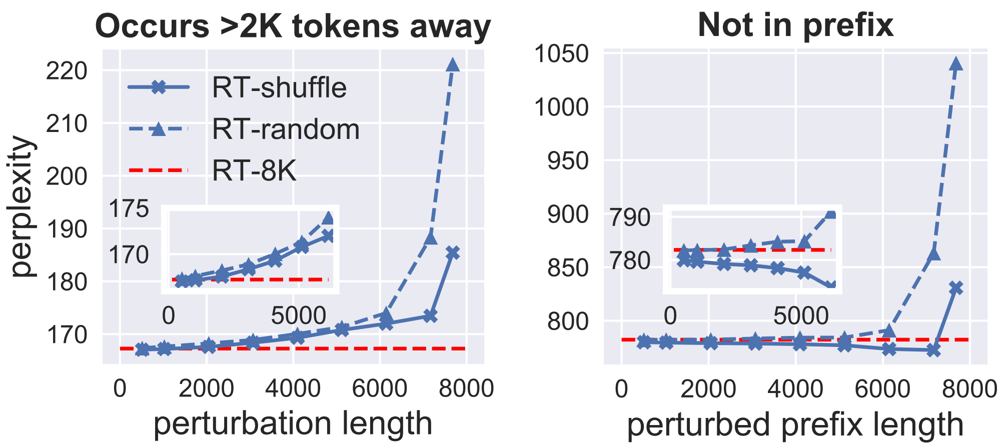


Figure 2.11: Both perturbation operations increase the perplexity of target tokens whose last appearance in the prefix is more than 2K tokens away.

Routing Transformer encodes token identity in the long-range context: While the previous perturbations affected entire contiguous blocks of the prefix, we move now to

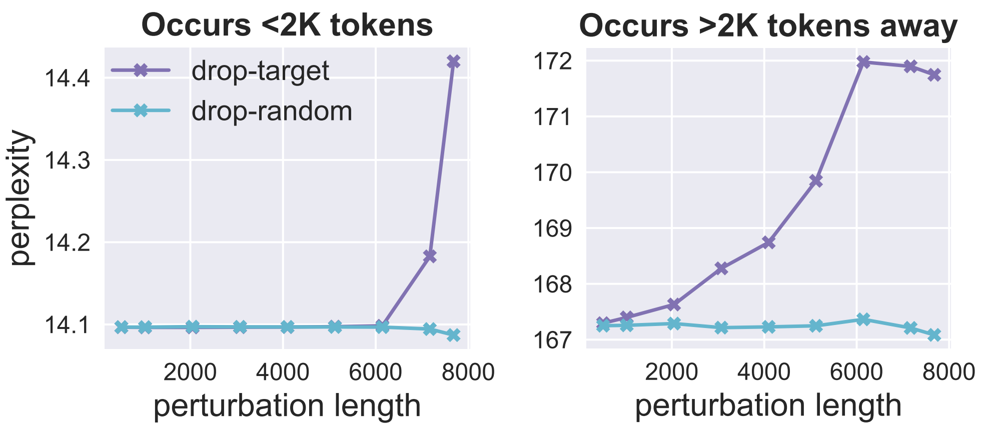


Figure 2.12: Perplexity of target tokens whose last occurrence in the prefix is within previous 2K tokens (**left**) or more than 2K tokens away (**right**), when dropping either these target tokens or random tokens in the perturbed range. The curve on the right indicates RT memorizes token identity in the distant prefix to some extent.

more targeted perturbations. An interesting question to ask given the observation that RT perplexity decreases on copied tokens as sequence length increases (§ 2.3) is how much that perplexity decrease depends on word order and surrounding content. In response, we drop tokens in the distant context whose next appearance is in the target sequence. As a control experiment, we drop the same number of random tokens for each perturbation length.

As shown in the right plot of Figure 2.12, dropping the previous long-range occurrences of target tokens increases the perplexity of those target tokens, which shows that RT indeed memorizes token identity in the long-range context to some extent. The left plot shows that dropping long-range duplicate tokens does not affect tokens that also occur within the local context (i.e., the prior 2K tokens). The flat curve before 6K indicates the model relies only on the most recent occurrences for prediction.

2.5 Sequence-level analysis

All of the previous experiments have focused on *token-level perplexity*, which is the standard way in which LMs are evaluated. However, the prefixes in these evaluations consist solely of ground-truth text, mirroring the "teacher-forcing" setup that LMs are trained with. When these models are deployed practically to generate text, they have to rely on their previous predictions instead of ground-truth text, and several prior works have noted different behavior in this setting [191, 62, 202]. In this section, we shift from token-level tasks to analyzing RT and LT performance on *sequence-level* tasks. In particular, we first look at how well the models can memorize an exact sequence in the distant context, as opposed to a single token as we did previously. Next, we examine the models' ability to identify which of six 128-token suffixes follows a given prefix, which examines their behavior outside the standard teacher-forced setting.

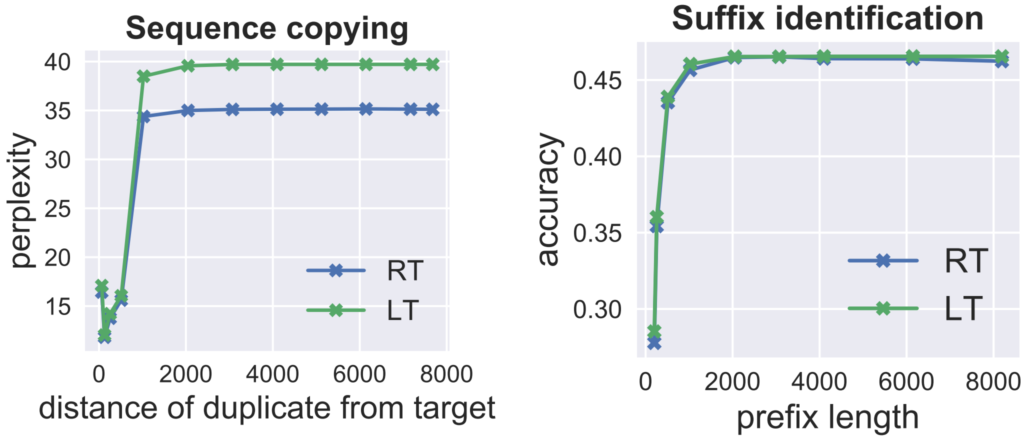


Figure 2.13: **Left:** Both models assign low perplexity if a duplicate sequence appears within previous 512 tokens. **Right:** Both models have almost identical performance on our suffix identification task. Adding context beyond 2K tokens does not improve performance of either sequence-level prediction task.

Sequence-level copying: As a sequence-level analogue to the token-copying analysis in the previous section, we examine both RT and LT's ability to memorize a sequence that occurs in the distant context. To test this ability, we copy the target sequence and paste it into different positions of the prefix. The left plot in Figure 2.13 shows that both models give a very low perplexity to the target sequence if its duplicate appears within the previous 512 tokens. However, **both models lose their ability to take advantage of the copied**

sequence if it occurs more than 2K tokens away. This confirms our previous discovery that sequence order is in general not encoded in the long-range context.

Prefix	... If the doctor's prophecy is correct... (∼700 tokens) ... "How far is it to his place?""Oh, a mile at least.We can have a cab.""A mile?
Gold suffix:	Then we shall see if there is any truth in what that swab of a doctor said ...
Negative 1:	If I can see Mr.Elberry to-day we may let you have a cheque ...
Negative 2:	It was not until he had signed and sent it off that the full significance of all...
Negative 3:	He hurried in, fearing that she might have taken some turn for the worse...
Negative 4:	look!!!"Her voice had fallen suddenly to a quivering whisper and she was...
Negative 5:	Again the Admiral burst out cheering."There remains, therefore...

Table 2.1: An example of suffix identification task, the full version of this example is included in Appendix A.5.

Suffix identification: To move beyond token-level experiments, we adopt a similar setting as the multiple choice task in SWAG [219]. Specifically, a prefix is paired with the ground-truth next 128 tokens (or *suffix*) as well as five randomly sampled sequences of length 128 that come from the same book and do not occur in the prefix or gold suffix. We constrain the prefix to end at a full stop and each candidate suffix to start from a new sentence so that the difference in perplexity is not due to ungrammaticality. An example is shown in Table 2.1. We construct 7K examples and compute the accuracy of both models at correctly choosing the correct suffix. The model makes a correct prediction when the gold suffix has lower perplexity than all other suffixes. As shown in the right plot of Figure 2.13, increasing prefix length beyond 2K does not improve suffix identification accuracy. Surprisingly, the LT and RT model have almost identical (and poor) performance on this task. While evaluating on the newly released LT checkpoint, the performance of LT is slightly worse, but the trend is similar. Adding context beyond 2K tokens does not keep improving the suffix identification accuracy. We direct reader to Appendix A.6 for more details. While RT is a significantly better LM in terms of token-level perplexity, it does not appear to be superior in terms of using long-range context to improve sequence prediction. Overall, both models often predict obviously wrong negative suffixes: the full version of Table 2.1 together with RT's perplexity score for each suffix is included in Appendix A.5.

Combined with our previous token-level analysis, we conclude that the distant context helps a subset of tokens in superficial ways; however, distant context is currently not helpful for sequence-level prediction tasks.

2.6 Related work

Our work in this chapter examines recent advances in efficient Transformer variants [173, 87, 29, 179, 82, 193, 207] that accept longer sequences than prior approaches. Longer effective context size is often achieved by sparse attention [28], recurrence [34], and cached memory [204, 145]. Our work is also related to methods that incorporate long context [194] as well as document-level tasks that inherently require modeling long-range context [220, 61, 221]. This section is also similar to other analysis of language models, especially for long-range context. [84] analyze the usage of long-term context of smaller LSTM LMs. [161] prove long-term context is not needed for HMM LM due to teacher forcing. [144] conduct an analysis exclusively for the Transformer-XL [34] model. [145] show that Compressive Transformer improves the performance of infrequent tokens. Our work also relates to that of [99], who investigate the impact of context for pretrained masked LMs. More recently, [139] also observe negligible benefits of long-term context; we step further in this direction by exploring larger models with more fine-grained analysis.

2.7 Conclusion

In this chapter, we perform a fine-grained analysis of the impact of long-range context to both token- and sequence-level improvements on two long-range Transformer language models, using the PG-19 dataset as a testbed. Our results suggest these models rarely take advantage of the long-term context, and when they do it is mostly in superficial ways (e.g, by copying rare tokens from far away). With the proliferation of research in increasing the input size of Transformer LMs, we hope that our research will lead to more meaningful progress on integrating discourse information into these models.

Limitation

This work was conducted in 2021 spring, in retrospect, this work may now seem limited given the long-range language models available nowadays. Recent advances such as more efficient attention mechanisms, Flash-attention, state-space models and other hardware acceleration techniques have led to models that can process much longer sequences (e.g., sequence of 100k tokens), making the Routing Transformer trained on 8k sequence a relatively short-range language model. Moreover, the new paradigm of fine-tuning language models to follow instructions have greatly expanded the task types that we can evaluate a language model. A simple test for instruction fine-tuned language model is the in-context fact retrieval. Due to limited model capability back in 2021, these tests were impossible to test back then.

CHAPTER 3

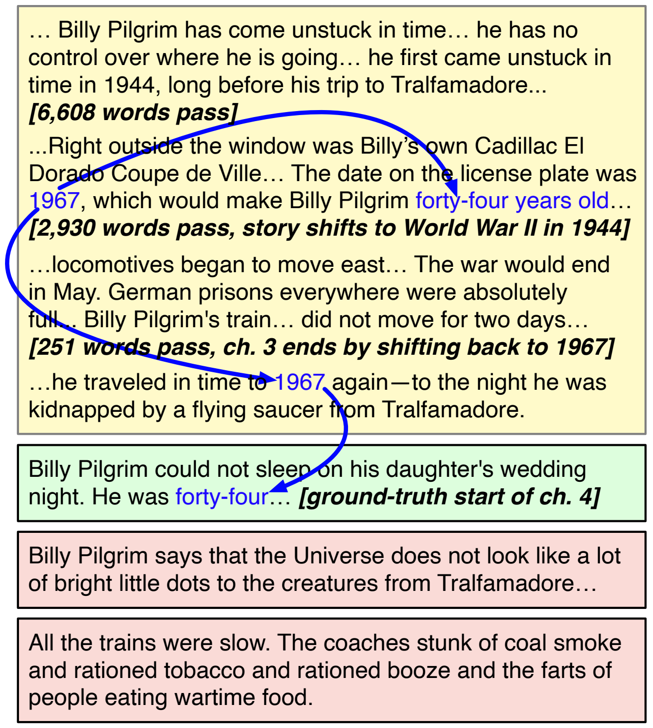
CHAPTERBREAK: A CHALLENGE DATASET FOR LONG-RANGE LANGUAGE MODELS

3.1 Introduction

Research on *long-range language models* (LRLMs) aims to process extremely long input sequences by making the base Transformer architecture more efficient (e.g., through sparse attention, recurrence, or cached memory). These modifications are commonly validated by training LRLMs on PG-19 [145], a long-document language modeling dataset, and demonstrating small perplexity decreases over shorter context models [148, 205]. However, recent analysis experiments [176, 140], such as those in the previous chapter, show that modern LRLMs rely mostly on local context (i.e., the immediately preceding 1-2K tokens) and are insensitive to various perturbations applied to more distant context.

In this chapter, we move beyond token-level perplexity by evaluating LRLMs on a task that requires a rich understanding of long-range dependencies. Our task is an instance of *suffix identification*, in which a language model is given a long input sequence (or *prefix*) and asked to disambiguate the next n-token segment from a set of hard negatives sampled from the same narrative. To succeed at this task, an LRLM should assign high probability to the ground-truth next segment and low probability to the negatives. To specifically test long-range dependencies, we restrict our prefixes to end at *chapter breaks* of a longer cohesive narrative (e.g., a novel).

We construct a challenge dataset, CHAPTERBREAK, by automatically detecting chapter boundaries within both held-out PG-19 documents (in-domain for pretrained LRLMs) and



… Billy Pilgrim has come unstuck in time… he has no control over where he is going… he first came unstuck in time in 1944, long before his trip to Tralfamadore...
[6,608 words pass]

...Right outside the window was Billy's own Cadillac El Dorado Coupe de Ville… The date on the license plate was 1967, which would make Billy Pilgrim forty-four years old…
[2,930 words pass, story shifts to World War II in 1944]

…locomotives began to move east… The war would end in May. German prisons everywhere were absolutely full… Billy Pilgrim's train… did not move for two days…
[251 words pass, ch. 3 ends by shifting back to 1967]

…he traveled in time to 1967 again—to the night he was kidnapped by a flying saucer from Tralfamadore.

Billy Pilgrim could not sleep on his daughter's wedding night. He was forty-four… ***[ground-truth start of ch. 4]***

Billy Pilgrim says that the Universe does not look like a lot of bright little dots to the creatures from Tralfamadore…

All the trains were slow. The coaches stunk of coal smoke and rationed tobacco and rationed booze and the farts of people eating wartime food.

Figure 3.1: An illustrative example of our suffix identification task from Kurt Vonnegut's *Slaughterhouse-Five*, in which an LRLM needs to make connective inferences across temporal and spatial shifts in a long prefix of the narrative to correctly disambiguate the start of the next chapter from negative examples.

works of fan fiction published on the Archive of Our Own (out of domain).[1] We perform a detailed analysis of the types of chapter transitions in our dataset and discover a high frequency of narrative shifts in point-of-view, location, and time, all of which require global narrative understanding over long input sequences. For example, Figure 3.1 contains a complex prefix in which the time-traveling Billy Pilgrim moves between World War II, 1960s suburban life, and an alien planet. Understanding the cliffhanger ending, in which the narrative abruptly switches from a wartime scene to a 1967 alien abduction, requires an LRLM to make connective inferences using details buried far back in the context (e.g., Billy's age in 1967).

[1]`https://archiveofourown.org`

We evaluate three LRLMs on CHAPTERBREAK, including BigBird [218], the Routing Transformer [148], and its local attention variant, all pretrained or fine-tuned on PG-19. Our experiments show that these LRLMs perform poorly at selecting the ground-truth suffix, regardless of the length of the input sequence. As an upper bound, we train a small RoBERTa-based segment-level language model on PG-19 and discover that it substantially outperforms all LRLMs on CHAPTERBREAK, which suggests that LRLMs have considerable room for improvement on this suffix identification task. Finally, we perform an analysis on the instances for which all models struggle to choose the correct suffix, which shows that shifts in location and events in focus are particularly challenging to disambiguate. Taken together, these results suggest that CHAPTERBREAK is a useful benchmark for future research into LRLMs.

3.2 The CHAPTERBREAK dataset

Authors often break long-form narratives into a sequence of discrete chapters to impose "an order and shape over events in time" [168]. Henry Fielding writes in his novel *Joseph Andrews* that the space between chapters is like "an Inn or Resting Place" for readers to reflect on the preceding chapter [48]. Chapters come in many flavors: for example, Murakami's *Kafka on the Shore* uses chapter breaks to alternate between parallel narratives focusing on the two protagonists, while cliffhanger endings such as the one in Figure 3.1 add suspense. Making sense of the complex narrative shifts associated with chapter transitions (e.g., changes in point-of-view, time, location, and theme) requires a deep understanding of the entire text. To maintain global narrative coherence, [125] show that human readers tend to reactivate memory about "backgrounded" information from the long-range context.

3.2.1 Task overview

Given that chapter transitions requires global context understanding, how can we turn this into a task to evaluate LRLMs? A simple approach is to evaluate the token-level perplexity of an LRLM only at chapter boundaries (i.e., on the first n tokens of each chapter); however, the vast majority of tokens can be predicted using just local context [176] under the teacher-forcing setup, which obscures an LRLM's usage of long-range context as we show in Section 3.3. We instead turn to the task of ***suffix identification***, which closely resembles existing datasets such as SWAG [219].

Each instance of our task is defined by a triplet $(c, s^+, s_i^- \in \mathbf{N})$, where c is a prefix sequence of up to 8K tokens that ends at a chapter break, s^+ is the gold suffix of length 128 tokens (i.e., the beginning of the next chapter), and s_i^- is a negative 128-token-long suffix from a set **N** of five future chapter beginnings sampled from the same narrative. We use a small number of negatives because it is time-consuming and resource-intensive to evaluate the probabilities of long sequences with LRLMs. In Appendix B.5, we show that in-book negatives are much harder than out-of-book negatives as they often contain the same named entities and rare tokens as the gold suffix. Thus, disambiguating the correct suffix requires a deep understanding of the context. All negatives are modified to begin with the same chapter index (e.g., if the gold suffix begins with "Chapter III", the chapter indices of all negatives is set to "Chapter III") to eliminate the effect found by [176] of language models memorizing chapter indices in long contexts. We then evaluate whether an LRLM assigns higher probability to the gold suffix $P(s^+|c)$ than to all negative suffixes $P(s_i^-|c)$.

3.2.2 Dataset overview

Where do we get these triplets from? We collect a dataset, CHAPTERBREAK, with two splits: CHAPTERBREAK$_{PG19}$, which contains 241 examples extracted from the PG-19 validation set [145], and CHAPTERBREAK$_{AO3}$, which contains 7,355 examples extracted from an online dump[2] of fanfiction posted on Archive of Our Own (AO3). We only collect examples from validation set as two baseline models in the later sections are trained on PG-19. We apply filtering to remove fanfiction works that are too short or not rated for general

[2]`https://archive.org/download/AO3_story_dump_continuing`

Category	**Definition**	**Pct.**
Events	Previous event ends and new event starts	76%
	Previous event continues into next chapter	24%
Actors	Change of perspective or character in focus	36%
	No change in POV or main character	64%
Locations	Change of location	68%
	No change in location	32%
Continuity	Discontinuous but chronological	29%
	Continuous	62%
	Analepsis	2%
	Parallel	6%

Table 3.1: Our human annotation on 300 chapter transitions randomly sampled from CHAPTERBREAK$_{AO3}$ shows the diversity and complexity of the dataset.

audiences. Each work contains on average 42K words and 21.5 chapters. We include more preprocessing details and statistics in Appendix B.1. Even though the CHAPTERBREAK$_{PG19}$ split is small, we include it because many LRLMs are pretrained on PG-19; the much larger CHAPTERBREAK$_{AO3}$ split is out-of-distribution for all models that we evaluate. To extract chapters in PG-19, we match for lines beginning with the string "chapter", while AO3 stories already have chapter-level metadata.

To get a better sense of our dataset, we perform a fine-grained annotation of 300 randomly-selected chapter transitions from CHAPTERBREAK$_{AO3}$. For each transition, we annotate any changes in the following four aspects: events, actors (characters in focus), locations, and continuity. To annotate continuity, we follow a simplified version of the scheme proposed by [73], which considers five categories: **continuous** (the next chapter occurs within a day of the previous chapter), **discontinuous** (the next chapter occurs more than a day after the previous chapter), **analepsis** (the next chapter is a "flashback" to an earlier point in the narrative), and **parallel** (the next chapter reverts to the time of a previous chapter, switching the character or event in focus). To validate our continuity annotations, we also annotate every chapter in *Pride and Prejudice* and obtain almost the same proportion of continuous transitions (67%) as the number reported by the expert annotation of [73] (72%). A more concrete annotation scheme is provided as follows:

- **Events** We define two subcategories based on whether (1) previous event ends in the previous chapter and new event starts in the new chapter, (2) old event does not end and continues into the next chapter.
- **Actors** We define two subcategories based on whether there is a shift in POV or main character in focus.
- **Location** We define two subcategories based on whether the location described in the prefix and in the new chapter is different.
- **Continuity** Following [73]'s work, we categorize the chapter transition by timeline continuity into four subcategories:
  - **Discontinuous but chronological**: Reusing the standard by [73], discontinuous represents a gap in time forward for more than one night.
  - **Continuous**: The time interval between chapters lasts for no more than one night.
  - **Analepsis**: Analepsis represents retrospective evocation of an event, or "flashback" to an earlier point in the narrative.
  - **Parallel**: This includes timeline reverting back to the time of any previous chapter, typically accompanied by switching character in focus or description of a separate set of events independent of the last chapter. This category is a collapse of "alternate phase", "parallel phase" and "simultaneous phase" introduced in [73].

The results, shown in Table 3.1, demonstrate that CHAPTERBREAK covers a diverse array of transitions, including many that require global narrative understanding.

	#Params	Seq Len	PPL$_{\text{PG19}}$	Acc$_{\text{PG19}}$	Acc$_{\text{AO3}}$
LT	516M	8K	76.8	25%	24%
RT	490M	8K	72.3	22%	24%
Bigbird	128M	4K	56.2	27%	26%
GPT-2	1.5B	1K	78.2	23%	24%
GPT-3	175B	2K	-	36%*	28%*
SuffixLM	87M	10K	-	**52%**	**41%**

Table 3.2: Summary of LRLMs (top), Transformer LMs (middle), and our SuffixLM (bottom). All models are trained or fine-tuned on PG-19 except for GPT-2. The third column shows the word-level perplexity of gold suffix in the PG-19 split. The last two columns show the suffix identification accuracy of each model on the two CHAPTERBREAK splits when evaluated at maximum input length. * indicates results are on a subset of CHAPTERBREAK.

3.3 Experiments

We evaluate three different long-range language models on CHAPTERBREAK and compare their results to those of standard Transformer language models as well as an upper bound directly trained for suffix prediction.

Language models: We evaluate three LRLMs pretrained on PG-19: the Local Transformer [148], Routing Transformer (RT) [148] which are the main focus of last section in this chapter, and BigBird [218]. The BigBird model is the decoder part of the released checkpoint fine-tuned with causal LM objective on 14k books of PG-19 for 100k steps. We also evaluate two standard Transformer language models, GPT-2 large [143] and GPT-3 [20].[3] We summarize these models in Table 3.2, more details about each model are included in Appendix B.2.

An upper bound directly trained for suffix identification: As authors often write stories that are intended to surprise readers, it is possible that many examples in CHAPTERBREAK are ambiguous by nature (i.e., the upper bound for suffix identification accuracy may not be 100%). To obtain a reasonable upper bound, we also train a model (SuffixLM) directly

[3]Due to OpenAI's API costs for GPT-3, we only evaluate in total a subset of 200 examples instead of the full dataset.

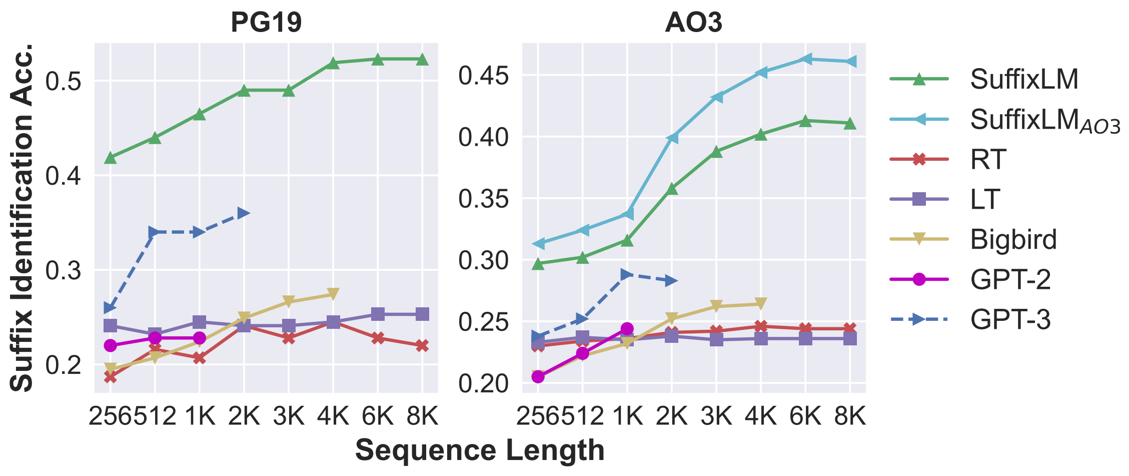


Figure 3.2: Suffix identification accuracy on both splits (PG-19 and AO3) of CHAPTER-BREAK is much lower for LRLMs than our SuffixLM upper bound.

on the suffix identification task by scaling up the sentence-level language model proposed by [72].Our SuffixLM can process up to 10K tokens, while the model of [72] supports only up to ten sentences. We divide an input sequence into multiple segments, each of which is embedded via the `[CLS]` vector of a small fine-tuned RoBERTa network [113]. Our SuffixLM then performs "language modeling" atop the dense `[CLS]` vectors, predicting the next segment representation given the representations of previous segments via contrastive predictive coding [186]. Our SuffixLM is closely related to the model in [2], but differs crucially by predicting the representation of next segment instead of summaries. Formally, our SuffixLM minimizes the following loss:

$$\mathcal{L}_i = -\log \frac{\exp(\hat{\mathbf{z}}_i{}^\top \mathbf{z}_i^+)}{\sum_{\mathbf{z}_i \in \{\mathbf{z}_i^+, \mathcal{Z}_i^-\}} \exp(\hat{\mathbf{z}}_i{}^\top \mathbf{z}_i)} \tag{3.1}$$

where $\hat{\mathbf{z}}_i$ is the predicted representation by SuffixLM, $\mathbf{z}_i^+$ is the gold suffix representation obtained from a small encoder (RoBERTa), and $\mathcal{Z}_i^-$ is the set of dense representations of the negatives. More details about our SuffixLM are included in Appendix B.3.

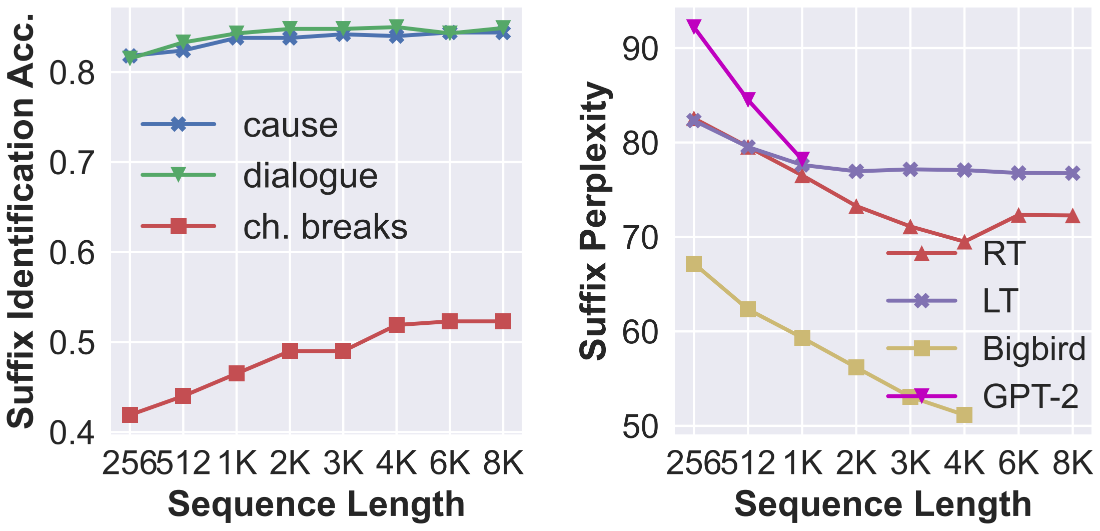


Figure 3.3: **Left:** Prefixes ending at chapter breaks benefit more from long-range context than other types of discourse boundaries. **Right:** Word-level perplexity of the gold suffix does not correlate to accuracy (e.g., GPT-2 has high perplexity but outperforms RT on suffix identification).

3.4 Results & Analysis

Overall, the results in Table 3.2 (rightmost two columns) confirm that all of the language models studied in this chapter struggle on CHAPTERBREAK, especially when compared to the SuffixLM upper bound, which outperforms the best LM by ∼25% absolute accuracy when evaluated on the entire PG-19 split. We describe other interesting results and analysis below:

Accuracy increases with longer prefixes: Figure 3.2 shows that as prefix sequence length increases, some LRLMs (e.g., LT) barely improve, while others show modest improvements (e.g., GPT-3 and fine-tuned BigBird). However, all LRLMs significantly underperform our SuffixLM upper bound, even when the SuffixLM is given prefixes that are only 256 tokens long. Additionally, SuffixLM's accuracy increases far more than those of LRLMs when increasing the prefix length (from 31% at prefix length of 256 to 46% at 8K on the AO3 split). We collected 13,682 fan-fictions posted on AO3 and fine-tuned our SuffixLM on subset of this dataset to be the model SuffixLM$_{AO3}$. More details about the filtered

AO3 works are included in Appendix B.1. This result suggests that the token-level LRLMs evaluated in our work are not taking full advantage of information in the long-range context to solve CHAPTERBREAK.

Perplexity does not always correlate with accuracy: Previous LRLM efforts use validation perplexity (e.g., on PG-19) to compare against other models. However, we show that perplexity is not by itself a predictor of suffix identification accuracy: As shown in Table 3.2, GPT-2 achieves higher accuracy than RT despite yielding a word-level perplexity of 78.2 on gold suffixes, compared to 72.3 for RT. As these models use different tokenizers, we normalize the subword-level perplexities to the word level as suggested by [145]. More details about this can be found in Appendix B.4. We advocate that future research on LRLMs includes evaluation on suffix identification tasks like CHAPTERBREAK, as perplexity alone does not reflect LRLMs' capabilities to model long-range dependencies.

Why chapter breaks over other discourse boundaries? Other discourse markers, including *cause* and *dialogue*, also often prompt human readers to reactivate memories of global context [3]. We create suffix identification datasets for these two discourse markers by string matching over corresponding cue phrases ('because', 'due to' for the *cause* subset and text within quotation marks for *dialogue*). Appendix B.1 contains more details about data for these two discourse markers.Figure 3.3 (left) shows that with prefixes of length 256 tokens, our SuffixLM is able to successfully disambiguate the correct suffixes for both discourse markers more than 80% of the time, while the accuracy is much lower at chapter boundaries. As the prefix length increases, accuracy only slightly increases for *cause* and *dialogue*, especially compared to the robust improvement at chapter boundaries.

Short-context Transformers are comparable to LRLMs: Our results show that GPT-2, despite its high perplexity on gold suffixes and short maximum sequence length (1024 tokens), achieves comparable performance to RT and LT on both splits. Meanwhile, GPT-3 achieves much higher performance on both CHAPTERBREAK at a sequence length of 2,048 tokens, and the increasing GPT-3 curve in Figure 3.2 is promising for future work scaling LMs to longer sequence lengths.

Limitations of our work: While we have used the SuffixLM as an upper bound in this chapter and demonstrated that it substantially outperforms LRLMs on CHAPTERBREAK, a more compelling comparison would include human performance on our task at varying prefix lengths, especially since some chapter transitions are specifically intended by their authors to be unpredictable. However, obtaining reliable human performance numbers is very difficult, as it requires in-depth comprehension of long narratives on the part of workers. Due to the time-consuming nature of this task and its high cognitive demand, it is not possible (within a reasonable budget) to use crowdsourcing, as ensuring that the annotators fully read the prefix instead of skimming or ignoring it is a major challenge. These issues also carry over to experiments performed with in-person subjects. As such, we leave a thorough human evaluation on CHAPTERBREAK to future work.

3.5 Related Work

Efficient context scaling Our work in this chapter depends heavily on recent advances in efficient Transformers [181] that process long sequences [145, 13, 218, 2, 148]. While most recent research on hardware aware optimizations such Flash-Attention [35] has successfully reduce memory consumption from $O(L^2)$ to $O(L)$, the previously predominant approaches to handling long sequences have been leveraging sparse attention [28] variants, employing recurrence mechanism such as Transformer-XL [34, 71] and adding cache or memory [204, 205], and using retrieval techniques to retrieve relevant context from the history [205] Other methods have also been proposed [162, 82, 57, 170, 216, 16, 59, 108] to make the operation on long sequences more time/memory efficient. However, nowadays (as of 2023), the common practice has been continually pre-training a short-range language model on longer sequences [137]. These short-range language models tend to utilize position encoding methods capable of extrapolating to unseen length to some extent, such as AliBi [140] and rotary position embeddings (RoPE) [172]. Recent research, such as Position interpolation [24] and NTK-aware [49] method have been applied to RoPE to make the continual pre-training on long sequences more efficient and effective.

Retrieval, sparse attention, and effective long-range context modeling The retrieval-based approach [205] involves encoding long context into separate representations and retrieving relevant information for predicting future tokens. This method serves as a common solution for overcoming the long sequence bottleneck. Recent work [211] demonstrates the improved quality and faster inference speed of models by incorporating a retrieval component. In this approach, retrieval module can be thought of as applying sparse attention to the long context, with the salient part receiving the highest attention. One related finding uncovers the induction head circuits [129] in the Transformer, where the pair of induction attention heads can be responsible for copying and completing patterns/trigger pairs [147] within the context. While models with induction circuits should ideally be capable of copying patterns from *any* position in the context, tests like passkey retrieval [122] demonstrate that these models still face challenges when dealing with in-context copying tasks, likely due to the attention competition between irrelevant nearby tokens and relevant far-away tokens. An earlier study [147] on language modeling suggests that substantial information is present and distributed thinly across long-range context. As such, building a comprehensive understanding of the long-range context can be of greater importance than simple in-context retrieval. This can be achieved by collecting aggregated statistics of the long context or applying more intricate methods that form a gestalt of in-context retrieval processes through multi-layer contextualization.

Long context evaluation Besides perplexity, many extrinsic tasks for evaluating long-range language models were developed recently , such as long-form QA [46, 135], document-level summarization [94, 70], and machine translation [110]. More recently, benchmarks [159, 158, 4, 11] covering multiple domains and tasks have been introduced, while [180] propose multimodal long sequence tasks. Another type of evaluation worthy of highlight is the in-context retrieval task which has been tested in various formats and domains, such as passkey retrieval [122], which we discuss in detail in section 3.6.2. The sequence-copying task proposed in the previous chapter can also be thought as a preliminary version of such test. However, more tasks need to be developed to evaluate not only the accuracy of extracting short relevant excerpts from the long context, but also the capability of making non-trivial inference, reasoning and synthesizing information, understanding the implicature within the context, etc.

Discourse segmentation In this chapter, we use chapter break as a heuristic for detecting coherence breaks. This approach is closely related to research on discourse segmentation, where extended sequences are divided into smaller topically coherent discourse segments via either linguistic cues [189, 136] or automated methods [119, 44, 198]. In this chapter, we divide long sequences into prefixes and suffixes using chapter boundaries. We segment the long prefixes using sentence boundaries and an upper length limit, however, without taking into account the discourse structures in the prefixes. Thus, an interesting future analysis involves probing the segment representations produced by SuffixLM to evaluate their accuracy in predicting various topics, functions, eventualities, and discourse relations [199].

Limitation

We provide the evaluation of language models across various model sizes, architectures, context sizes, and training objectives. However, a better result that complements our existing results is the human performance on ChapterBreak. While the SuffixLM can serve as a soft upper-bound for the attainable performance on ChapterBreak, it still may differ than human performance by large margins. The reason why we did not provide human performance is because (1) it is hard to properly control human behavior when reading long sequences is involved, and (2) since we also require the human performance at various sequence length, relegating the performance at different context sizes to different persons can be very costly. We believe this is a promising research direction for both computer science and cognitive science in the future. Another limitation of this chapter lies in the prefix length. Recent progress in the efficient context scaling methods allow processing up to hundreds of thousands of tokens in one shot with some proprietary models. The prefix of at most 8K tokens in our dataset can be a bit short to recent large language models. Thus, it can be promising to sample longer prefix from books and other narratives, and construct a longer version of CHAPTERBREAK in the future.

Ethical Considerations

CHAPTERBREAK is constructed from two sources: public domain books published prior to 1919 (from the held-out set of PG-19) and works of fanfiction extracted from an online dump of stories posted on Archive of Our Own (AO3). We refer readers to [145] for more details about PG-19. For AO3, we apply multiple filters to obtain long fanfiction stories rated as suitable for "General Audiences". We refer readers to Appendix B.1 for more preprocessing details. More generally, this work focuses on long-range language models, which could potentially be misused to generate offensive output. However, the main purpose of this chapter is to present a dataset which provides a better evaluation of the discourse-level capabilities of such models.

3.6 Long-range context modeling in the era of "large language models"

In Chapter 2 and the preceding sections of the current chapter, we have examined models primarily consisting of millions of parameters and trained on constrained domains such as PG-19. Recent advancements, exemplified by Flash-attention [35] and more powerful hardwares, coupled with open-source and proprietary efforts at scaling model and context window size have led to the burgeoning of "large language models" (LLMs). These models have more parameters, are trained on carefully preprocessed web-scale data across diverse domains, and some incorporate instruction fine-tuning and RLHF [132] to align with human intents. In this section, we provide additional empirical evaluations of more recent "large

Prompt for evaluating OpenAI models on ChapterBreak
You are given a snippet and six possible continuations of the snippet. Please select the most natural and coherent continuation of the given snippet. Please provide a letter A, B, C, D, E, F in a separate line. Only one of the options is correct. [Snippet] {prefix} [Option A] {suffix A} [Option B] {suffix B} ... [Answer] (select from A, B, C, D, E, F):

Table 3.3: Prompts used to evaluate instruction fine-tuned large language models (e.g. GPT-4 and GPT-3.5-Turbo) on CHAPTERBREAK.

language models", and engage in a general discussion on the evaluation of long-range context modeling.

3.6.1 Evaluating large language models on CHAPTERBREAK

In this section, we evaluate the performance of recent large language models on CHAPTER-BREAK. These results stand alone as a separate section due to the lack of direct comparability between these models and those we examined before. Despite this, we believe it is still beneficial to evaluate these recent LLMs to understand the limit of frontier language models. Our results indicate that these models, though scaling up across multiple factors, still struggle with some examples in CHAPTERBREAK, and under-perform our segment-level language model SuffixLM.

3.6.1.1 Models & Evaluation setup

For open-source models, we employ the same approach as in the previous sections, i.e., computing the length-normalized probability of the gold suffix and negatives. For proprietary models, since log probabilities of input tokens were not available with OpenAI API by the time we evaluate these models, we frame each example as a multiple-choice question and instruct the model to select the best suffix. We evaluate five open-source language models, including four Transformer-based models (Mistral-1-7B [76], Mistral-1-7B-instruct, [4][5] LLaMA-2-7B [183], and Cerebras-BTLM-8K [39]), and a selective state-space model (Mamba-2.8B [56] trained on SlimPajama [163]). For all models, we use A100 80GB GPUs for evaluation and evaluate with bfloat16 data type, with the exception of Cerebras-BTLM-8K when evaluating on 8K-token sequences, where we load the model with 4-bit due to memory constraints. We evaluate two proprietary models, `gpt-4` and `gpt-3.5-turbo-16k`, with the prompt shown in Table 3.3.

3.6.1.2 Results & Analysis

CHAPTERBREAK is still challenging for open-source LLMs. As shown in Figure 3.4, in contrast to models evaluated in the previous sections, recent open-source LLMs exhibit increasing trends with the increase of prefix length. Recall that models we have evaluated in section 3.4 demonstrate either zero or only marginal gains when prefix length is extended. Although recent open-source models demonstrate the notable improvement, they still lag behind proprietary models and SuffixLM by large margins, suggesting less effective utilization of long-range context.

GPT-4 outperforms SuffixLM but shows severe degradation when sequence is long. For closed-source models, GPT-3.5-Turbo consistently underperforms SuffixLM, while

[4]`https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1`

[5]While this version is instruction fine-tuned, we still evaluate with the length-normalized probability for consistency with other models.

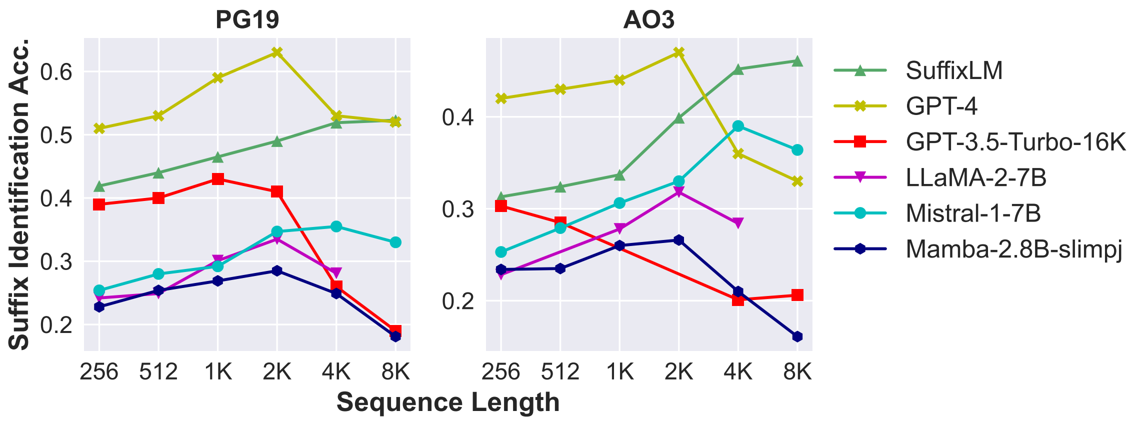


Figure 3.4: Suffix identification accuracy of recent open-source and proprietary language models on ChapterBreak. SuffixLM outperforms open-source language models (both Transformer-based and state-space models) despite having 87M parameters. GPT-4 outperforms SuffixLM when prefix length is short, and displays degraded performance as prefix length increases.

GPT-4 is the only model that outperforms SuffixLM, though improvement is not consistent. Moreover, both models exhibit abrupt degradation when the prefix extends beyond 2K tokens. This suggests limited base LM capabilities in identifying the correct continuation given long prefix, and/or potential inadequacy of alignment process for tasks requiring reasoning over long-range context. Indeed, obtaining high quality human preference data on long sequences can be costly and challenging, and is of an interesting direction for further exploration. Despite being the only model that outperforms SuffixLM, GPT-4's peak performance remains less satisfactory (less than 50% on AO3). Considering the performance of open-source models, there is still significant room for improvement in recent LLMs on ChapterBreak.

Instruction fine-tuned Mistral exhibits a slight performance decline but the increasing trend persists. Among all open-source models that we evaluated, Mistral-1-7B achieves the best accuracy and displays the steepest increase in accuracy with extended prefix length. In fact, it is the only model that shows increased suffix identification accuracy when prefix becomes longer than 2K tokens. Will instruction fine-tuning erase the capability of long-range context modeling and lead to declined accuracy as demonstrated in the proprietary models? We conduct another experiment with the "chat" version of Mistral (Mistral-1-7B-instruct). While this model can follow instructions, we still evaluate it with the length-normalized log probability method for a consistent comparison with Mistral-1-7B. Figure 3.5 shows that instruction fine-tuning marginally degrades performance on CHAPTERBREAK, but the overall upward trend is maintained, suggesting the instruction tuning phase does not significantly change the underlying contribution from long-range context.

Mamba trails behind Transformer-based models on CHAPTERBREAK. Structured state space models are popular new architectures for language modeling. The recurrent structure in state space models (SSMs) theoretically allows for the processing of infinitely long sequence; adding in the selection mechanism, the recent best performant SSM, the Mamba model, demonstrates comparable performance to Transformer-based models on relatively large scale training. However, our results show that the performance of Mamba-2.8B-slimpj increases with the extension of prefix length only up to a certain context size, and then decreases. As Mamba is significantly smaller than other open-source large language models, to ablate the effect of model size, we also evaluate Cerebras-BTLM-3B [39], a model that shares the same training data and model size with Mamba ($\sim$3B). As shown in Figure 3.6, Mamba-2.8B-slimpj consistently under-performs its Transformer counterpart. Nevertheless, it remains unclear if the gap is primarily due to the architectural difference or simply the differences of optimization process. Further experiments are necessary to fully understand the cause of the performance gap.

Evaluation on passkey retrieval test. In-context retrieval, such as passkey retrieval [122], is a common evaluation for long context modeling. To understand how retrieval-based evaluation relates to or diverges from CHAPTERBREAK, we also evaluate the open-source models on passkey retrieval test. Following previous works [122, 184], we insert a short sequence containing the passkey information (*"The pass key is {7-digit passkey}"*) into a random position in a long sequence. The input sequence consists of distraction alphanumeric

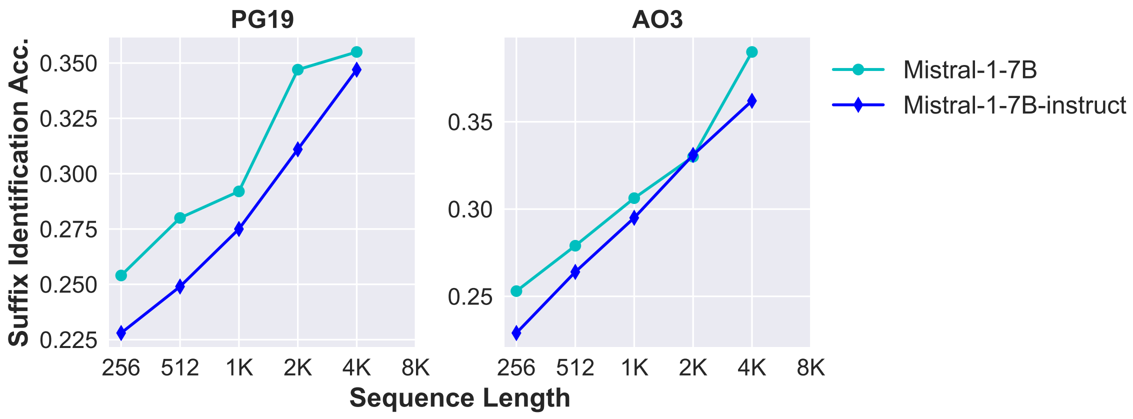


Figure 3.5: Instruction fine-tuning leads to slight degradation in suffix identification accuracy, but the upward trend with the increase of prefix length persists.

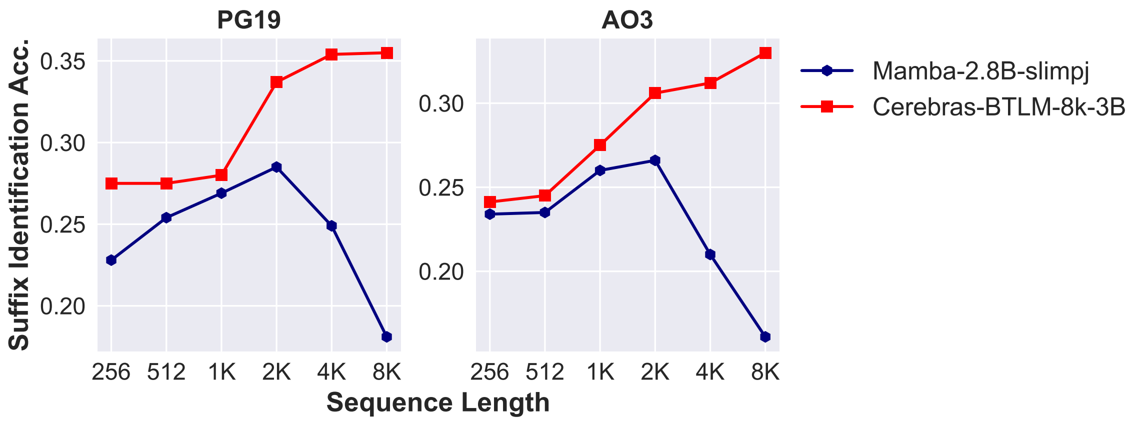


Figure 3.6: Comparison of Transformer-based model (Cerebras-BTLM-8k-3B) and state-space model (Mamba-2.8B-slimpj) on ChapterBreak. Both models are trained on SlimPJ dataset, and are of similar sizes.

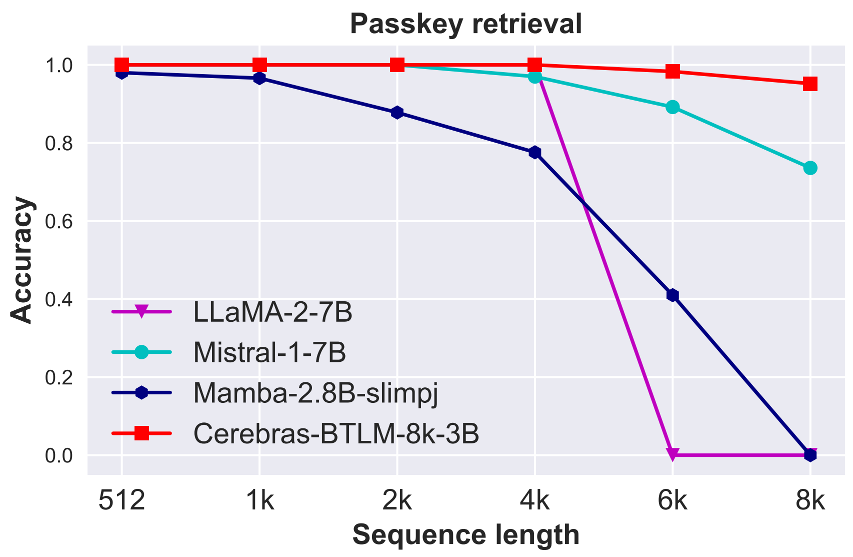


Figure 3.7: Comparison of open-source models on the passkey retrieval task. Model performance on passkey retrieval does not correlate with performance on CHAPTERBREAK.

strings and ends with *"The pass key is"*, which prompts the models to generate the passkey inserted in the context. We evaluate the exact match accuracy between the generated passkey and gold passkey while varying the sequence length.

As shown in Figure 3.7, all models exhibit varying degrees of accuracy degradation, with the LLaMA-2-7B completely fails the retrieval when sequence length is longer than its pre-training window size. On the other hand, Mistral-7B is able to handle the retrieval better due to its context size. Cerebras-BTLM-3B also undergoes training on 8K tokens and demonstrates the smallest degradation when increasing sequence length. Lastly, while Mamba demonstrates impressive performance in selective copying task under toy settings [56], the Mamba-2.8B-slimpj model exhibits significant degradation for sequences longer than 2K tokens, suggesting further room to improve the selection mechanism in current model.

Passkey retrieval performance does not correlate with CHAPTERBREAK. As shown in Figure 3.7, Transformer-based models demonstrate near-perfect passkey retrieval when the sequence is shorter than 4k token. However, these models underperform SuffixLM by large margin on CHAPTERBREAK regardless of the sequence length. Moreover, the ranking of these models in passkey retrieval test does not correlate with that in the ChapterBreak evaluation. For instance, Mistral underperforms BTLM in passkey retrieval when the sequence is long, whereas it either outperforms or achieves comparable results to BTLM in the ChapterBreak evaluation. The disparity in model comparisons indicates that the two tasks evaluate different aspects of context modeling. Passkey retrieval, a retrieval-based evaluation, and ChapterBreak, a prediction-based evaluation, require different context modeling capabilities. Evaluating the model on both fronts can offer complementary insights into its overall ability to handle diverse contextual tasks.

3.6.2 General discussion on the evaluation of long-range language models

Token-level evaluation Chapter 2 section 2.3 has challenged the suitability of perplexity alone for evaluating long-range language models; here we discuss a few followup works that echo our observation and two recommended token-level evaluation setups. Consistent with our findings, Perceiver AR [59] also demonstrates that gains from having context longer than 2K tokens are negligible on PG-19; Memorizing Transformer [205] aligns with our findings in section 2.4 that when having longer context, a small subset of tokens obtains significant gains, which are offset by other tokens to which the model has degraded predictability.

In additional to these empirical evidence, other operational factors also make perplexity no longer a suitable intrinsic metric. As discussed in section 2.3, without evaluating at various context sizes, the gains in perplexity can be misleadingly attributed to the incorporation of long-range context, when in fact, they are the consequence of better modeling of local short-range context. Employing an improper evaluation setup, e.g., non-overlapping evaluation, may also falsely suggest drop in perplexity, when the true cause is the reduction of “early token curse” [138]. A more suitable approach to evaluating language model is thus the sliding window method – fixing context window of size c and sliding by x ($x < c$) tokens every step, allowing the evaluated x tokens to have sufficiently long prefix of size ($c - x$). Another potentially helpful token-level evaluation practice is to fit the context scaling power law function. According to a recent study [209], the loss l of a token can be modeled as a function of its position t in the context: $l = (\frac{\alpha}{t})^{\beta} + \gamma$, where α, β, γ are power law parameters. The tangent of the power law function at various positions is indicative of the utilization of long-range context (counting from the beginning of the sequence) at those points, therefore it can be a more comprehensive evaluation than sliding window approach with a single fixed context size.

Sequence-level evaluation Besides the sequence-copying and in-book suffix identification task introduced in section 2.5 and enhanced in Chapter 3, recent works have advocated various sequence-level evaluations. One common task is the passkey retrieval test [122], wherein the model needs to extract a key from long distracting text. A similar in-context retrieval task, also known as "needle in the hay" test [78] involves inserting a sentence into a long narrative and evaluating the accuracy recalling the information of the inserted sentence at various insertion positions in the context. More recently, Anthropic has adopted this method to evaluate their models with context size of 200k tokens [7]. Similar kind of retrieval task is also systematically examined by a recent study [109] as a real natural language task of multi-document QA. An interesting phenomenon of "lost in the middle" suggests that models can retrieve facts more accurately when they appear in the beginning and end of the sequence compared to the middle. While further evidence is necessary, this phenomenon can potentially be explained by the recently discovered attention sink phenomenon [208], which states that tokens at the beginning of the sequence receive larger attention weights.

While in-context retrieval is a promising method to stress test a model's maximum effective context size, it restricts the evaluation to block retrieval, which can be better tackled if a retrieval module is augmented to the langauge model. An alternative and still under-developed evaluation setup demands comprehensive understanding of long-range context. This includes synthetic tasks such as parity prediction [228] and variable assignment [5] as well as real natural language tasks like CHAPTERBREAK and the long document QA task QuALITY [135], which we dive into details in the next chapter.

Overall, both in-context retrieval and long-range understanding are indispensable for a thorough evaluation of long-range context modeling. In the latter part of 2023, several benchmarks focused on long-context modeling have been introduced [11, 4]. Despite these advancements, numerous challenges persist within this domain, including issues related to data contamination [40] and the need for large-scale, high-quality human annotation [105] and evaluation. We conjecture that human-machine collaborative evaluation will be particularly helpful for evaluating long-context LMs, and will discuss as a future direction in section 6.2.

CHAPTER 4

PEARL: PROMPTING LARGE LANGUAGE MODELS TO PLAN AND EXECUTE ACTIONS OVER LONG DOCUMENTS

4.1 Introduction

Performing complex reasoning over long input documents often requires forming high-level abstractions of the text (e.g., plots and themes in a narrative) and then conducting a variety of inferences on top of those abstractions [53]. Consider the following question about the story “Breakaway” from the QuaLITY dataset [134]:

> *What part of the final scene best connects to the story’s opening conversation?*

To answer this question, we need to gather and synthesize information from across the story, which motivates decomposing the question into a *plan of actions*, as in:

1. Identify all participants in initial conversation.
2. Summarize the initial conversation.
3. Summarize events and themes of final scene.
4. Summarize roles of conversation participants in final scene.
5. Identify and rank connections between conversation and final scene.

Each action in the above plan varies in complexity, from simple lookup-style actions (Step 1) to more challenging query-focused summarization (Steps 2-4) and conceptual linking (Step 5) actions that require deep narrative understanding.

Given the rapidly advancing capabilities of large language models (LLMs), how can we use them to answer questions like these? While we could directly prompt LLMs to generate the answer, prior work on simpler reasoning-based tasks shows that this method is inferior

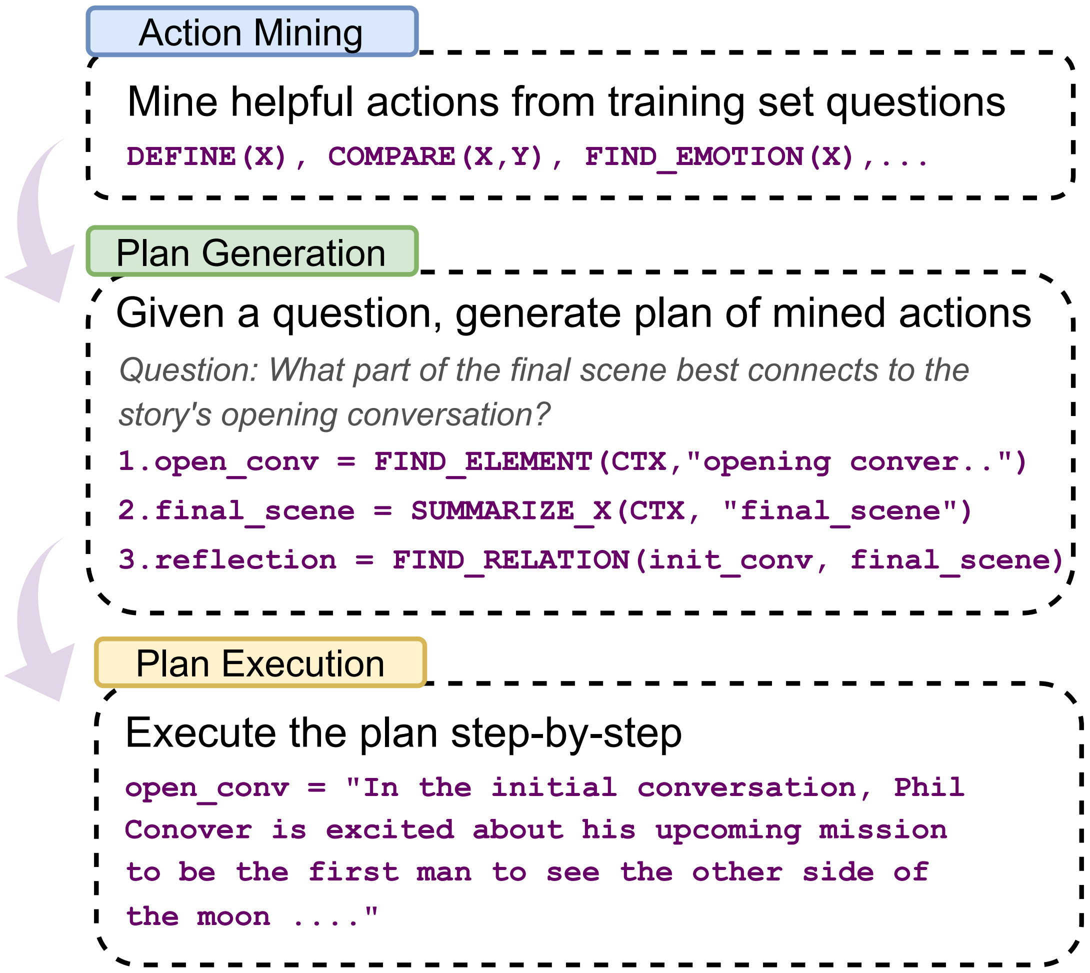

Action Mining
Mine helpful actions from training set questions
DEFINE(X), COMPARE(X,Y), FIND_EMOTION(X),...
Plan Generation
Given a question, generate plan of mined actions
Question: What part of the final scene best connects to the story's opening conversation?
1.open_conv = FIND_ELEMENT(CTX,"opening conver..")
2.final_scene = SUMMARIZE_X(CTX, "final_scene")
3.reflection = FIND_RELATION(init_conv, final_scene)
Plan Execution
Execute the plan step-by-step
open_conv = "In the initial conversation, Phil Conover is excited about his upcoming mission to be the first man to see the other side of the moon ...."

Figure 4.1: High-level overview of our framework PEARL. Each stage in PEARL is achieved via zero-shot or few-shot prompting of an LLM (in our work, GPT-4). We also provide `example outputs` from each stage.

to chain-of-thought prompting [200], which encourages the LLM to provide step-by-step explanations and intermediate outputs before producing the answer. Unfortunately, CoT is not well-suited for tasks involving complex reasoning over long input documents, as both the decomposition of the original question and the intermediate outputs of each step are non-trivial to obtain, as in the above example.

Given the difficulty of obtaining plans and intermediate explanations for long documents, one potential solution is to delegate this task to smaller *executable* modules instead of forcing the LLM to come up with all of them at once. In this work, we introduce PEARL,

Prompting Methods	Explicit plan	Iterative prompting	Does not rely on external tools	Long documents
Chain-of-Thought [200]	✗	✗	✓	✗
Program-of-Thought [26]	✗	✗	✗	✗
Self-Ask [141]	✗	✓	✗	✗
Toolformer [153]	✗	✗	✗	✗
ReAct [213]	✗	✓	✗	✗
Plan-and-Solve [192]	✓	✗	✓	✗
PEARL (*this work*)	✓	✓	✓	✓

Table 4.1: Comparison of PEARL to other recently-proposed prompting techniques. PEARL is the only one designed for and evaluated on tasks that require complex reasoning over long documents.

a framework that combines **P**lanning with **E**xecutable **A**ctions for **R**easoning over **L**ong documents. Each stage of PEARL — action mining, plan decomposition, and plan execution — is implemented by applying zero-shot or few-shot prompting to an LLM. The stages (Figure 4.1) can concisely be described as follows:

1. **Action mining:** An LLM is prompted to come up with simple actions that can help solve questions from an input training dataset. Unlike predefined "toolboxes" in methods such as Toolformer [153] or ReACT [213], the action set in PEARL is also generated by an LLM.

2. **Plan generation:** Given an input test question, an LLM generates an executable plan consisting of a series of actions selected from the action set produced in the previous stage. The plan is formatted as a simple program in which the execution result of one action can serve as an argument to future actions, which enables complex composition.

3. **Plan execution:** The LLM executes the plan action-by-action via a prompt template that includes an action and the long-form input document. Note that this is the only stage that includes the document, as the other stages operate over just questions.

We demonstrate PEARL's effectiveness on a challenging subset of QuALITY [134], a reading comprehension dataset that contains questions about long-form articles. While

QuALITY is originally a multiple-choice dataset, we reformulate it into a generation task: given a question and an article, an LLM is asked to generate a free-form answer. As a proxy for measuring answer correctness, we adopt a similar approach to [190] by asking the LLM to map its generated answer to one of the multiple choice options, which allows us to compute its accuracy.

Prompting LLMs with PEARL yields more accurate and comprehensive answers than those generated by directly prompting the LLM to answer the question, particularly for questions that require reasoning over the full long document. This result is particularly impressive given the potential for error propagation in the PEARL framework: as each stage is implemented via an LLM, errors in plan formulation or execution can significantly affect the output answer. To further verify the integrity of the plans, we perform human evaluation by asking annotators to provide feedback and ratings; annotators generally find the plans to be reasonable, although a small percentage contain unnecessary actions or omit critical actions. Overall, we hope PEARL further opens the door towards using LLMs for complex reasoning over long documents.

4.2 Related work

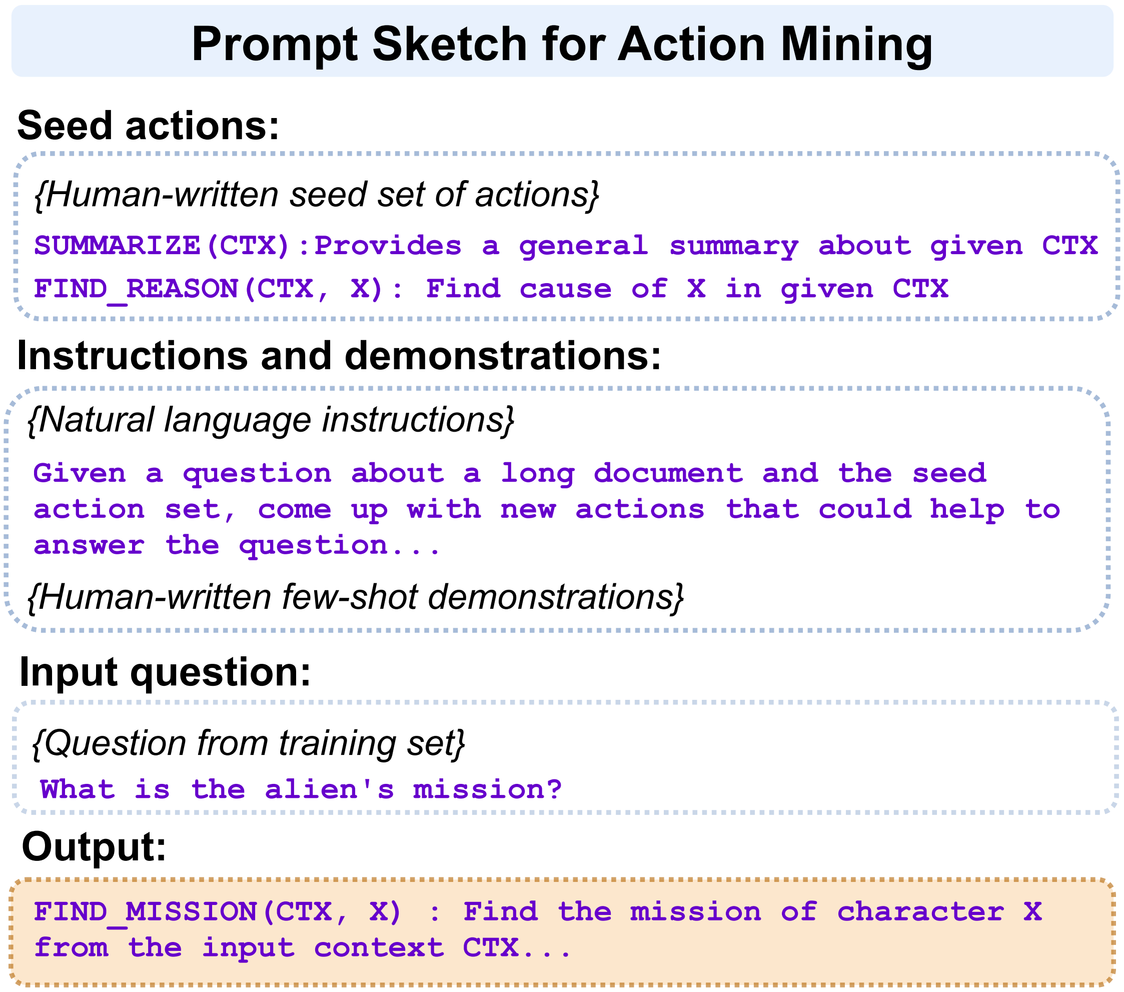
Our work builds on recent LLM prompting research and also connects to work on reasoning over long documents. Before describing PEARL, we first survey related papers to contextualize our work within this fast-moving field. **Prompting methods:** Recently, the capabilities of large language models [19, 224, 182] have significantly increased as a result of learning from instructions or feedback [169, 132, 30] to better align their outputs to human preferences. When provided with well-crafted prompts, such as chain-of-thought [200] explanations, these state-of-the-art models exhibit impressive reasoning abilities. A plethora of new prompting techniques (Table 4.1) has been recently introduced to unlock more capabilities of LLMs via leveraging exteral tools [26, 153, 115], problem decomposition [141, 42, 85, 213], self-reflection and self-refinement [67, 165, 118, 86], planning [212, 192, 114], and other techniques [217, 195, 227].

Reasoning over long documents: Large language models have showcased remarkable reasoning capabilities [68], including mathematical reasoning [31], commonsense reasoning [177], and symbolic reasoning [128]. Most of these tasks do not involve long context inputs, and thus they are able to benefit from few-shot in-context CoT prompting. In this chapter, we primarily focus on tasks that contain long input contexts [88, 36, 160, 174], specifically generative question answering based on long input articles. To address the absence of reliable evaluation for long-form QA [92], [167] proposes automatic metrics for evaluating the correctness of the answer, whereas in this work, we use LLM-based evaluation by taking advantage of the multiple-choice setup of existing QA dataset. Prior to the shift to prompting-based methods, approaches including contrastive learning-based sequence-level objectives [22], iterative hierarchical attention [175], and joint modeling of machine reading and answer generation [171] have been employed to enhance long-context question answering.

4.3 PEARL: Planning and Executing Actions for Reasoning over Long Documents

We are interested in using LLMs to solve tasks that require complex reasoning over long documents.[1] In this chapter, we focus on the task of answering questions about long-form narratives. Most prompting strategies that aim to improve the reasoning abilities of LLMs (e.g., CoT) are not applicable to this task due to the length and complexity of the input document. In this section, we specify our PEARL framework, which consists of three LLM-implemented stages that mine actions from a training corpus, formulate plans to answer held-out questions, and then execute the resulting plans to obtain answers.

[1]As there is no consensus on what is "long", we consider it to mean documents of several thousands of tokens in length.



Prompt Sketch for Action Mining

Seed actions:

{Human-written seed set of actions}

```
SUMMARIZE(CTX):Provides a general summary about given CTX
FIND_REASON(CTX, X): Find cause of X in given CTX
```

Instructions and demonstrations:

{Natural language instructions}

```
Given a question about a long document and the seed
action set, come up with new actions that could help to
answer the question...
```

{Human-written few-shot demonstrations}

Input question:

{Question from training set}

```
What is the alien's mission?
```

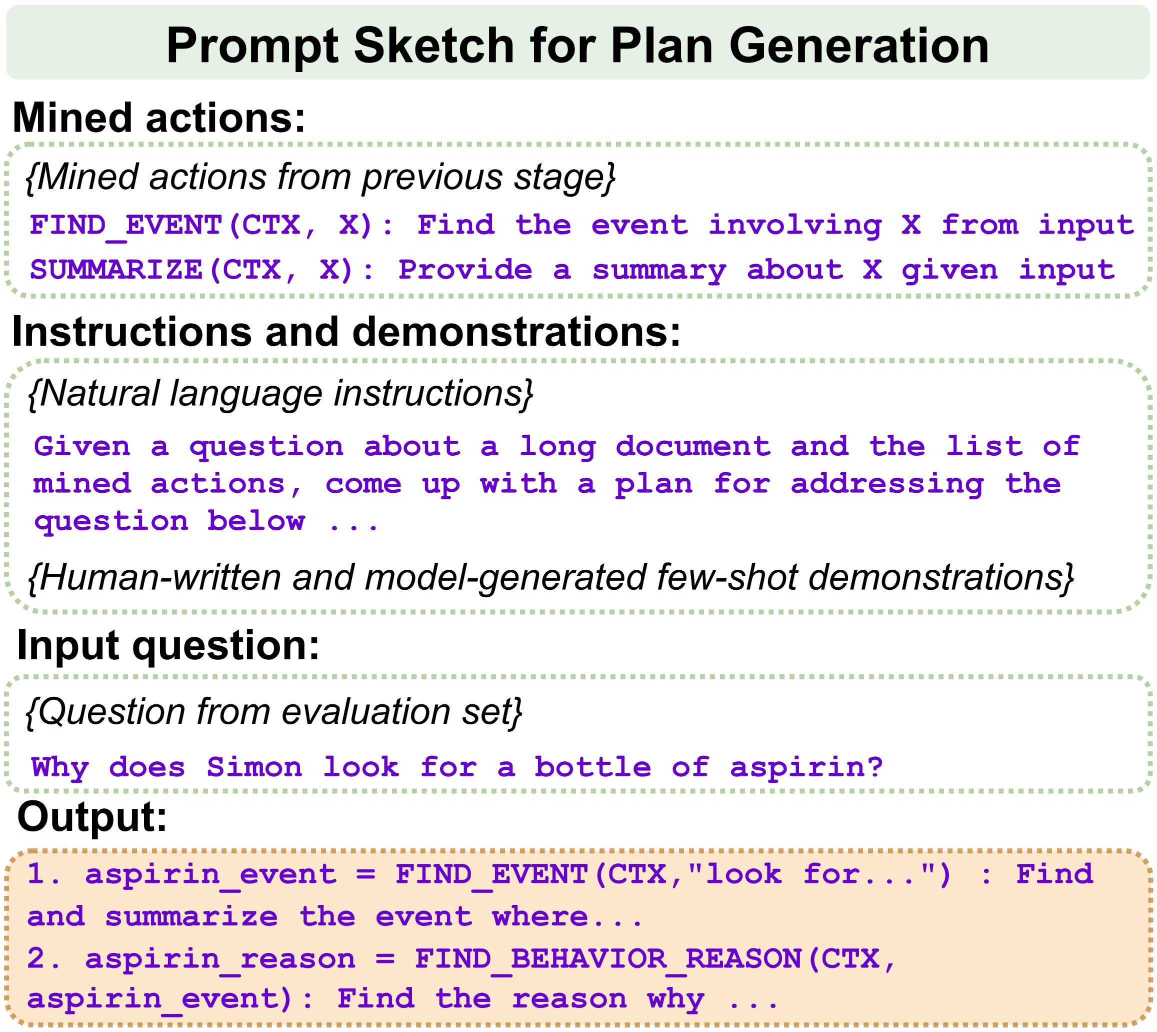
Output:

```
FIND_MISSION(CTX, X) : Find the mission of character X
from the input context CTX...
```

Figure 4.2: Prompt sketch for action mining. It comprises human-written seed actions set and instructions, as well as question for which LLM will extract action(s) from. Finally, we also present an example mined action. More details can be found in the Appendix C.6.

4.3.1 Action mining

In many prior prompting techniques such as ReACT and Toolformer, the LLM is able to query external APIs (e.g., Wikipedia search or a calculator) to solve a given task. Unlike these works, which assume a predefined action space, PEARL mines actions directly from data of similar distribution (in our case, training set questions of QuALITY). As shown by prior research [53], answering complex queries over long documents requires specific reasoning techniques; as further evidence, [210] demonstrate the presence of various discourse structures in good answers to long-form questions on Reddit. Learning dataset-



Prompt Sketch for Plan Generation

Mined actions:

{Mined actions from previous stage}

```
FIND_EVENT(CTX, X): Find the event involving X from input
SUMMARIZE(CTX, X): Provide a summary about X given input
```

Instructions and demonstrations:

{Natural language instructions}

```
Given a question about a long document and the list of
mined actions, come up with a plan for addressing the
question below ...
```

{Human-written and model-generated few-shot demonstrations}

Input question:

{Question from evaluation set}

```
Why does Simon look for a bottle of aspirin?
```

Output:

```
1. aspirin_event = FIND_EVENT(CTX,"look for...") : Find
and summarize the event where...
2. aspirin_reason = FIND_BEHAVIOR_REASON(CTX,
aspirin_event): Find the reason why ...
```

Figure 4.3: Prompt sketch for plan generation. In the prompt, we include the list of actions mined from previous stage in-context, natural language detailing the task, and few-shot examples guiding the plan generation.

specific actions enables PEARL to scale to different domains and tasks, as user queries may differ considerably in terms of complexity. Moreover, mining actions from training set can reduce human efforts in designing new actions. In this work, we define an "action" as a basic unit for long document reasoning. To obtain these actions, we first manually create a small set of *seed* actions to use as demonstrations.[2] Then, as shown in Figure 4.2, given an example question, we feed it along with the seed actions and instructions to the LLM to

[2]See prompt for QuALITY action mining in Appendix C.6

generate more task-specific actions. Each `ACTION` is formatted as a programmatic function with input arguments and is followed by a *model-generated function definition in natural language*. Below is an example action generated by the LLM:

`ANALYZE(CTX, X, Y)` *# Analyze the relationship, attitude, or feelings between X and Y given the input context CTX*

After a full pass over example questions in the training data, we obtain a final set of actions and their corresponding definitions which are then incorporated into the prompt of the next stage after model-based filtering and simplification (more details about filtering in Section 4.4.1).

4.3.2 Plan generation

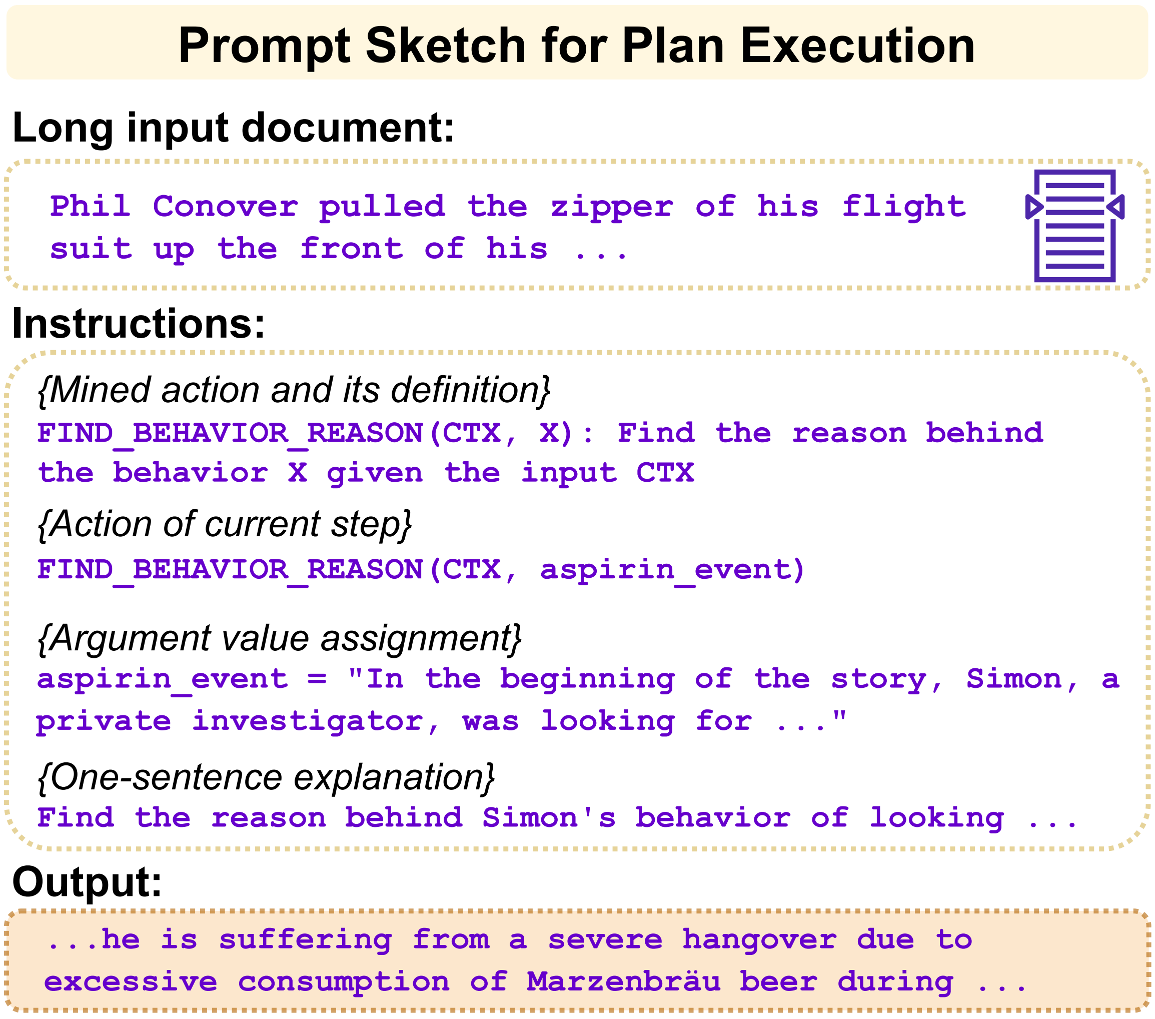
A plan serves as the guiding framework or outline for answering complex questions that may involve multi-step reasoning and/or global understanding of long documents. Given a question, as shown in Figure 4.3, we prompt an LLM to generate a plan based on the previously-mined action set. The plan is formatted as a program, and can be thought of as a more flexible generalization of the program for summarization [150]. Each step of the plan is formatted as

output = ACTION(arg_1, arg_2, ...),

where the `output` variable stores the result of the current `ACTION` , and the `arguments` can be (1) the input document, (2) a string, or (3) an output variable from previous steps of the plan. When generating the plan, we do not show the LLM the entire document as input, which provides ample space for incorporating few-shot in-context examples. Similar to the seed actions in the previous stage, we provide a seed set of plans and allow the model to generate more demonstrations automatically, which we provide more details in Section 4.3.4.

4.3.3 Plan execution

In the previous stage, the LLM generates a plan that serves as a blueprint for producing a response. To execute each step in the plan, we prompt the LLM with a template filled



Prompt Sketch for Plan Execution

Long input document:

```
Phil Conover pulled the zipper of his flight
suit up the front of his ...
```

Instructions:

{Mined action and its definition}

```
FIND_BEHAVIOR_REASON(CTX, X): Find the reason behind
the behavior X given the input CTX
```

{Action of current step}

```
FIND_BEHAVIOR_REASON(CTX, aspirin_event)
```

{Argument value assignment}

```
aspirin_event = "In the beginning of the story, Simon, a
private investigator, was looking for ..."
```

{One-sentence explanation}

```
Find the reason behind Simon's behavior of looking ...
```

Output:

```
...he is suffering from a severe hangover due to
excessive consumption of Marzenbräu beer during ...
```

Figure 4.4: Prompt sketch for plan execution. This prompt contains multiple *{placeholders}* that will be filled with output from previous stages.

with output from previous stages. Concretely, as shown in Figure 4.4, to execute the action `FIND_BEHAVIOR_REASON`, the model fills in the prompt template with (1) the planned action and definition, (2) current action with specific input argument (e.g., `aspirin_event`) , (3) assignment of argument name with output from previous stage (e.g., `aspirin_event` = "`in the beginning of the story, ...`"), and (4) a one-sentence instruction for the current step, all of which are generated by LLM. As the long article is involved in this stage, the prompt is executed in a zero-shot manner.

4.3.4 Self-correction and self-refinement

LLM-generated plans can have two major issues: (1) they can be syntactically-invalid, which prevents execution; and (2) they can semantically irrelevant to the question. To address these issues, we prompt the LLM to "debug" its own generated plans via self-correction and self-refinement, inspired by Reflexion [165]. **Self-correction of syntax errors:** Given a held-out question, we first generate a plan via an LLM and then pass it into a simple parser[3] that returns relevant error messages when the plan does not conform to the defined format. Then, we feed the LLM the question, plan, and error messages, and we ask it to correct the errors in the plan, repeating the process until the parser returns no errors.[4]
Self-refinement of demonstrations: Since we use LLM-generated plans from *training* questions as few-shot demonstrations (Section 4.3.2), it is important for these plans to be semantically meaningful. To ensure the quality of these demonstrations, we validate them by executing the plan and verifying whether they output the correct answer (see Section 4.4.1). If the answer is wrong, we pass the plan to the LLM for further self-refinement and repeat until the execution result is correct; only then do we include the plan as a demonstration.

4.4 Experiments

Dataset selection: We focus on the QuALITY QA dataset [134], which is a multiple-choice QA task in the SCROLLS benchmark [160]. While we would love to experiment on more datasets, this area remains unexplored: QuALITY is the only known dataset that has verified human annotations on whether the *usage of long contexts* is critical to answering a question. Other QA datasets such as NaturalQuestions [96] and NarrativeQA [88], which take long documents as inputs, are not relevant for our work as the vast majority of answers

[3]The simple parser checks the format of the plan, and returns errors such as *No '=' found in one of the actions*, etc.

[4]It is possible for the LLM to fail to generate a syntactically-valid plan even after multiple retries. In such cases, we revert to the zero-shot baseline (i.e., without PEARL). This happens for only 4 out of 1K examples in our experiments, so it is not a major issue.

	QuALITY Long	QuALITY Short	All	p-val
Prompting Methods				
GPT-4 zero-shot	64.3	**79.1**	68.8	-
GPT-3.5 zero-shot (text-davinci-003)	45.5	56.3	48.8	0.000
GPT-4 zero-shot chain-of-thought	65.9	77.2	69.3	0.766
GPT-4 Pearl	**70.9**	77.8	**73.0**	0.005
Ablations of GPT-4 Pearl				
w/o plan execution	67.3	77.2	70.3	0.295
w/o self-refinement of plan demonstrations	67.0	78.8	70.6	0.245

Table 4.2: We present baseline and Pearl as well as ablation results on our generative subset of QuALITY questions. **Long** denotes the split where the questions require reasoning over long contexts to answer accurately. As we only evaluate on a subset, we also provide p-values to verify statistical significance against the zero-shot GPT-4 baseline.

can be located by retrieving short excerpts without processing long-range dependencies within the context.

In total, we extract a dataset of 1K examples from QuALITY divided into two splits, one of which requires long context understanding to answer and the other of which doesn't. Each QuALITY question contains a human-annotated score of how much context is required to answer it, which ranges from 1 (*only a sentence or two of context is needed*) to 4 (*most or all of the passage for context is needed*). The two splits are (1) **Long**, which consists of 330 examples from the QuALITY dev set and 368 examples from training set marked with a context score ≥ 3, and (2) **Short**, which has 302 examples from the dev set that do not require long contexts to answer (context score < 3). The latter is a control dataset to make sure our methods do not overly worsen performance on simpler questions.

Evaluation: While QuALITY is a multiple-choice dataset, we reframe it into a generative task in which an LLM does not have access to the choices and must instead generate a long-form answer. We do this for two reasons: (1) transforming the task to a novel setting reduces the risk of data leakage, and (2) the generative task better resembles the usage of LLMs in real world. In our generative setup, we automatically map the long-form answer generated by the models back to one of the choices with an LLM to evaluate the accuracy.[5] The accuracy of mapped answers serves as a proxy for assessing the correctness of the provided answer.

However, to better simulate LLMs usage in real-world scenarios, we turn this dataset into a *generative* task[6] in which an LLM does not have access to the choices and must instead generate a long-form answer. Then, we automatically map the generated answer back to one of the choices with an LLM to evaluate the accuracy.[7] The accuracy of mapped answers serves as a proxy for assessing the correctness of the provided answer. QuALITY contains a diverse variety of questions, each of which is annotated with the amount of context needed to answer the question.

Focusing on long-document reasoning: In contrast to questions that can be correctly answered with local context once a piece of information is located, as in *"Who found Retief and Magnan in the trees?"*, we are more interested in questions that require reasoning over long context, as in *"How would you describe the changes in tone throughout the passage?"*. These questions constitute an interesting and difficult subset that, unlike more straightforward information seeking questions, require global understanding and reasoning over the document to provide accurate answers. Therefore, we select a subset of questions marked by the QuALITY creators as requiring long contexts to answer. In total, we create a dataset of 1K examples divided into two splits:[8] (1) **Long**: 330 examples from the dev set, 368 examples from training set, and (2) **Short**: 302 examples from dev set that do not require

[5]We provide a generic illustration of the evaluation process in Figure C.1. In Appendix C.4, we confirm through human evaluation that GPT-4, the model we test, demonstrates considerable—but not perfect—agreement with human annotators for the answer mapping stage.

[6]We provide the performance of GPT-4 with standard multi-choice setup on the full dev set in Appendix C.3.

[7]We provide a generic illustration of the evaluation process in Figure C.1. In Appendix C.4, we confirm through human evaluation that GPT-4, the model we test, demonstrates considerable—but not perfect—agreement with human annotators for the answer mapping stage.

[8]Human annotation score on the required context ranges from 1 to 4. Questions in the long split have average human annotation score ≥ 3, and the short split scores < 3.

long contexts to answer. The latter forms a control dataset to make sure our methods do not overly worsen performance on simpler questions.

4.4.1 Experimental setup

As each of the stages in PEARL has critical hyperparameters and implementation details, we describe our specific configurations here.

Action mining: We provide an LLM with seven seed actions and two in-context examples to demonstrate the required format for generating new actions.[9] We collect new actions by passing all training set questions into the model, excluding those questions in our evaluation set. Ultimately, we obtain 407 actions and corresponding definitions, of which several are duplicates or overly specific, and in total exceeds GPT-4's maximum context window of 8K tokens. We thus instruct GPT-4 to simplify and abstract over existing actions to reduce the total number of actions. After repeating this process twice,[10] the number of actions is reduced to 81, forming the final action set for PEARL.

4.4.2 Baselines

As existing sophisticated prompting methods require few-shot examples in-context, which is not feasible when long document is involved, we compare PEARL with simple zero-shot baselines (GPT-4 [130] and GPT-3.5 [132]), where we directly prompt the model to provide a detailed free-form answer. Additionally, we also evaluate zero-shot CoT prompting for GPT-4 by adding "Let's think step-by-step," to the prompt.

[9]We present the prompt template in Appendix C.6

[10]After one round, the actions reduced to $\sim$140, and after four rounds to $\sim$20. We provide ablations on the number of actions in Section 4.5.

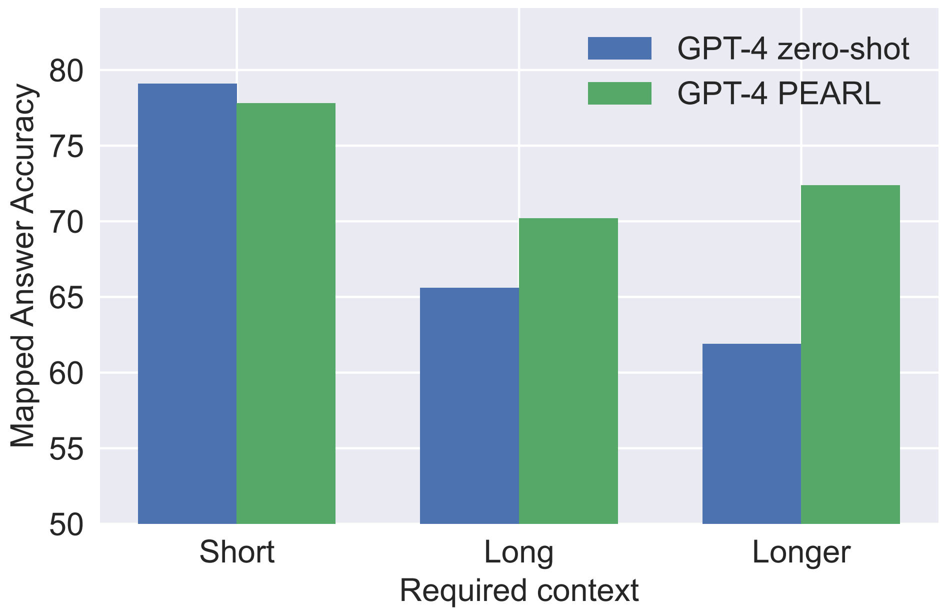


Figure 4.5: Accuracy by the amount of required context to answer, as annotated by humans in QuALITY. The short, long, and longer splits correspond to average annotation scores on the amount of required context [1, 3), [3, 3.5), and [3.5, 4), respectively.

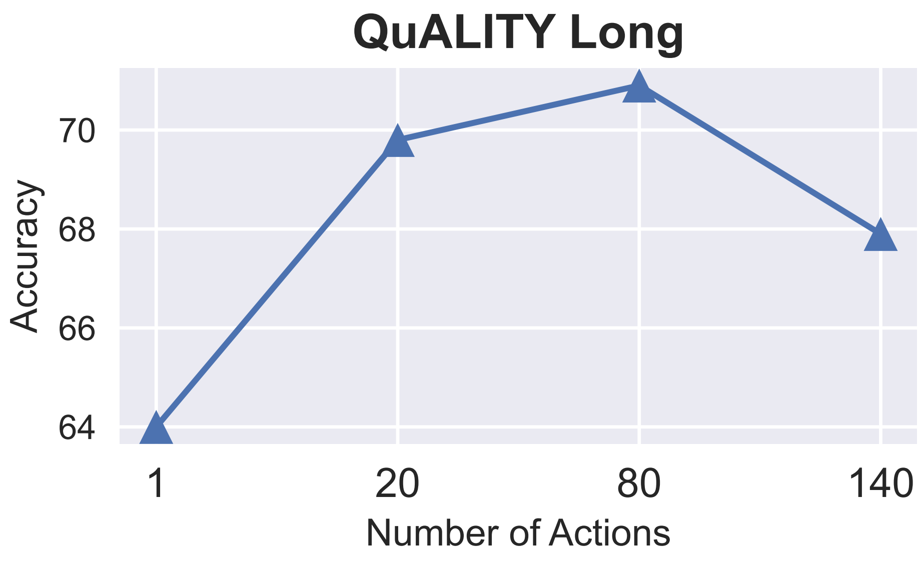


Figure 4.6: PEARL accuracy given in-context action sets of various sizes. Having too few or too many actions impairs the performance.

4.5 Main results

We discover that PEARL significantly outperforms competing prompting methods on questions that require reasoning over long contexts, which demonstrates the utility of the planning module. We also observe a small drop in accuracy on questions that require only short contexts, possibly because the plans end up over-complicating what is a simple reasoning process. In this section, we dig deeper into the main results of our experiments, which are presented in Table 4.2.

PEARL improves accuracy on long-document QA: Overall, PEARL's accuracy is higher than that of all competing methods, particularly for the QuALITY split annotated by humans as requiring long contexts to answer (**Long**). Furthermore, we observe in Figure 4.5 that for questions marked by QuALITY workers as requiring the longest possible context, PEARL improves substantially compared to the zero-shot GPT-4 baseline (72.4% vs 61.9%). Our method's slightly diminished performance on the **short** split is likely due to both "overthinking" these simpler questions, as well as error propagation from plan execution steps as highlighted in Section 4.6. Finally, we point out that all methods achieve higher accuracies on the **Short** split compared to the **Long** split, indicating the challenging nature of this set of questions.

Number of actions impacts performance: In Figure 4.6, we show that the size of the action set is an important factor in PEARL's performance. With just a single action (i.e., `EXECUTE` a free-form natural language instruction),[11] PEARL's accuracy on the **Long** subset drops to 64%. With too many actions (140 in the plot), its accuracy also degrades, likely because the action space is too fine-grained for the model to properly execute all actions. We note that the optimal number of actions likely differs from task to task, so it is an important hyperparameter to consider when tuning PEARL.

[11]We additionally preserve the `CONCAT` action in this setting due to its necessity when aggregating execution results.

	Count	GPT-4 PEARL	GPT-4 zero-shot
Why/reason	316	0.79*	0.71*
Person	216	0.75*	0.66*
Event	199	0.69	0.68
Not/except	118	0.70*	0.53*

Table 4.3: Accuracy by reasoning types. * denotes statistically significant improvement with p-val < 0.005. We provide other reasoning types in Appendix C.2.

Action execution is necessary: Do we actually need to *execute* the generated plans to answer these questions? Feeding just the generated plan to the model along with the question (minus any execution results) may still encourage the LLM to follow the plan's reasoning steps and generate a better answer. However, we observe that removing the execution results from the model's input reduces absolute accuracy by around 3 points, which suggests that it is important to perform multiple passes over the document to execute each action before answering the original question. With that said, we do observe a modest improvement over the GPT-4 zero-shot and CoT baselines ($\sim$ 2 absolute points), which suggests that the plan itself is also valuable.

Self-refinement improves performance: To reduce human input, the majority of the plan generation demonstrations are generated by the LLM with self-refinement. We observe that self-refinement is critical to performance: without it, the overall accuracy drops nearly 3 absolute points (ablations in Table 4.2), which highlights the importance of high-quality few-shot examples for plan generation.

4.6 Analysis

In this section, we analyze the behavior of PEARL by diving into the composition of its generated plans, its most preferred actions, and what types of questions it improves most on. We also offer a qualitative error analysis as well as a human evaluation on the correctness of the generated plans.

Plan statistics: Plans are roughly 4 actions long on average, with around 3.4 unique actions per plan. The most commonly used actions are shown in Figure C.2 in Appendix C.2. Apart from the string concatenation action `CONCAT`, the most frequently used action is `FIND_CHARACTER`, which can be convenient for understanding long literary text. Other less often used actions cover both those that can transfer across domains, e.g., `COMPARE`, and those specific to narrative understanding, e.g., `FIND_EMOTION`.

Accuracy by reasoning types: Since QuALITY questions require different reasoning strategies to solve, what types of reasoning does PEARL help improve the most? To this end, we further evaluate questions based on the type of reasoning required to answer them.[12] Table C.2 shows that PEARL significantly improves three reasoning types: *why* questions (reasoning about a cause), *person* questions (reasoning about the person(s) involved in an event), and *not/except* questions (e.g., "which of the following is not a reason for...").

PEARL is significantly slower than zero-shot prompting: The improved performance of PEARL comes at the cost of longer running time and cost: PEARL requires 4.4 times more tokens in the prompt, and it needs to generate 1.3 times more tokens owing to the intermediate steps.[13]

Specific examples where PEARL helps: To better understand PEARL, we qualitatively analyze 40 examples for which zero-shot GPT-4 generates incorrect answers while PEARL answers correctly. This analysis reveals two key advantages of PEARL. First, while zero-shot prompting is reasonably good at finding salient information from the input document, its generative answers tend to be based only on *local* context around this information. For instance, when asked about the number of wives the character "Dan Merrol" has, the baseline successfully identifies six names that appear to be Dan's wives. However, PEARL takes into account the revelation that these names " *were actually memories from the brain donors*

[12]We prompt GPT-4 with the definition of each reasoning type presented in the Appendix [134] and ask it to label each question with up to two reasoning types.

[13]These multiples were estimated from a small run of 30 examples.

whose parts were used to reconstruct his brain" and thus correctly reasons that Dan only has one wife. Second, PEARL generates more detailed and thorough answers. For instance, given the question *"Why is Kumaon a good region for potential forest preservation?"*, the zero-shot answer considers only one aspect of the reason, whereas PEARL elaborates on multiple aspects, allowing PEARL's answer to be mapped to the correct option (" All other choices"), while the zero-shot answer maps to the option that describes the single aspect.

Where does PEARL go wrong? We additionally examine 40 examples for which PEARL answers incorrectly, and group the errors into three categories (detailed examples in Appendix C.2 Table C.3):

- **True negatives:** Questions for which PEARL's generative answer is mapped to the wrong option. This category can be further divided into two subcategories: (1) cases where the plan has critical issues, and (2) cases where the plan is satisfactory but the intermediate execution produces incorrect output. Out of the 40 examples, 29 are true negatives, with 7 plan errors and 22 execution errors.
- **False negatives:** Questions for which PEARL's generative answers are correct but incorrectly mapped to the wrong option. This kind of error is unavoidable as we use LLM for automatic answer mapping. Out of the 40 examples, 5 are false negatives.
- **Other:** Some QuALITY questions are heavily dependent on the options; that is, the correct answer can only be determined after examining all the options. For instance, Table C.3 presents a question asking who would enjoy the story the most of the given options. Although PEARL offers an answer based on the story's genre—which is not incorrect—it is not as accurate as the gold label. Furthermore, there are instances where the model's free-form answers lack sufficient details and can thus be mapped to more than one option or no options at all. We classify these responses as a separate category. Out of 40 examples, 6 fall into this **Other** category.

Human evaluation of model-generated plans: The quality of plans generated by PEARL is critical, as they serve as the basis for the plan execution stage. To gain further insight on the quality of these plans, we perform a human evaluation by hiring annotators on Upwork[14] to provide feedback on the generated plans.[15] Concretely, we ask annotators to assess (1) the correctness of the plans (binary choice), assuming error-free execution at each step, and (2) provide free-form feedback on any flaws or potential improvements. On average, annotators regard over 97% of all plans as correct, with over 94% confidence, although these numbers are inflated because the annotators do not have access to the long story when making these judgments. More interestingly, after aggregating their feedback over common themes (more details in Table C.1 Appendix C.2), we find that the primary issue with existing plans is the presence of unnecessary steps (10% of the total annotated plans). Annotators also notice that GPT-4 can be inattentive to subtle details while generating plans. For example, given the question "*Do you think it would be fun to live in the universe in which this story takes place?*", the model decides to " *evaluate the pros and cons of living in the universe based on the features found in the input article*". However, human annotator argues that "*just because something is positive doesn't necessarily mean it is "fun". Any pros on the list might outweigh the dangers noted, resulting in an incorrect answer of 'yes'...*".

4.7 Conclusion

In this work, we introduce PEARL, a framework for tackling complex reasoning over long documents. To answer a question, PEARL first proposes a plan based on a set of actions mined from a training set, and then it executes the plan step by step via prompting itself with a template filled with output from previous stages. We demonstrate the effectiveness of PEARL on a challenging subset of QuALITY. Experiments and analysis show that

[14]We pay the annotators at the rate of $25/h.

[15]We provide a few examples in Appendix C.7.

prompting GPT-4 with PEARL yields more accurate and comprehensive answers than zero-shot and chain-of-thought prompting, and human annotators judge the generated plans to be reasonable.

Limitations

While PEARL shows promising results for long document reasoning, there are several limitations to our approach. Like other prompting methods, PEARL is susceptible to generating misinformation or hallucinations. It is also more time-consuming and computationally costly than the baseline approach of directly prompting an LLM to answer the question. Moreover, PEARL may over-complicate simple questions that only need superficial reasoning over long-form narratives. Due to our limited budget and the cost of API access to proprietary LLMs, we did not stress test the framework with extensive variations in the prompt aside from the ablations mentioned in this chapter. Finally, PEARL is still bounded by the maximum context window size of the LLMs, and we have not tested it on less powerful LLMs. Overall, prompting on document-level with continuous dependencies is still an under-explored area, and we hope our work spur future research in this space (e.g., new datasets, modules, stage refinements).

Ethics Statement

PEARL relies heavily on closed-source large language models, which while tuned to align with human preferences, are still susceptible to generating hallucination and misinformation. The documentation of these models is opaque, and it is difficult to know to what extent the copyrighted data is used during pre-training. We use these models for purely research purposes. We hope our method can shed light on mitigating similar issues when an LLM needs to process long document. Finally, human annotators are paid hourly, and the evaluation process was deemed exempt from IRB review.

CHAPTER 5

SUFFIXRL: TRAIN LANGUAGE MODELS WITH SEGMENT-LEVEL SIGNALS

5.1 Introduction

In the previous chapter, we introduced a prompting framework to improve the capability of large language models in reasoning over long-range context. However, the efficacy of the prompting-based approach relies heavily on pre-trained language models with robust instruction-following capabilities. Language model alignment is an evolving area on its own. To untangle potential complexities and delve deeper into the training process of language modeling, we shift our focus in this chapter towards a more fundamental aspect – modifying the training objective. We propose a new training objective for language modeling based off of the insights we discussed in Chapter 3.

In Chapter 3, we demonstrated that SuffixLM, a segment-level language model, achieves superior performance on the suffix identification task CHAPTERBREAK compared to many open-source large language models trained with next-token prediction objective. Unlike conventional models, SuffixLM operates at a coarser granularity, predicting the representation of the entire future segment instead of predicting the most likely next token. By conditioning on the predicted representation of the future segment, we posit that models can generate continuations that are more coherent and logically connected to the prefix compared to leveraging only the tokens in the previous context, as in standard autoregressive language models. However, SuffixLM cannot be used for token-by-token decoding due to its segment-level training objective. This prompts a new research question: how can we train a language model with segment-level signals while still capable of generating token-by-token?

Recall that SuffixLM is trained with a contrastive loss (Eq. 3.1), where the predicted representation is rewarded when it is close to the gold representation and penalized when it is more similar to the negatives. This contrastive loss can be viewed as a generalized version of Bradley-Terry [17] model, which is commonly used to model pairwise preferences, and is now widely used for training reward model in the alignment stage of large language models [132, 33, 100, 45]. Drawing a connection between the SuffixLM training objective and the reward modeling in recent popular RLHF paradigm, we propose SuffixRL – a training method that allows model to leverage segment-level signals while still maintaining the autoregressive decoding capability.

In the proposed SuffixRL training, unlike conventional reward modeling in RLHF framework, which relies on large amounts of high quality human preference data, SuffixRL operates in a more self-supervised manner, eliminating the need for any form of human feedback data. SuffixRL consists of two stages:

- **Reward modeling:** In this initial stage, we train a reward model using Bradley-Terry model by contrasting gold continuation sampled from human-written documents and model generated continuation, given the same prefix.

- **Reinforcement learning:** After we get a robust reward model, we train the policy with Proximal Policy Optimization (PPO) [157] against the reward model.

Due to the large memory consumption and resource-demanding nature of PPO training, we introduce two changes to conventional experimental setup and validate our changes on a recent large language model evaluation benchmark AlpacaFarm [43]. In our experiments, we adopt low-rank adaptation [66] to bypass the large memory consumption in the fine-tuning large language models, and adopt Jensen-Shannon divergence regularizer which we empirically find to perform better than standard KL regularizer during the RL stage in our LoRA setting. With a validated setup, we then verify if the model trained with SuffixRL can lead to more coherent output. We employ an LLM-based evaluation which has been demonstrated to correlate well with human judgements [43, 111]. Our experimental results indicate that models trained with SuffixRL can indeed yield more coherent continuations compared to models trained with standard next-token prediction.

5.2 SuffixRL

Reward modeling Consistent with recent works on RLHF, we employ the standard Bradley-Terry model to train a reward model. Specifically, let x be the prefix, y_w be the gold continuation of the prefix x, and y_l be the continuation generated by the model which we will fine-tune in the second stage. The training objective of the reward model is:

$$\mathcal{L} = -\log \sigma(r(x, y_w) - r(x, y_l)).$$

Intuitively, models trained with this objective learn to assign higher score to the winning (gold) continuation in the presence of a losing (generated) continuation. An underlying assumption in this stage is that gold continuations are consistently better than model-generated ones. However, in reality, many recent large language models can generate output that is fluent, and at first glance, of high quality and indistinguishable from human generated output. Recent works uncover that, despite the good quality when examined superficially, these models still lack robustness [69, 41] and are susceptible to incoherence [126] and hallucination [214]. We posit that a reward model trained on pairs where surface-level impressions are indistinguishable from each other, should learn to make the preference judgments based on more nuanced aspects such as coherence and relevance. In section 5.6, we will verify this hypothesis with fine-tuned LLaMA-1-7B models.

Reinforcement learning In this stage, we optimize the policy (language model to be fine-tuned) with proximal policy optimization algorithm (PPO). PPO training [157] iterates between a rollout phase and an optimization phase. (1) **Rollout**: In rollout, π_θ, the policy being optimized, generates responses to a batch of input instructions. Each response is then assigned a scalar score by the reward model, which we trained in the previous stage. This score is then used to estimate the advantage[1] for optimization in the next phase. (2) **Optimization**: The policy is optimized to maximize a surrogate objective using the advantage estimated during rollout. Then, the newly-optimized policy is used to generate responses in the next rollout phase. The surrogate objective in the optimization phase has the form

$$\mathcal{J}(\theta) = \mathbb{E}[\min(r(\theta)\hat{A}_{\theta_{\text{old}}}, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{old}}})], \tag{5.1}$$

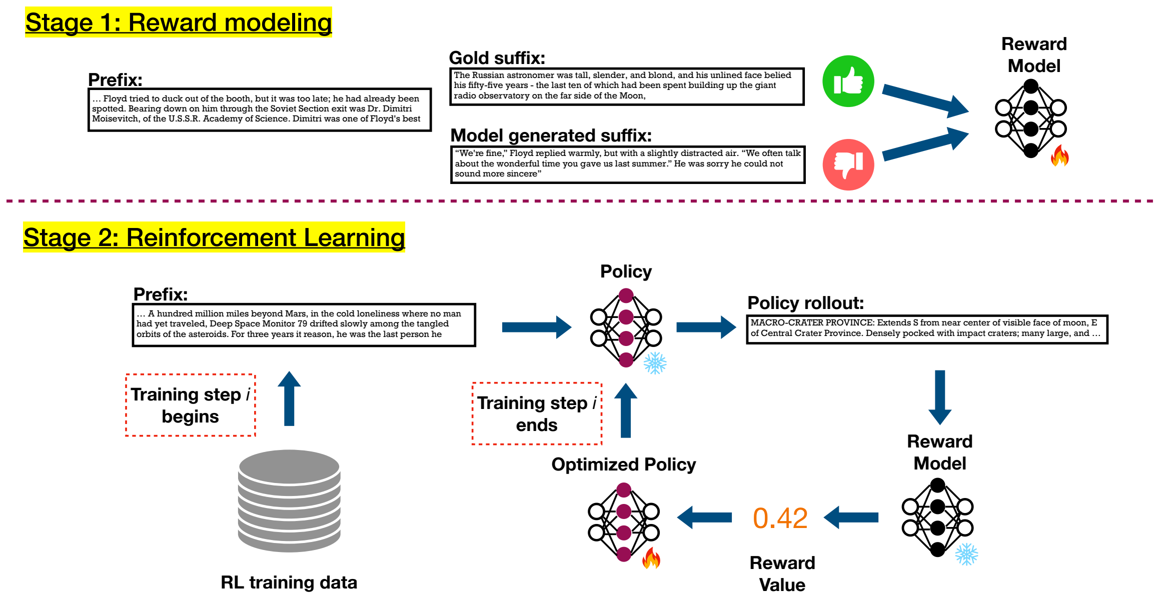
where $r(\theta) = \frac{\pi_\theta(y|x)}{\pi_{\theta_{\text{old}}}(y|x)}$ denotes the output probability ratio between the current policy π_θ and the policy π_{old} from the previous rollout step, computed on responses y sampled during the previous rollout. $\hat{A}$ denotes the estimated advantage [156], which depends on the reward value; we refer readers to other materials [156, 203] for understanding advantage estimation. The clipped output probability ratio stabilizes training by preventing large policy updates, thus discouraging π_θ from deviating too much from π_{old} during optimization.

5.3 Efficient training and regularization for SuffixRL

5.3.1 Low-rank Adaptation

Full-model fine-tuning large language models with PPO is resource demanding due to the need of loading multiple large models into GPU memory and storing necessary optimizer states. For instance, full-model fine-tuning 7-billion level LLaMA with PPO requires 8 A100 80GB GPUs [43]. Due to a lack of enough compute resources, we employ parameter-efficient fine-tuning (PEFT) method for all experiments in this chapter. A common PEFT method gaining popularity recently is low-rank adaptation [66] (LoRA). In LoRA, an input hidden state $\mathbf{h}_{\text{in}} \in \mathbb{R}^d$ is projected to $\mathbf{h}_{\text{out}} \in \mathbb{R}^d$ via a weight matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$ and two

[1] A positive advantage suggests that when π_θ generates token y given a certain prefix, it receives a reward higher than the average reward expected from generating other tokens in the vocabulary given the same prefix.



… Floyd tried to duck out of the booth, but it was too late; he had already been spotted. Bearing down on him through the Soviet Section exit was Dr. Dimitri Moisevitch, of the U.S.S.R. Academy of Science. Dimitri was one of Floyd's best

The Russian astronomer was tall, slender, and blond, and his unlined face belied his fifty-five years - the last ten of which had been spent building up the giant radio observatory on the far side of the Moon,

“We're fine,” Floyd replied warmly, but with a slightly distracted air. “We often talk about the wonderful time you gave us last summer.” He was sorry he could not sound more sincere”

… A hundred million miles beyond Mars, in the cold loneliness where no man had yet traveled, Deep Space Monitor 79 drifted slowly among the tangled orbits of the asteroids. For three years it reason, he was the last person he

MACRO-CRATER PROVINCE: Extends S from near center of visible face of moon, E of Central Crater Province. Densely pocked with impact craters; many large, and …

Figure 5.1: An illustration of SuffixRL pipeline. In the first stage (reward modeling), synthetic pairs consisting of gold suffix and model generated suffix are used to train a reward model. In the second stage, the policy is trained against the reward model derived from the first stage with proximal policy optimization (PPO). The second stage iterates between rollout and optimization. During rollout the policy is frozen and generates suffixes which combined with corresponding prefixes are then passed into the reward model to get reward signals. During optimization, the policy model is unfrozen and trained on the reward signals collected during rollout.

low-rank decomposition matrices $\mathbf{A} \in \mathbb{R}^{k \times d}$ and $\mathbf{B} \in \mathbb{R}^{d \times k}$, where k is the rank of the low-rank matrices ($k \ll d$) and α is a scaling hyperparameter:

$$\mathbf{h}_{\text{out}} = (\mathbf{W} + \frac{\alpha}{k}\mathbf{B}\mathbf{A})\mathbf{h}_{\text{in}} \tag{5.2}$$

During LoRA training, $\mathbf{W}$ is kept frozen while the decomposition matrices $\mathbf{A}$ and $\mathbf{B}$ are trained.

5.3.2 KL regularization

While the clipped ratio in Equation 5.1 constrains the extent to which π_θ can change from a recent policy π_{old}, π_θ can still reach a sub-optimal region by "reward hacking" [133] after enough rollout-optimization steps. To avoid this, a KL regularization term is added to penalize π_θ when it deviates too far from a reference policy π_{ref} during PPO training; the reference policy is usually set to the output of the first stage of RLHF, i.e., a model that has gone through supervised fine-tuning.

Formally, let $\mathbf{x}$ be an instruction and $\mathbf{y}$ be a corresponding response of L tokens sampled during rollout. While the reward model produces a scalar value $r(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$ given the pair $(\mathbf{x}, \mathbf{y})$, the total reward $\mathbf{r}(\mathbf{x}, \mathbf{y})$ of the response $\mathbf{y}$ is a vector of dimensionality $\mathbb{R}^L$ due to the KL penalty:

$$\mathbf{r}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} 0 & 0 & \dots & r(\mathbf{x}, \mathbf{y}) \end{bmatrix}^\top - \beta \text{KL}(\pi_\theta(\mathbf{y} \mid \mathbf{x}), \pi_{\text{ref}}(\mathbf{y} \mid \mathbf{x})) \tag{5.3}$$

Here, $\text{KL}(\cdot, \cdot)$ denotes the KL divergence [95] between π_θ and π_{ref} at each position (token) of $\mathbf{y}$, and the scalar reward r is added to the last position of the KL term. The shaped reward $\mathbf{r}$ is then used to estimate the advantage, which is at the core of PPO.

As it is memory-consuming to compute the KL divergence over the entire vocabulary at each timestep,[2] prior work approximates the KL term via Monte-Carlo estimation [155]. Does the form of this approximation make an impact when implementing RLHF with LoRA? In this thesis, we compare the following divergence estimators based on win rate:

- **KL divergence:** Given the distributions over the vocabulary predicted by both the current policy π_θ and the reference policy π_{ref} at token y of the response, the Kullback-Leibler divergence is defined as $\text{KL}(\pi_\theta(y), \pi_{\text{ref}}(y)) = \sum_{y \in \mathcal{V}} \pi_\theta(y) \log \frac{\pi_\theta(y)}{\pi_{\text{ref}}(y)}$. By minimizing the KL divergence, we effectively add an entropy regularizer that diversifies the generated tokens while also maintaining high likelihood under the reference policy. In practice (and in many open-source implementations), the true KL divergence is approximated as $\log \pi_\theta(y) - \log \pi_{\text{ref}}(y)$, which saves memory as only the probability of the generated token y is stored instead of the entire distribution. While this estimator is unbiased, it also suffers from high variance [154]; to enforce non-negativity, the AlpacaFarm implementation clamps the minimum value to zero, which reduces variance at the cost of biasing the estimator.

- **Alternative divergence approximations:** Past work [154] proposes two alternative KL approximations that have lower variance. The first takes the form of squared error between $\log \pi_\theta(y)$ and $\log \pi_{\text{ref}}(y)$, which is biased and approximates a different f-divergence measure.[3] The second one is the Bregman divergence $B_{\mathcal{F}}(\frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)}, 1)$ associated with the convex function $\mathcal{F}(x) := -\log x$, which is an unbiased KL

[2]Computing the true KL requires summing over the entire vocabulary, which is not economical if done for every token in the sequence.

[3]$f(x) := \frac{1}{2}(\log x)^2$

estimator.[4] Both alternatives are shown to reduce variance and bias in a toy setting by [154]; in this section, we verify their effectiveness on real natural language tasks.

- **Jensen-Shannon divergence:** We also experiment with the Jensen-Shannon divergence, which is the average of the KL divergence for both π_{ref} and π_θ against the intermediate distribution $\pi_m(y) = \frac{1}{2}(\pi_{\text{ref}}(y) + \pi_\theta(y))$. Note that the responses are sampled from π_θ instead of π_{ref}. Similar to the AlpacaFarm implementation, the minimum value is clamped to zero for stabilized training. We fully specify the estimator in Table 5.1.

- **No KL regularization:** Finally, to evaluate whether KL regularization is necessary under the LoRA setup, we also discard KL penalty entirely in two of our configurations, one with dropout [166] and the other without.

5.4 Validation of efficient training and alternative regularization on AlpacaFarm

5.4.1 Validation setup

To validate the feasibility and correctness of our experimental setup, we conduct evaluation on AlpacaFarm [43]. AlpacaFarm collates existing open-source instruction-following datasets [197, 9, 27, 51] to form an evaluation dataset of 805 diverse instructions. To evaluate two models against each other on this dataset, we first collect responses generated by both models for all 805 instructions. Then, a pool of large language models (`GPT-4-0314`, `GPT-3.5-turbo`, `text-davinci-003`) are prompted to provide preference judgments (i.e., which model's output is better for a given instruction) by simulating human

[4]It is an unbiased estimator of KL because

$$\mathbb{E}_{y\sim\pi_\theta}[\frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)} - 1 - \log\frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)}] = -1 + \sum_{y\in V}\pi_\theta(y)\frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)} - \sum_{y\in V}\pi_\theta(y)\log\frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)}$$

Adaptation method	**Regularization**	**Divergence estimator**	**Win rate**
Publicly-released supervised fine-tuning (SFT-10k) checkpoint from [43]			
Full model tuning	-	-	37.0
Publicly-released PPO checkpoint from [43]			
Full model tuning	Clamped KL	$\max(0, \log \pi_\theta(y) - \log \pi_{\text{ref}}(y))$	46.7
Our PPO models trained with LoRA			
LoRA	Clamped KL	$\max(0, \log \pi_\theta(y) - \log \pi_{\text{ref}}(y))$	47.5 ± 0.2
LoRA	KL	$\log \pi_\theta(y) - \log \pi_{\text{ref}}(y)$	46.7 ± 0.02
LoRA	Bregman	$\frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)} - 1 - \log \frac{\pi_{\text{ref}}(y)}{\pi_\theta(y)}$	49.0 ± 0.1
LoRA	Squared error	$\frac{1}{2}(\log \pi_\theta(y) - \log \pi_{\text{ref}}(y))^2$	47.1 ± 0.1
LoRA	Jensen-Shannon	$\frac{1}{2}\max(0, \log \pi_\theta(y) - \log \frac{1}{2}(\pi_\theta(y) + \pi_{\text{ref}}(y)))+$ $\frac{1}{2}\max(0, \log \pi_{\text{ref}}(y) - \log \frac{1}{2}(\pi_\theta(y) + \pi_{\text{ref}}(y)))$	49.8 ± 0.3
LoRA	*No regularization*	-	48.2 ± 0.1
LoRA	Dropout only (p=0.5)	-	48.4 ± 0.2

Table 5.1: Our experiments demonstrate that (1) LoRA is a competitive parameter adaptation method to full-model fine-tuning for PPO training in RLHF; and (2) PPO training with LoRA does not require any additional regularization (KL or otherwise) to succeed. We evaluate the win rate of each model in the table against `text-davinci-003` on the AlpacaFarm evaluation data. Preference judgments to calculate win rate are simulated by a pool of automated LLM annotators (e.g., GPT-4). We evaluate the public AlpacaFarm checkpoints in the top two rows (SFT-10k and PPO); for each of the remaining rows, we perform three runs of PPO optimization with LoRA and the corresponding regularizer, and we report the mean and standard error with bootstrap sampling.

annotators, which allows us to compute the *win rate* of one model over the other. The LLM-based simulated workflows were shown to highly correlate with human raters in system-level comparison with a Spearman correlation of 0.98 [43]. As in the original codebase, we compute the win rate of each of our model configurations against OpenAI's `text-davinci-003`.

Baselines AlpacaFarm release checkpoints for the SFT-10K step (stage 1 of RLHF), the reward model trained on human preference (stage 2), and the PPO-optimized stage 3 checkpoint. Their SFT-10K model[5] reaches a win rate of 37% against `text-davinci-003`, while the PPO-optimized checkpoint[6] has a win rate of 47%. We use their released SFT-10K and PPO-optimized models as full-model tuning baselines.

5.4.2 Results & analysis

We observe several interesting phenomena while varying the regularization estimator within our LoRA RLHF setup. First and perhaps most notably, completely *removing* the KL regularization penalty does not affect win rate (it actually increases from 47.5% to 48.2% given same amount of compute). Of the alternative estimators, the Jensen-Shannon divergence yields the highest overall win rate (49.8%). We also compare the quality of the estimator against the true KL divergence computed over the full distributions and discover that win rate is not necessarily correlated with low KL divergence.

KL regularization is not critical when using LoRA. While previous work [132, 169] include the KL regularization penalty during PPO training, it is not necessary to achieve a high win rate within our LoRA-based experimental setup. In fact, as shown in Table 5.1, a LoRA configuration without any KL regularization outperforms the released AlpacaFarm PPO checkpoint (48.2 vs 46.7). We conjecture that LoRA provides implicit regularization by freezing most of the parameters (e.g., feed-forward layers, layernorm, and embeddings),

[5] `https://huggingface.co/tatsu-lab/alpaca-farm-sft10k-wdiff`

[6] `https://huggingface.co/tatsu-lab/alpaca-farm-ppo-human-wdiff`

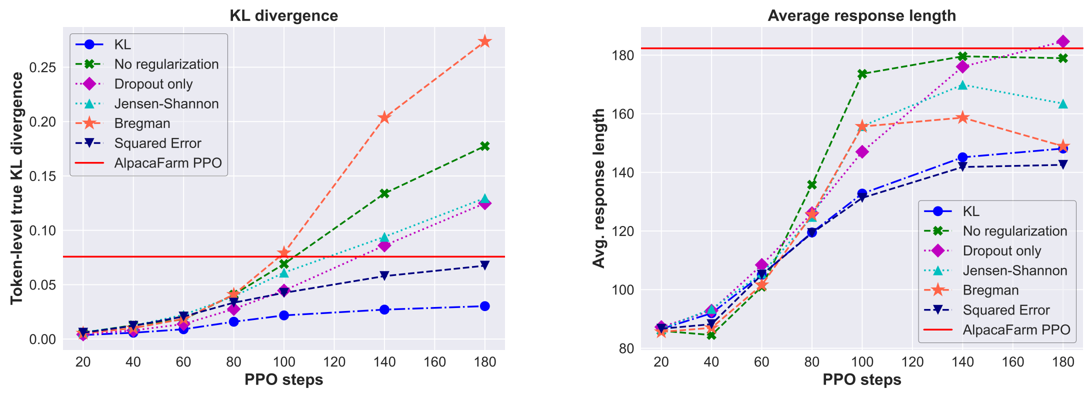
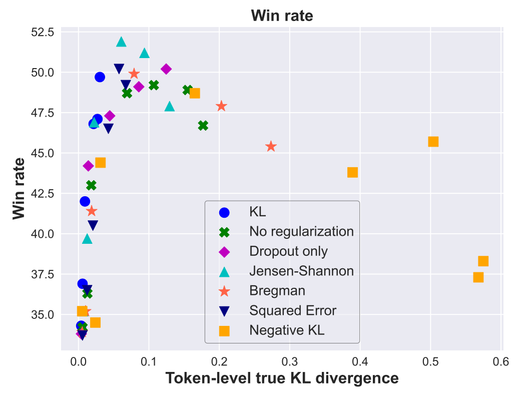


Figure 5.2: (**Left**): Regardless of the estimator, the true KL divergence (measured on 28K tokens sampled from AlpacaFarm) steadily increases as training proceeds. The standard KL estimator is predictably the most effective regularizer in terms of reducing true KL divergence. (**Right**): Response length increases until about 100 PPO steps, after which it plateaus or drops for most configurations.

which already discourages large deviations from the reference policy in parameter space. Previous work on other parameter-efficient fine-tuning methods [65] such as prompt tuning [102, 188] and prefix tuning [107] also demonstrate less overfitting in low-data regimes, which corroborates our hypothesis. One major caveat in our experiments is that properly verifying the regularization effect of LoRA requires a comparison to full-model fine-tuning without the KL penalty, for which we do not have adequate resources.

Other divergence estimators outperform the standard KL estimator. Table 5.1 shows that the KL estimator used in open-source RLHF implementations such as TRLX underperforms alternative estimators. KL estimator does benefit from clamping (46.7 vs. 47.5 when clamped), but it is still worse than Jensen-Shannon regularizer (46.7 vs. 49.8) at step 100. The Jensen-Shannon estimator consistently outperforms all other estimators after step 60 in our experiments, and thus appears to be a promising alternative for future RLHF work. This finding aligns with previous work [52], who also show that the Jensen-Shannon divergence outperforms other divergence measures when fine-tuning language models to approximate energy-based models.



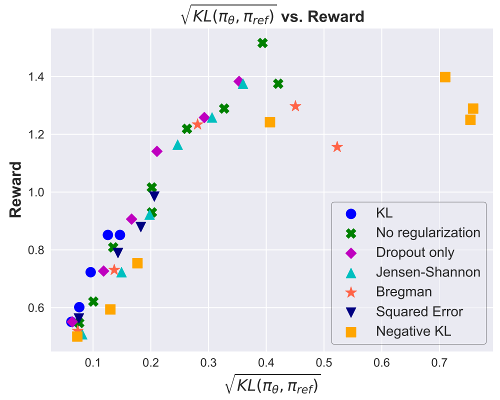


Figure 5.3: (**Left**): We plot the win rate vs. token-level true KL divergence for the checkpoints from steps 20, 40, 60, 80, 100, 140, and 180 of multiple configurations and multiple runs. In general win rates sharply increases and then more gradually decreases as the true KL divergence increases. (**Right**): The linear relationship between $\sqrt{\mathrm{KL}(\pi_\theta, \pi_{\mathrm{ref}})}$ and reward, observed in prior work, also holds in our LoRA implementation within a certain regime.

We observe high win rates even when the KL divergence is moderately large. RLHF uses Monte-Carlo estimates of the KL divergence, and it is unclear how good these estimates are of the actual KL divergence (i.e., when computed over the entire distribution). To understand the effectiveness of minimizing the *true* KL, we sample 28K tokens from the AlpacaFarm evaluation data and plot the KL divergence in Figure 5.2 (left). Regardless of the estimator used, the KL divergence increases as the training proceeds. The choice of regularizer impacts the speed in which KL grows: KL divergence increases faster without any regularization (“No regularization”) or when using the Bregman divergence,[7] it grows slower when regularized by the standard KL and the squared error estimators. While removing KL penalty entirely leads to larger KL (e.g., 6 times that of the standard KL estimator at step 180), the resulting model still reaches win rates of $47\% \sim 49\%$, which are higher than that of the released AlpacaFarm PPO checkpoint. That said, the best win rates

[7]In the Bregman divergence penalty, $\frac{\pi_{\mathrm{ref}}(y)}{\pi_\theta(y)}$ is minimized, which encourages $\pi_\theta(y)$ to be large when $\pi_{\mathrm{ref}(y)}$ is small despite the entropy bonus.

are achieved when the KL divergence from the reference policy is neither small nor large (on our evaluation set, around 0.05 to 0.12 per token).

A linear relationship between $\sqrt{\mathbf{KL}(\pi_\theta, \pi_{\mathbf{ref}})}$ and reward exists in our LoRA setup. Previous work [9, 50] demonstrates an approximately linear relationship between $\sqrt{\mathrm{KL}(\pi_\theta, \pi_{\mathrm{ref}})}$ and reward, which suggests that π_θ stays within a small region relative to π_{ref} (i.e., $\pi_{\mathrm{ref}}+\delta\pi_{\mathrm{ref}}$) during PPO training. In Figure 5.3 (right), we confirm that this linear relationship also holds in our LoRA setup. To go beyond the $\delta\pi_{\mathrm{ref}}$ region, we also experiment with a configuration that *maximizes* KL during training. We find that this "negative KL" estimator leads to a region where the linear relationship breaks ($\sqrt{\mathrm{KL}(\pi_\theta, \pi_{\mathrm{ref}})} > 0.4$ in our experiments). In this region, reward either plateaus or starts decreasing instead of linearly increasing. This suggests that the ratio between $\sqrt{\mathrm{KL}(\pi_\theta, \pi_{\mathrm{ref}})}$ and the reward can be a useful metric to monitor for over-optimization during training.

We observe positive correlations between rewards, win rates, response length, and KL in certain regimes. In addition to rewards, we find that win rates and response length also positively correlate with KL within a certain regime (step < 100 and KL < 0.12 in our experiments). Going beyond this region, the average response length does not change much as KL divergence keeps increasing (Figure 5.2, right), whereas win rates start to drop significantly (Figure 5.3, left). This has practical implications on PPO training – early stopping leads to better performance while also saving compute.

PPO training has a larger negative impact on factuality with full model fine-tuning than with LoRA. To quantify the effects of PPO training on the factuality of LLM-generated text, we evaluate several checkpoints using the FActScore metric [121]. FActScore evaluates the factual precision of a language model by breaking a long-form response into a collection of atomic facts and then computing the precision of these atomic facts.[8] Table 5.2 shows that the SFT10k model (without any PPO training) obtains a higher FActScore than any

[8]We use the ChatGPT+retrieval configuration of FActScore. The prompt for all of our experiments is "Tell me a bio of X", and we perform all evaluations on the labeled split of 183 people entities released by [121].

Configuration	**FActScore** (↑)	**# facts per response**	**# tokens per response**
Publicly-released AlpacaFarm checkpoints from [43]			
AlpacaFarm SFT10k	39.7%	19.3	121.1
AlpacaFarm PPO	34.5%	37.1	247.9
Our PPO models trained with LoRA			
LoRA PPO w/ KL	39.4%	26.7	170.8
LoRA PPO w/ Jensen-Shannon	38.2%	30.7	199.1
LoRA PPO w/o regularization	38.4%	33.9	217.5

Table 5.2: Evaluation on the FActScore labeled split [121], which requires each model to generate a biography of 183 people entities. Models fine-tuned with the PPO training objective consistently underperform the SFT10k checkpoint in terms of factual precision. Implementing PPO training with LoRA somewhat mitigates the negative impact on factuality of model-generated text.

checkpoint trained with PPO. Meanwhile, the released AlpacaFarm PPO checkpoint with full-model fine-tuning achieves the lowest FActScore (34.5%), with all of our LoRA-based implementations outperforming it (39.4% for the most comparable configuration). This result suggests that while PPO training can effectively steer the output to those preferred by humans for stylistic reasons (e.g., by increasing response length), it also may hurt the factuality of the generated text, and perhaps LoRA's regularization properties mitigates this effect to some extent.

5.4.3 Discussion

In this section, we conduct an empirical analysis of the last stage of RLHF (PPO training) when implemented with low-rank adaptation, a parameter-efficient fine-tuning method. Our LoRA-based implementation reduces the required hardware from eight to two A100 GPUs. Besides reduced memory consumption, we also observe that LoRA provides an implicit regularization effect during PPO training – good performance can be achieved even when the KL regularization term in the PPO objective is removed. Additionally, we find that alternative regularizers to the standard KL divergence estimator (e.g., Jensen-Shannon divergence) lead to higher win rates. Overall, results demonstrated in this section validate the correctness of our experimental setup and provide solid base for further application of SuffixRL on the LLaMA-1 models.

5.5 SuffixRL experimental setup

Data We leverage the PG-19 [145] dataset to fine-tune LLaMA-1-7B [182] with SuffixRL. Specifically, we randomly sample 10000 books that have at least 30K tokens from PG-19 training set. Among these sampled books, we further randomly sample 8K (prefix, suffix) pairs to train the reward model, and 10K prefixes for the reinforcement learning stage. When preprocessing the training data for reward modeling, we exclude examples that contain the word "gutenberg" in any of the prefix, gold suffix, and model generated suffix. Following the setup in AlpacaFarm, we use nucleus sampling [63] with the hyperparameter $p = 0.9$ to produce model generated suffixes. We also control the gold and model generated suffix to have the same length by truncating from right. These two filters prove to be helpful in our preliminary experiments. Without these filters, the policy can quickly learn to hack the reward model by generating output that overfits to dataset biases.

Models & Evaluation For the reward model, we initialize with LLaMA-1-7B and set LoRA rank to 8 and LoRA α to 64. Training on the filtered examples described above for one epoch with batch size 32 and learning rate $5e-6$ takes around 21 minutes on a single A100 80GB GPU. On a validation set of reward modeling, the final reward model achieves ~ 0.62 accuracy[9] at differentiating gold and model generated continuation. For the reinforcement learning stage, we evaluate two policy models: LLaMA-1-7B and AlpacaSFT10K, the latter being the LLaMA-1-7B model fine-tuned on 10K instruction following examples from the Alpaca dataset [178], which we used in the previous section. For both models, we run PPO for 100 steps with rollout batch size 256, optimization batch size 128, LoRA rank 8 and α

[9]While this is not a sufficiently good accuracy, our results show that a weak reward supervisor like this can still result in significant gains in policy, echoing the recent finding of weak-to-strong generalization [21], i.e., strong capabilities can be elicited with weak supervision (reward) signals.

Model setting	$slen$=16	$slen$=32	$slen$=64	$slen$=128
LLaMA-1-7B + SFT on PG-19	51.2	52.4	51.6	49.7
LLaMA-1-7B + SuffixRL on PG-19	51.8	52.0	52.6	54.4
AlpacaSFT10K	56.0	58.0	60.8	62.2
AlpacaSFT10K + SuffixRL on PG-19	57.4	59.4	62.8	64.6

Table 5.3: Applying SuffixRL leads to more coherent output compared to SFT on the PG-19 data. The gains are consistent and also improve with the increase of suffix length. Instruction fine-tuning (AlpacaSFT10k) also yields coherent and natural continuation; however, it does not saturate the performance, and can be further improved by applying SuffixRL.

64, dropout 0.1, and learning rate $1e-5$, KL coefficient β equals to 0.02. Running 100 PPO steps on two 80GB A100 GPUs takes around 40 hours when the per step batch size during rollout equals to 8. Training time can be significantly reduced when setting a larger rollout step batch size. To evaluate the output quality, we adopt an LLM-based evaluation which have been shown to correlate with human judgements [225, 43]. We evaluate on 800 examples sampled from PG-19 without overlap with any of the reward modeling and PPO training data. Concretely, we instruct OpenAI GPT-3.5-turbo-16K to select a *"more natural and coherent continuation given the prefix"*.

5.6 Results & Analysis

Data control for rewarding modeling. Before showing the final performance of SuffixRL on the PG-19 coherence evaluation, we briefly discuss the harmfulness of not carefully controlling the data quality for reward modeling. Figure 5.4 presents the reward and loss curves throughout the PPO training stage, when training against two reward models, one with proper control of reward modeling data, and one without. The reward model trained on unfiltered data is easily hacked by the policy, as illustrated in the sudden jump in reward between 0 and 20 steps in Figure 5.4 left. While this model converges faster and achieves a smaller loss (Figure 5.4 right), the quality of the policy output deteriorates significantly. Since neither the reward score nor the loss alone can accurately reflect the gains from PPO

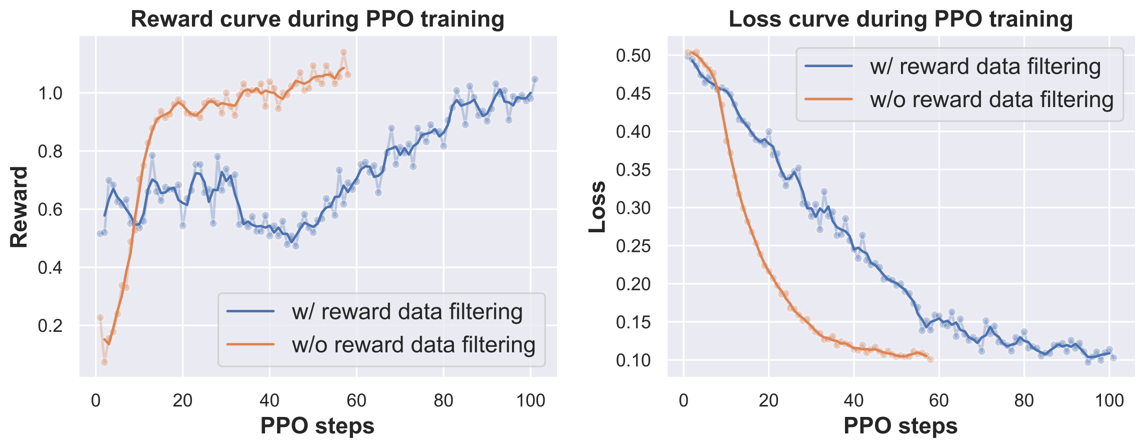


Figure 5.4: The change of reward (**Left**) and loss (**Right**) over the course of PPO training. When the reward model is trained on data without applying any preprocessing filters, the policy can quickly "hack" the reward model, achieving sudden jump in reward scores and decline in losses, whereas significantly degenerated output quality.

training stage, more empirical studies are necessary to further explore, strengthen, and stabilize the RL training stage.

SuffixRL leads to more coherent continuation than supervised fine-tuning. We evaluate the win rate of all models against LLaMA-1-7B at varying continuation lengths ($slen$ in Table 5.3) to verify if the gains are consistent and robust. Note that a LLaMA-1-7B model comparing against itself yields a roughly 50% win rate. After supervised fine-tuning on PG-19 (Table 5.3, LLaMA-1-7B + SFT on PG-19), there exists a marginal improvement in win rate against LLaMA-1-7B; however, the improvement only peaks at the length of 32 tokens. Beyond this point, the performance degrades as the continuation becomes longer. In contrast, applying SuffixRL leads to consistent improvements over LLaMA-1-7B; moreover, the improvement also increases as the length of generated suffixes extends.

SuffixRL yields additional gains after supervised fine-tuning. Another observation is that supervised fine-tuning on instruction following data can significantly improve the coherence. During the instruction fine-tuning stage, the model is trained to generate long and coherent answers given short instructions. This behavior likely generalizes to less constrained open-ended generation adopted in our evaluation setup. After applying SuffixRL on this better performant model, we observe additional gains. Moreover, the gains, similar to what we observed before, also increases with the increase of suffix length. This suggests that the performance gains achieved through SFT did not reach a saturation point, and that SuffixRL can lead to further improvements atop the gains obtained through supervised fine-tuning.

Qualitative comparison of SuffixRL and supervised fine-tuning. While LLM-based win rate evaluation is simple and efficient, it may not reflect the nuances exhibited in the long-form generation. As such, we also qualitatively examine the output of these models, and present a randomly picked example in Table 5.4. The prefix of this example ends with a question regarding the way the bird maintaining stability while descending a tree. LLaMA-1 without any fine-tuning generates a continuation that does not immediately address the question; additionally, it provides a description of "climbing up" the tree, which disrupts the logical connection with the prefix. The two SFT-ed versions (LLaMA-1-7B + SFT and AlpacaSFT10k) exhibit better logical connection to the prefix, whereas they tend to generate hallucinations or simply decline to provide a helpful description in the continuation (as highlighted in red in the table). The two SuffixRL versions not only produce more coherent continuations (e.g., emphasizing the descending process and coming up with plausible causes), but also exhibit a better stylistically consistency with the provided prefix.

5.7 Related work

In this chapter, we propose a new training method SuffixRL, inspired by recent works on aligning large language models (LLMs) to human intents via instruction-following methods [132]. Instruction-following methods can be broadly categorized into (1) those that just perform supervised fine-tuning and (2) those that additionally apply reinforcement learning.

Methods based on upervised fine-tuning. Methods in the first category differ in how they source instruction-following data for supervised fine-tuning. The instruction-response

Prefix	... Surely it is worth one's while to hie early to the haunts of the birds to hear such a tumult of song. One spring I made up my mind to make a closer study than ever of the dainty creeping warbler, wishing to know just how he contrives to scuttle up and down the boles and branches of the trees with so much ease and grace. He is the only warbler we have in eastern North America that makes a habit of scaling the tree trunks and descending them head downward. How does he do this? The muscles of his legs and pelvis are as elastic as India rubber, so that he can twist and twirl about in a marvelous way, pointing his head one moment to the east and the next, without losing his hold, in the opposite direction. He is able to swing himself around almost as if he were hung on a pivot. But how does he hold himself on his shaggy wall as he hitches head **downward**?
LLaMA-1-7B	The muscles of his legs are like springs, but his feet are too small to grasp the bark. He has to have a foot-hold, and he finds it in the form of a round projection on a tree trunk, a few inches in diameter, and about as high as his head. This is a natural protuberance formed in the wood by the pruning of the outer bark; and the bird takes advantage of it as a foothold, just as we do in climbing a ladder, we take hold with our hands and feet, but the tree-climber takes hold with his feet and hands....
LLaMA-1-7B + SFT	This is the really marvelous part of his performance. He has a kind of sucker-foot on the sole of each of his hind feet, something like the sucker-foot of the frog, and with this he can adhere to the trunk as well as a leech. But there is still another method by which he is able to ascend and descend the tree trunks, and that is by means of his outer toe, which has a powerful hook on its end. This is the toe, I believe, that is sometimes called the "hooligan toe."
LLaMA-1-7B + SuffixRL	His claws are very strong, and the muscles of his legs, back, and breast are flexible. In some cases he has a second claw on each toe, which helps him to hold on. But the most important part of the secret is that he is able to grasp the bark with his toes so as to feel every little bump and hollow, and thus he knows just when to grasp with his feet and when to let go and spring up. It is a wonderful thing to watch a bird climbing a tree with his feet, but it is still more wonderful to watch a bird descending a tree with his feet. He feels his way by means of his toes, ...
AlpacaSFT10k	He has two powerful claws on the ends of each foot; and, so adjusted, he can cling to the smallest twig or the smallest knot in the wood. It is difficult to understand how he can do this, for his claws are not curved like those of the chickadee, and his toes are not strong enough to grasp a hold. But he makes himself useful in many ways. He picks up the leaves and twigs, which fall before the storm, and carries them to the ground. In short, he is one of the most useful birds in the woods.
AlpacaSFT10k + SuffixRL	With the claws of his strong feet, which are furnished with sharp, curved nails. He can easily grasp the bark, and with his toes he can spread himself fairly well, so as to be able to steady himself. He does not hang by his claws, but by the web between them, which is so strong that it is almost impossible to tear it away. And so the dainty warbler makes his way through the branches with the most wonderful ease and grace, never making a false step, and always keeping his song going. He has the reputation of being a shy bird, but I have never found this to be the case.

Table 5.4: Randomly sampled example from the PG-19 coherence evaluation set. Spans that do not contribute to the coherence of the continuation are highlighted in red, while those contributing to coherence are highlighted in teal.

pairs can be generated from existing (close-sourced) models [178, 64, 124, 197], curated by humans [97, 226], or even curated by LLMs [106]. Open-source models, when fine-tuned on high quality [226] and diverse data [196], can compete with blackbox LLMs on certain evaluations.

Methods based on reinforcement learning. RL-based methods [229, 132] align LLMs via online policy rollout and optimization. In contrast to SFT-based methods, RL incorporates feedback collected from either humans [169, 132] or AI [10, 101]. These judgments are distilled into a preference (reward) model for evaluating responses during online policy rollout with (sparse) rewards [206], and the whole process is more involved than the single-stage teacher-forced training used in SFT. While most RL-based methods depend on pairwise preference judgments [43], feedback can take other forms including natural language [151, 47]. Recently, [12] show that human feedback protocol (e.g., rating or ranking) has significant impact on the evaluation of aligned LLMs. RL-based methods are typically regularized by a distributional term (e.g., a KL divergence penalty) to avoid degeneration caused by large deviations from a reference model. Previous work [90] shows that KL-regularized RL can be viewed as Bayesian inference. Minimizing KL divergence is also related to distribution matching (DM) methods [83], where the target optimal distribution is available. Recently, [89] introduced KL-regularized RL from the perspective of DM, and [52] propose a framework that unifies KL-regularized RL, DM, and other f-divergence minimization methods.

LLM-based evaluation. In this chapter, instead of using human evaluation, which can be costly and time-consuming to obtain, we adopt LLM-based evaluation. This evaluation approach involves leveraging large language models fine-tuned with human feedback to provide judgments for the given text. Recent studies [43, 225, 112] have shown a high correlation between LLM-based evaluation and human judgments, suggesting that models tuned with human feedback data can effectively capture human values, and reflect them in simple evaluation tasks. However, it is important to note that existing LLMs, when not adequately calibrated, may still exhibit biases, including position/ordering bias, length bias, and self-enhancement bias [225]. Another limitation of LLM-based evaluation arises in its validation primarily on short sequence tasks. Chapter 3 and recent studies [109, 23] provide evidence on the limitations of LLMs when evaluating on long sequences. Experiments in this chapter, however, are restricted to short sequences due to limited compute budget, so the evaluation falls within the valid range of LLMs' instruction-following capability.

5.8 Conclusion

In this chapter, inspired by the connection between SuffixLM and recent RLHF framework, we introduce a new fine-tuning paradigm SuffixRL to improve generation quality. SuffixRL consists of two stages: reward modeling and reinforcement learning. Experiments on PG-19 shows that models trained with SuffixRL can generate more coherent continuation compared to models trained with standard supervised fine-tuning. Due to limited compute resources, we apply low-rank adaptation to train the models of 7B parameters, and constrain the sequence length to be at most 512. More experiments should be done to verify if the improvement is consistent when scaling up the model size and context size, and when employing full-model fine-tuning. Overall, this is an exciting new direction that allows training of autoregressive language models with segment-level signals.

Limitation

Due to constraints in compute resources, we only report the LoRA-based fine-tuning results without experimenting with full-model fine-tuning. It remains to be seen if alternative regularization approach is necessary for achieving better performance with full-model fine-tuning using the SuffixRL objective. Another limitation pertains to the model and context size. Despite being more than twenty times larger than models we explored in Chapter 2, models with 7B parameters fall still on the smaller end of the spectrum, e.g., compared to models of 70 billion parameters. Additionally, limited GPU memory prevented us from training SuffixRL with longer sequences beyond 512 tokens. Nevertheless, SuffixRL is a promising new method that enables training token-level language models with segment-level signals, the prediction of which has been shown to depend more on long-range context compared to single subword token in Chapter 3. Therefore, scaling the context size for SuffixRL with more recent sequence parallelism [74] can be an interesting direction to explore in the future. Finally, in this chapter, we only focus on PPO training objective. A few recent works propose simplified variants such as DPO [146] and IPO [8]; further experimental exploration is required to determine if SuffixRL can be applied to these alternative variants.

Ethics Statement

In this section, we deal with more powerful large language models than previous chapters. These models not only are of larger sizes but also have undergone alignment processes. These models, when not properly aligned, can generate harmful content, disinformation, and information which can be exploited for malicious uses. While this chapter primarily delves into the algorithmic and experimental aspects of language models, it is crucial to pay attention to the potential risks and take necessary precautions. In the era of large language models, the amount of compute as well as the carbon footprint of total experimental setup can be alarming. While this chapter is experimentally heavy, we would like to emphasize that the low-rank adaptation method and alternative regularization method both contribute to more efficient fine-tuning with faster convergence, thereby aiding the reduction of energy consumption.

CHAPTER 6
FUTURE DIRECTIONS

Thus far, we have scrutinized the utilization of long-range context in language models (Chapter 2), demonstrated the benefits of segment-level language models (Chapter 3), introduced a new prompting framework for reasoning over long documents(Chapter 4), and a new method to train token-level language model with segment-level signals (Chapter 5). In this chapter, we provide three general directions and seven concrete directions that can be fruitful to explore in the future.

6.1 Follow-up works of SuffixRL

Scaling up context size In Chapter 5, due to memory constraints, we only apply SuffixRL on short sequences. An immediate future direction of SuffixRL thus involves scaling up the context size. As shown in Chapter 3, segment-level signals prove to be especially helpful for tasks that require reasoning over long-range context. Therefore, it is natural to ask if applying SuffixRL, a training method based on segment-level signals, would bring further gains on tasks such as CHAPTERBREAK when the context size is increased. Apart from the substantial compute required, one obstacle for this direction is the inherently sparse signals that the policy obtains from the reward model. This issue becomes severe as the suffix becomes longer – a single reward signal for a long suffix segment may not be informative enough for steering the policy to the desired direction. As such, potential solutions such as fine-grained reward signals can be especially promising to explore in the future.

Applying SuffixRL to improve factuality Recall that in SuffixRL, the reward model is trained to favor the gold suffix while dispreferring the model generated output. While large language model nowadays can generate text indistinguishable from human written text, they are prone to hallucinations [69], and often generate output not grounded in existing facts [126] or information given in the context. SuffixRL can potentially help alleviate this issue by allowing models to disprefer its own unfaithful output. Previous research has shown that model tends to hallucinate more frequently about long-tail knowledge even if it is present in the pre-training data [79]. Thus, it would be interesting to apply SuffixRL to data covering the long-tail knowledge, and encourage the model to disprefer its potentially hallucinated output during PPO training. This experiment is not costly, because unlike the standard RLHF, SuffixRL does not require the access to expensive, high-quality human feedback data.

6.2 Better evaluations of long-range language models

Long-form instruction following dataset As discussed briefly in Chapter 3, existing evaluation of long-range language models are flawed: tests such as passkey and needle in the hay retrieval only evaluate block-level retrieval from context, while tasks such as CHAPTERBREAK, which do rely on comprehensive understanding of long context, are too "short" for some of nowadays large language models. With the development of instruction fine-tuning, it can be rewarding to develop instruction following datasets tailored for long-context, long-form generation tasks. An ideal long-form instruction following dataset should (1) cover diverse domains and task types (i.e., both short context retrieval and full context comprehension), (2) have sufficient length, and (3) involve more complex instructions than existing short instruction-following datasets. While it is tempting to distill proprietary models for constructing such a dataset, it is however safer to cleverly use human-written data, and perform instruction back-translation [106] considering that the proprietary models still have upper-limit when dealing with extremely complex instructions.

Human-machine collaborative evaluation of long sequence task Human evaluation has traditionally been the gold standard for evaluating language generation tasks due to its reliability. However, human evaluation can be costly and may not be reliable enough [80] with regard to long-form generation. Given recent developments in LLM-based evaluation, it is intriguing to delve into human-machine collaborative evaluation, or machine-guided evaluation of long sequence task for reduced cost. Many open questions remain to be answered, such as: “What is the optimal approach for dividing the evaluation task between machine and humans, and how can the results from each component be effectively integrated?”, “How reliable and consistent is the human-machine collaborative evaluation?” and “How does the calibration of LLMs and the dynamic interaction between human and machine affect the validity and fairness of the evaluation?”. Resolving these problems can be instrumental for building standardized long-context language model evaluation benchmarks [4, 105], where obtaining large scale high-quality annotation data is currently very expensive and sometimes unreliable.

6.3 Better training and architectures for long-range foundation models

Including other modalities of context As mentioned in Chapter 1, this thesis mainly focuses on textual context, which, in its current form, conveys quite limited information. Including other types of textual context and other modalities of context can potentially enhance the capability of recent large foundation models. Examples of alternative textual contexts include spatiotemporal co-ordinates associating with the main event in the document, preceding and subsequent activities, language-events taking place in the vicinity [117], etc. Other modalities such as speech, image, and video content can not only help resolve ambiguities in unimodal context but also provide more instances for the LMs to associate and ground concepts. Adding these additional form of contexts requires the model to handle even larger context window, and is impossible without long-range foundation models. An immediate challenge in developing such models is properly mixing signals from various modalities. Recent research has delved into such topic [1], but there is still a long way to go to create a powerful model capable of consuming diverse types of data similar to how humans do.

Learned position embeddings Recent research on long-range language models usually adjusts position embeddings, e.g., position interpolation [24] or adjust base frequency [209] in rotary position embeddings (RoPE), before fine-tuning a short-range language model into a long-range one. However, as the sequence length increases, the long-term decay effect in RoPE makes the attention competition between irrelevant nearby tokens and relevant far-away tokens more severe. A straightforward approach to alleviating this issue is to adopt a RoPE-based learned position embedding. Alongside recent progress in sequence parallellism [74], the learned position embeddings can be promising to explore with context sizes on the order of millions of tokens.

Combining SuffixRL with hierarchical architectures and curriculum learning SuffixLM, as proposed in Chapter 3, uses a hierarchical structure, segmenting long sequences into smaller chunks and encoding each chunk using an encoder. This hierarchical inductive bias can be complementary and more conducive to the utilization of segment-level signals, when scaling the context to hundreds of thousands of tokens. Additionally, another underexplored direction in long-range context modeling is curriculum learning, where the model progressively learns to incorporate longer effective context ranges during training. However, it remains an open question whether this training approach can ease the fine-tuning from short to long-range language models.

BIBLIOGRAPHY

[1] Aghajanyan, Armen, Huang, Bernie, Ross, Candace, Karpukhin, Vladimir, Xu, Hu, Goyal, Naman, Okhonko, Dmytro, Joshi, Mandar, Ghosh, Gargi, Lewis, Mike, and Zettlemoyer, Luke. Cm3: A causal masked multimodal model of the internet, 2022.

[2] Ainslie, Joshua, Ontanon, Santiago, Alberti, Chris, Cvicek, Vaclav, Fisher, Zachary, Pham, Philip, Ravula, Anirudh, Sanghai, Sumit, Wang, Qifan, and Yang, Li. Etc: Encoding long and structured inputs in transformers, 2020.

[3] Albrecht, Jason E., and Myers, Jerome L. Role of context in accessing distant information during reading. *Journal of experimental psychology. Learning, memory, and cognition 21 6* (1995), 1459–68.

[4] An, Chenxin, Gong, Shansan, Zhong, Ming, Zhao, Xingjian, Li, Mukai, Zhang, Jun, Kong, Lingpeng, and Qiu, Xipeng. L-eval: Instituting standardized evaluation for long context language models, 2023.

[5] Anil, Cem, Wu, Yuhuai, Andreassen, Anders Johan, Lewkowycz, Aitor, Misra, Vedant, Ramasesh, Vinay Venkatesh, Slone, Ambrose, Gur-Ari, Guy, Dyer, Ethan, and Neyshabur, Behnam. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems* (2022), Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, Eds.

[6] Anil, Rohan, Dai, Andrew M., Firat, Orhan, Johnson, Melvin, Lepikhin, Dmitry, Passos, Alexandre, Shakeri, Siamak, Taropa, Emanuel, Bailey, Paige, Chen, Zhifeng, Chu, Eric, Clark, Jonathan H., Shafey, Laurent El, Huang, Yanping, Meier-Hellstern, Kathy, Mishra, Gaurav, Moreira, Erica, Omernick, Mark, Robinson, Kevin, Ruder, Sebastian, Tay, Yi, Xiao, Kefan, Xu, Yuanzhong, Zhang, Yujing, Abrego, Gustavo Hernandez, Ahn, Junwhan, Austin, Jacob, Barham, Paul, Botha, Jan, Bradbury, James, Brahma, Siddhartha, Brooks, Kevin, Catasta, Michele, Cheng, Yong, Cherry, Colin, Choquette-Choo, Christopher A., Chowdhery, Aakanksha, Crepy, Clément, Dave, Shachi, Dehghani, Mostafa, Dev, Sunipa, Devlin, Jacob, Díaz, Mark, Du, Nan, Dyer, Ethan, Feinberg, Vlad, Feng, Fangxiaoyu, Fienber, Vlad, Freitag, Markus, Garcia, Xavier, Gehrmann, Sebastian, Gonzalez, Lucas, Gur-Ari, Guy, Hand, Steven, Hashemi, Hadi, Hou, Le, Howland, Joshua, Hu, Andrea, Hui, Jeffrey, Hurwitz, Jeremy, Isard, Michael, Ittycheriah, Abe, Jagielski, Matthew, Jia, Wenhao, Kenealy, Kathleen, Krikun, Maxim, Kudugunta, Sneha, Lan, Chang, Lee, Katherine, Lee, Benjamin, Li, Eric, Li, Music, Li, Wei, Li, YaGuang, Li, Jian, Lim, Hyeontaek, Lin, Hanzhao, Liu, Zhongtao, Liu, Frederick, Maggioni, Marcello, Mahendru, Aroma, Maynez, Joshua, Misra, Vedant, Moussalem, Maysam, Nado, Zachary, Nham, John, Ni, Eric, Nystrom, Andrew, Parrish, Alicia, Pellat, Marie, Polacek, Martin, Polozov, Alex, Pope, Reiner, Qiao, Siyuan, Reif, Emily, Richter, Bryan, Riley, Parker, Ros, Alex Castro, Roy, Aurko, Saeta, Brennan, Samuel, Rajkumar, Shelby, Renee, Slone, Ambrose, Smilkov, Daniel, So, David R., Sohn, Daniel, Tokumine, Simon, Valter, Dasha, Vasudevan, Vijay, Vodrahalli, Kiran, Wang, Xuezhi, Wang, Pidong, Wang, Zirui, Wang, Tao, Wieting, John, Wu, Yuhuai, Xu, Kelvin, Xu, Yunhan, Xue, Linting, Yin, Pengcheng, Yu, Jiahui, Zhang, Qiao, Zheng, Steven, Zheng, Ce, Zhou, Weikang, Zhou, Denny, Petrov, Slav, and Wu, Yonghui. Palm 2 technical report, 2023.

[7] Anthropic. Long context prompting for claude 2.1, 2023.

[8] Azar, Mohammad Gheshlaghi, Rowland, Mark, Piot, Bilal, Guo, Daniel, Calandriello, Daniele, Valko, Michal, and Munos, Rémi. A general theoretical paradigm to understand learning from human preferences, 2023.

[9] Bai, Yuntao, Jones, Andy, Ndousse, Kamal, Askell, Amanda, Chen, Anna, DasSarma, Nova, Drain, Dawn, Fort, Stanislav, Ganguli, Deep, Henighan, Tom, Joseph, Nicholas, Kadavath, Saurav, Kernion, Jackson, Conerly, Tom, El-Showk, Sheer, Elhage, Nelson, Hatfield-Dodds, Zac, Hernandez, Danny, Hume, Tristan, Johnston, Scott, Kravec, Shauna, Lovitt, Liane, Nanda, Neel, Olsson, Catherine, Amodei, Dario, Brown, Tom, Clark, Jack, McCandlish, Sam, Olah, Chris, Mann, Ben, and Kaplan, Jared. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

[10] Bai, Yuntao, Kadavath, Saurav, Kundu, Sandipan, Askell, Amanda, Kernion, Jackson, Jones, Andy, Chen, Anna, Goldie, Anna, Mirhoseini, Azalia, McKinnon, Cameron, Chen, Carol, Olsson, Catherine, Olah, Christopher, Hernandez, Danny, Drain, Dawn, Ganguli, Deep, Li, Dustin, Tran-Johnson, Eli, Perez, Ethan, Kerr, Jamie, Mueller, Jared, Ladish, Jeffrey, Landau, Joshua, Ndousse, Kamal, Lukosuite, Kamile, Lovitt, Liane, Sellitto, Michael, Elhage, Nelson, Schiefer, Nicholas, Mercado, Noemi, DasSarma, Nova, Lasenby, Robert, Larson, Robin, Ringer, Sam, Johnston, Scott, Kravec, Shauna, Showk, Sheer El, Fort, Stanislav, Lanham, Tamera, Telleen-Lawton, Timothy, Conerly, Tom, Henighan, Tom, Hume, Tristan, Bowman, Samuel R., Hatfield-Dodds, Zac, Mann, Ben, Amodei, Dario, Joseph, Nicholas, McCandlish, Sam, Brown, Tom, and Kaplan, Jared. Constitutional ai: Harmlessness from ai feedback, 2022.

[11] Bai, Yushi, Lv, Xin, Zhang, Jiajie, Lyu, Hongchang, Tang, Jiankai, Huang, Zhidian, Du, Zhengxiao, Liu, Xiao, Zeng, Aohan, Hou, Lei, Dong, Yuxiao, Tang, Jie, and Li, Juanzi. Longbench: A bilingual, multitask benchmark for long context understanding, 2023.

[12] Bansal, Hritik, Dang, John, and Grover, Aditya. Peering through preferences: Unraveling feedback acquisition for aligning large language models, 2023.

[13] Beltagy, Iz, Peters, Matthew E., and Cohan, Arman. Longformer: The long-document transformer, 2020.

[14] Bender, Emily M., and Koller, Alexander. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 5185–5198.

[15] Bengio, Yoshua, Ducharme, R., Vincent, Pascal, and Janvin, Christian. A neural probabilistic language model. *J. Mach. Learn. Res. 3* (2003), 1137–1155.

[16] Borgeaud, Sebastian, Mensch, Arthur, Hoffmann, Jordan, Cai, Trevor, Rutherford, Eliza, Millican, Katie, Driessche, George van den, Lespiau, Jean-Baptiste, Damoc, Bogdan, Clark, Aidan, et al. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426* (2021).

[17] Bradley, Ralph Allan, and Terry, Milton E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika 39*, 3/4 (1952), 324–345.

[18] Brown, Peter F., Della Pietra, Vincent J., deSouza, Peter V., Lai, Jenifer C., and Mercer, Robert L. Class-based *n*-gram models of natural language. *Computational Linguistics 18*, 4 (1992), 467–480.

[19] Brown, Tom, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared D, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, Agarwal, Sandhini, Herbert-Voss, Ariel, Krueger, Gretchen, Henighan, Tom, Child, Rewon, Ramesh, Aditya, Ziegler, Daniel, Wu, Jeffrey, Winter, Clemens, Hesse, Chris, Chen, Mark, Sigler, Eric, Litwin, Mateusz, Gray, Scott, Chess, Benjamin, Clark, Jack, Berner, Christopher, McCandlish, Sam, Radford, Alec, Sutskever, Ilya, and Amodei, Dario. Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 1877–1901.

[20] Brown, Tom B, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[21] Burns, Collin, Izmailov, Pavel, Kirchner, Jan Hendrik, Baker, Bowen, Gao, Leo, Aschenbrenner, Leopold, Chen, Yining, Ecoffet, Adrien, Joglekar, Manas, Leike, Jan, Sutskever, Ilya, and Wu, Jeff. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023.

[22] Caciularu, Avi, Dagan, Ido, Goldberger, Jacob, and Cohan, Arman. Long context question answering via supervised contrastive learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Seattle, United States, July 2022), Association for Computational Linguistics, pp. 2872–2879.

[23] Chang, Yapei, Lo, Kyle, Goyal, Tanya, and Iyyer, Mohit. Booookscore: A systematic exploration of book-length summarization in the era of llms, 2023.

[24] Chen, Shouyuan, Wong, Sherman, Chen, Liangjian, and Tian, Yuandong. Extending context window of large language models via positional interpolation, 2023.

[25] Chen, Tianqi, Xu, Bing, Zhang, Chiyuan, and Guestrin, Carlos. Training deep nets with sublinear memory cost, 2016.

[26] Chen, Wenhu, Ma, Xueguang, Wang, Xinyi, and Cohen, William W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588* (2022).

[27] Chiang, Wei-Lin, Li, Zhuohan, Lin, Zi, Sheng, Ying, Wu, Zhanghao, Zhang, Hao, Zheng, Lianmin, Zhuang, Siyuan, Zhuang, Yonghao, Gonzalez, Joseph E., Stoica, Ion, and Xing, Eric P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[28] Child, Rewon, Gray, Scott, Radford, Alec, and Sutskever, Ilya. Generating long sequences with sparse transformers, 2019.

[29] Choromanski, Krzysztof Marcin, Likhosherstov, Valerii, Dohan, David, Song, Xingyou, Gane, Andreea, Sarlos, Tamas, Hawkins, Peter, Davis, Jared Quincy, Mohiuddin, Afroz, Kaiser, Lukasz, Belanger, David Benjamin, Colwell, Lucy J, and Weller, Adrian. Rethinking attention with performers. In *International Conference on Learning Representations* (2021).

[30] Chung, Hyung Won, Hou, Le, Longpre, Shayne, Zoph, Barret, Tay, Yi, Fedus, William, Li, Yunxuan, Wang, Xuezhi, Dehghani, Mostafa, Brahma, Siddhartha, Webson, Albert, Gu, Shixiang Shane, Dai, Zhuyun, Suzgun, Mirac, Chen, Xinyun, Chowdhery, Aakanksha, Castro-Ros, Alex, Pellat, Marie, Robinson, Kevin, Valter, Dasha, Narang, Sharan, Mishra, Gaurav, Yu, Adams, Zhao, Vincent, Huang, Yanping, Dai, Andrew, Yu, Hongkun, Petrov, Slav, Chi, Ed H., Dean, Jeff, Devlin, Jacob, Roberts, Adam, Zhou, Denny, Le, Quoc V., and Wei, Jason. Scaling instruction-finetuned language models, 2022.

[31] Cobbe, Karl, Kosaraju, Vineet, Bavarian, Mohammad, Chen, Mark, Jun, Heewoo, Kaiser, Lukasz, Plappert, Matthias, Tworek, Jerry, Hilton, Jacob, Nakano, Reiichiro, Hesse, Christopher, and Schulman, John. Training verifiers to solve math word problems, 2021.

[32] Correia, Gonçalo M., Niculae, Vlad, and Martins, André F. T. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 2174–2184.

[33] Coste, Thomas, Anwar, Usman, Kirk, Robert, and Krueger, David. Reward model ensembles help mitigate overoptimization, 2023.

[34] Dai, Zihang, Yang, Zhilin, Yang, Yiming, Carbonell, Jaime, Le, Quoc, and Salakhutdinov, Ruslan. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 2978–2988.

[35] Dao, Tri, Fu, Daniel Y., Ermon, Stefano, Rudra, Atri, and Ré, Christopher. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

[36] Dasigi, Pradeep, Lo, Kyle, Beltagy, Iz, Cohan, Arman, Smith, Noah A., and Gardner, Matt. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Online, June 2021), Association for Computational Linguistics, pp. 4599–4610.

[37] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186.

[38] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[39] Dey, Nolan, Soboleva, Daria, Al-Khateeb, Faisal, Yang, Bowen, Pathria, Ribhu, Khachane, Hemant, Muhammad, Shaheer, Zhiming, Chen, Myers, Robert, Steeves, Jacob Robert, Vassilieva, Natalia, Tom, Marvin, and Hestness, Joel. Btlm-3b-8k: 7b parameter performance in a 3b parameter model, 2023.

[40] Dong, Zican, Tang, Tianyi, Li, Junyi, Zhao, Wayne Xin, and Wen, Ji-Rong. Bamboo: A comprehensive benchmark for evaluating long text modeling capacities of large language models, 2023.

[41] Du, Mengnan, He, Fengxiang, Zou, Na, Tao, Dacheng, and Hu, Xia. Shortcut learning of large language models in natural language understanding, 2023.

[42] Dua, Dheeru, Gupta, Shivanshu, Singh, Sameer, and Gardner, Matt. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (Abu Dhabi, United Arab Emirates, Dec. 2022), Association for Computational Linguistics, pp. 1251–1265.

[43] Dubois, Yann, Li, Xuechen, Taori, Rohan, Zhang, Tianyi, Gulrajani, Ishaan, Ba, Jimmy, Guestrin, Carlos, Liang, Percy, and Hashimoto, Tatsunori B. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

[44] Eisenstein, Jacob. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (Boulder, Colorado, June 2009), Association for Computational Linguistics, pp. 353–361.

[45] Eisenstein, Jacob, Nagpal, Chirag, Agarwal, Alekh, Beirami, Ahmad, D'Amour, Alex, Dvijotham, DJ, Fisch, Adam, Heller, Katherine, Pfohl, Stephen, Ramachandran, Deepak, Shaw, Peter, and Berant, Jonathan. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking, 2023.

[46] Fan, Angela, Jernite, Yacine, Perez, Ethan, Grangier, David, Weston, Jason, and Auli, Michael. Eli5: Long form question answering, 2019.

[47] Fernandes, Patrick, Madaan, Aman, Liu, Emmy, Farinhas, António, Martins, Pedro Henrique, Bertsch, Amanda, de Souza, José G. C., Zhou, Shuyan, Wu, Tongshuang, Neubig, Graham, and Martins, André F. T. Bridging the gap: A survey on integrating (human) feedback for natural language generation, 2023.

[48] Fielding, Henry. *The History of the Adventures of Joseph Andrews..*, vol. 1. J. Fr. Valade, 1779.

[49] flozi00. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

[50] Gao, Leo, Schulman, John, and Hilton, Jacob. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning* (23–29 Jul 2023), Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, Eds., vol. 202 of *Proceedings of Machine Learning Research*, PMLR, pp. 10835–10866.

[51] Geng, Xinyang, Gudibande, Arnav, Liu, Hao, Wallace, Eric, Abbeel, Pieter, Levine, Sergey, and Song, Dawn. Koala: A dialogue model for academic research. Blog post, April 2023.

[52] Go, Dongyoung, Korbak, Tomasz, Kruszewski, Germán, Rozen, Jos, Ryu, Nahyeon, and Dymetman, Marc. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215* (2023).

[53] Graesser, Arthur C, Singer, Murray, and Trabasso, Tom. Constructing inferences during narrative text comprehension. *Psychological review 101*, 3 (1994), 371.

[54] Grimshaw, Jane. *Argument structure.* the MIT Press, 1990.

[55] Grosz, Barbara J., Joshi, Aravind K., and Weinstein, Scott. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics 21*, 2 (1995), 203–225.

[56] Gu, Albert, and Dao, Tri. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

[57] Gupta, Ankit, and Berant, Jonathan. Gmat: Global memory augmentation for transformers. *arXiv preprint arXiv:2006.03274* (2020).

[58] Haas, W. J.r. firth: Papers in linguistics, 1934–1951. xii, 233 pp., 11 plates. london, etc.: Oxford university press, 1957. 35s. *Bulletin of the School of Oriental and African Studies 21*, 3 (1958), 668–671.

[59] Hawthorne, Curtis, Jaegle, Andrew, Cangea, Cătălina, Borgeaud, Sebastian, Nash, Charlie, Malinowski, Mateusz, Dieleman, Sander, Vinyals, Oriol, Botvinick, Matthew, Simon, Ian, et al. General-purpose, long-context autoregressive modeling with perceiver ar. *arXiv preprint arXiv:2202.07765* (2022).

[60] Hobbs, Jerry R. Coherence and coreference. *Cognitive science 3*, 1 (1979), 67–90.

[61] Hofstätter, Sebastian, Zamani, Hamed, Mitra, Bhaskar, Craswell, Nick, and Hanbury, Allan. Local self-attention over long text for efficient document retrieval, 2020.

[62] Holtzman, Ari, Buys, Jan, Du, Li, Forbes, Maxwell, and Choi, Yejin. The curious case of neural text degeneration. In *International Conference on Learning Representations* (2020).

[63] Holtzman, Ari, Buys, Jan, Du, Li, Forbes, Maxwell, and Choi, Yejin. The curious case of neural text degeneration, 2020.

[64] Honovich, Or, Scialom, Thomas, Levy, Omer, and Schick, Timo. Unnatural instructions: Tuning language models with (almost) no human labor, 2022.

[65] Houlsby, Neil, Giurgiu, Andrei, Jastrzebski, Stanislaw, Morrone, Bruna, De Laroussilhe, Quentin, Gesmundo, Andrea, Attariyan, Mona, and Gelly, Sylvain. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning* (2019), PMLR, pp. 2790–2799.

[66] Hu, Edward J., Shen, Yelong, Wallis, Phillip, Allen-Zhu, Zeyuan, Li, Yuanzhi, Wang, Shean, Wang, Lu, and Chen, Weizhu. Lora: Low-rank adaptation of large language models, 2021.

[67] Huang, Jiaxin, Gu, Shixiang Shane, Hou, Le, Wu, Yuexin, Wang, Xuezhi, Yu, Hongkun, and Han, Jiawei. Large language models can self-improve. *arXiv preprint arXiv:2210.11610* (2022).

[68] Huang, Jie, and Chang, Kevin Chen-Chuan. Towards reasoning in large language models: A survey, 2022.

[69] Huang, Lei, Yu, Weijiang, Ma, Weitao, Zhong, Weihong, Feng, Zhangyin, Wang, Haotian, Chen, Qianglong, Peng, Weihua, Feng, Xiaocheng, Qin, Bing, and Liu, Ting. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023.

[70] Huang, Luyang, Cao, Shuyang, Parulian, Nikolaus, Ji, Heng, and Wang, Lu. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Online, June 2021), Association for Computational Linguistics, pp. 1419–1436.

[71] Hutchins, DeLesley S., Schlag, Imanol, Wu, Yuhuai, Dyer, Ethan, and Neyshabur, Behnam. Block-recurrent transformers. *ArXiv abs/2203.07852* (2022).

[72] Ippolito, Daphne, Grangier, David, Eck, Douglas, and Callison-Burch, Chris. Toward better storylines with sentence-level language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 7472–7478.

[73] Ireland, KR. Towards a grammar of narrative sequence: The model of the french lieutenant's woman. *Poetics today 7*, 3 (1986), 397–420.

[74] Jacobs, Sam Ade, Tanaka, Masahiro, Zhang, Chengming, Zhang, Minjia, Song, Shuaiwen Leon, Rajbhandari, Samyam, and He, Yuxiong. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models, 2023.

[75] Ji, Yangfeng, Cohn, Trevor, Kong, Lingpeng, Dyer, Chris, and Eisenstein, Jacob. Document context language models. *arXiv preprint arXiv:1511.03962* (2015).

[76] Jiang, Albert Q., Sablayrolles, Alexandre, Mensch, Arthur, Bamford, Chris, Chaplot, Devendra Singh, de las Casas, Diego, Bressand, Florian, Lengyel, Gianna, Lample, Guillaume, Saulnier, Lucile, Lavaud, Lélio Renard, Lachaux, Marie-Anne, Stock, Pierre, Scao, Teven Le, Lavril, Thibaut, Wang, Thomas, Lacroix, Timothée, and Sayed, William El. Mistral 7b, 2023.

[77] Jiang, Xue, Dong, Yihong, Wang, Lecheng, Fang, Zheng, Shang, Qiwei, Li, Ge, Jin, Zhi, and Jiao, Wenpin. Self-planning code generation with large language models, 2023.

[78] Kamradt, Greg. Needle in a haystack. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`, 2023.

[79] Kandpal, Nikhil, Deng, Haikang, Roberts, Adam, Wallace, Eric, and Raffel, Colin. Large language models struggle to learn long-tail knowledge, 2023.

[80] Karpinska, Marzena, Akoury, Nader, and Iyyer, Mohit. The perils of using Mechanical Turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (Online and Punta Cana, Dominican Republic, Nov. 2021), Association for Computational Linguistics, pp. 1265–1285.

[81] Karpinska, Marzena, and Iyyer, Mohit. Large language models effectively leverage document-level context for literary translation, but critical errors persist, 2023.

[82] Katharopoulos, Angelos, Vyas, Apoorv, Pappas, Nikolaos, and Fleuret, Francois. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning* (2020).

[83] Khalifa, Muhammad, Elsahar, Hady, and Dymetman, Marc. A distributional approach to controlled text generation. In *International Conference on Learning Representations* (2021).

[84] Khandelwal, Urvashi, He, He, Qi, Peng, and Jurafsky, Dan. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 284–294.

[85] Khot, Tushar, Trivedi, Harsh, Finlayson, Matthew, Fu, Yao, Richardson, Kyle, Clark, Peter, and Sabharwal, Ashish. Decomposed prompting: A modular approach for solving complex tasks, 2023.

[86] Kim, Geunwoo, Baldi, Pierre, and McAleer, Stephen. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491* (2023).

[87] Kitaev, Nikita, Kaiser, Lukasz, and Levskaya, Anselm. Reformer: The efficient transformer. In *International Conference on Learning Representations* (2020).

[88] Kočiský, Tomáš, Schwarz, Jonathan, Blunsom, Phil, Dyer, Chris, Hermann, Karl Moritz, Melis, Gábor, and Grefenstette, Edward. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics 6* (2018), 317–328.

[89] Korbak, Tomasz, Elsahar, Hady, Kruszewski, Germán, and Dymetman, Marc. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. In *Advances in Neural Information Processing Systems* (2022), Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, Eds.

[90] Korbak, Tomasz, Perez, Ethan, and Buckley, Christopher. RL with KL penalties is better viewed as Bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022* (Abu Dhabi, United Arab Emirates, Dec. 2022), Association for Computational Linguistics, pp. 1083–1091.

[91] Krishna, Kalpesh, Roy, Aurko, and Iyyer, Mohit. Hurdles to progress in long-form question answering. In *North American Association for Computational Linguistics* (2021).

[92] Krishna, Kalpesh, Roy, Aurko, and Iyyer, Mohit. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Online, June 2021), Association for Computational Linguistics, pp. 4940–4957.

[93] Krishna, Kalpesh, Song, Yixiao, Karpinska, Marzena, Wieting, John Frederick, and Iyyer, Mohit. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems* (2023).

[94] Kryściński, Wojciech, Rajani, Nazneen, Agarwal, Divyansh, Xiong, Caiming, and Radev, Dragomir. Booksum: A collection of datasets for long-form narrative summarization.

[95] Kullback, Solomon, and Leibler, Richard A. On information and sufficiency. *The annals of mathematical statistics 22*, 1 (1951), 79–86.

[96] Kwiatkowski, Tom, Palomaki, Jennimaria, Redfield, Olivia, Collins, Michael, Parikh, Ankur, Alberti, Chris, Epstein, Danielle, Polosukhin, Illia, Devlin, Jacob, Lee, Kenton, Toutanova, Kristina, Jones, Llion, Kelcey, Matthew, Chang, Ming-Wei, Dai, Andrew M., Uszkoreit, Jakob, Le, Quoc, and Petrov, Slav. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics 7* (2019), 452–466.

[97] Köpf, Andreas, Kilcher, Yannic, von Rütte, Dimitri, Anagnostidis, Sotiris, Tam, Zhi-Rui, Stevens, Keith, Barhoum, Abdullah, Duc, Nguyen Minh, Stanley, Oliver, Nagyfi, Richárd, ES, Shahul, Suri, Sameer, Glushkov, David, Dantuluri, Arnav, Maguire, Andrew, Schuhmann, Christoph, Nguyen, Huu, and Mattick, Alexander. Openassistant conversations – democratizing large language model alignment, 2023.

[98] Labov, William, and Waletzky, Joshua. Narrative analysis: Oral versions of personal experience.

[99] Lai, Yi-An, Lalwani, Garima, and Zhang, Yi. Context analysis for pre-trained masked language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (Online, Nov. 2020), Association for Computational Linguistics, pp. 3789–3804.

[100] Lambert, Nathan, and Calandra, Roberto. The alignment ceiling: Objective mismatch in reinforcement learning from human feedback, 2023.

[101] Lee, Harrison, Phatale, Samrat, Mansoor, Hassan, Lu, Kellie, Mesnard, Thomas, Bishop, Colton, Carbune, Victor, and Rastogi, Abhinav. Rlaif: Scaling reinforcement learning from human feedback with ai feedback, 2023.

[102] Lester, Brian, Al-Rfou, Rami, and Constant, Noah. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).

[103] Lewis, Mike, Liu, Yinhan, Goyal, Naman, Ghazvininejad, Marjan, Mohamed, Abdelrahman, Levy, Omer, Stoyanov, Veselin, and Zettlemoyer, Luke. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 7871–7880.

[104] Lewis, Patrick, Perez, Ethan, Piktus, Aleksandra, Petroni, Fabio, Karpukhin, Vladimir, Goyal, Naman, Küttler, Heinrich, Lewis, Mike, Yih, Wen-tau, Rocktäschel, Tim, Riedel, Sebastian, and Kiela, Douwe. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 9459–9474.

[105] Li, Jiaqi, Wang, Mengmeng, Zheng, Zilong, and Zhang, Muhan. Loogle: Can long-context language models understand long contexts?, 2023.

[106] Li, Xian, Yu, Ping, Zhou, Chunting, Schick, Timo, Zettlemoyer, Luke, Levy, Omer, Weston, Jason, and Lewis, Mike. Self-alignment with instruction backtranslation, 2023.

[107] Li, Xiang Lisa, and Liang, Percy. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* (2021).

[108] Li, Yucheng. Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering, 2023.

[109] Liu, Nelson F., Lin, Kevin, Hewitt, John, Paranjape, Ashwin, Bevilacqua, Michele, Petroni, Fabio, and Liang, Percy. Lost in the middle: How language models use long contexts, 2023.

[110] Liu, Siyou, and Zhang, Xiaojun. Corpora for document-level neural machine translation. In *Proceedings of the 12th Language Resources and Evaluation Conference* (Marseille, France, May 2020), European Language Resources Association, pp. 3775–3781.

[111] Liu, Yang, Iter, Dan, Xu, Yichong, Wang, Shuohang, Xu, Ruochen, and Zhu, Chenguang. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023.

[112] Liu, Yang, Iter, Dan, Xu, Yichong, Wang, Shuohang, Xu, Ruochen, and Zhu, Chenguang. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (Singapore, Dec. 2023), Houda Bouamor, Juan Pino, and Kalika Bali, Eds., Association for Computational Linguistics, pp. 2511–2522.

[113] Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, and Stoyanov, Veselin. Roberta: A robustly optimized bert pretraining approach, 2019.

[114] Long, Jieyi. Large language model guided tree-of-thought, 2023.

[115] Lu, Pan, Peng, Baolin, Cheng, Hao, Galley, Michel, Chang, Kai-Wei, Wu, Ying Nian, Zhu, Song-Chun, and Gao, Jianfeng. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842* (2023).

[116] Luong, Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (Lisbon, Portugal, Sept. 2015), Association for Computational Linguistics, pp. 1412–1421.

[117] Lyons, John. *Semantics*. Cambridge University Press, 1977.

[118] Madaan, Aman, Tandon, Niket, Gupta, Prakhar, Hallinan, Skyler, Gao, Luyu, Wiegreffe, Sarah, Alon, Uri, Dziri, Nouha, Prabhumoye, Shrimai, Yang, Yiming, Welleck, Sean, Majumder, Bodhisattwa Prasad, Gupta, Shashank, Yazdanbakhsh, Amir, and Clark, Peter. Self-refine: Iterative refinement with self-feedback, 2023.

[119] Malioutov, Igor, and Barzilay, Regina. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (Sydney, Australia, July 2006), Association for Computational Linguistics, pp. 25–32.

[120] Mikolov, Tomáš, Kombrink, Stefan, Burget, Lukáš, Černockỳ, Jan, and Khudanpur, Sanjeev. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (2011), IEEE, pp. 5528–5531.

[121] Min, Sewon, Krishna, Kalpesh, Lyu, Xinxi, Lewis, Mike, Yih, Wen-tau, Koh, Pang Wei, Iyyer, Mohit, Zettlemoyer, Luke, and Hajishirzi, Hannaneh. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251* (2023).

[122] Mohtashami, Amirkeivan, and Jaggi, Martin. Landmark attention: Random-access infinite context length for transformers. In *Workshop on Efficient Systems for Foundation Models @ ICML2023* (2023).

[123] Mooney, Raymond J, and DeJong, Gerald. Learning schemata for natural language processing. In *IJCAI* (1985), pp. 681–687.

[124] Mukherjee, Subhabrata, Mitra, Arindam, Jawahar, Ganesh, Agarwal, Sahaj, Palangi, Hamid, and Awadallah, Ahmed. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

[125] Myers, Jerome, O'Brien, Edward, Albrecht, Jason, and Mason, Robert. Maintaining global coherence during reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition 20* (07 1994), 876–886.

[126] Mündler, Niels, He, Jingxuan, Jenko, Slobodan, and Vechev, Martin. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation, 2023.

[127] Nye, Maxwell, Andreassen, Anders Johan, Gur-Ari, Guy, Michalewski, Henryk, Austin, Jacob, Bieber, David, Dohan, David, Lewkowycz, Aitor, Bosma, Maarten, Luan, David, Sutton, Charles, and Odena, Augustus. Show your work: Scratchpads for intermediate computation with language models, 2022.

[128] Nye, Maxwell, Tessler, Michael Henry, Tenenbaum, Joshua B., and Lake, Brenden M. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In *Advances in Neural Information Processing Systems* (2021), A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, Eds.

[129] Olsson, Catherine, Elhage, Nelson, Nanda, Neel, Joseph, Nicholas, DasSarma, Nova, Henighan, Tom, Mann, Ben, Askell, Amanda, Bai, Yuntao, Chen, Anna, Conerly, Tom, Drain, Dawn, Ganguli, Deep, Hatfield-Dodds, Zac, Hernandez, Danny, Johnston, Scott, Jones, Andy, Kernion, Jackson, Lovitt, Liane, Ndousse, Kamal, Amodei, Dario, Brown, Tom, Clark, Jack, Kaplan, Jared, McCandlish, Sam, and Olah, Chris. In-context learning and induction heads. *Transformer Circuits Thread* (2022). https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[130] OpenAI, :, Achiam, Josh, Adler, Steven, Agarwal, Sandhini, Ahmad, Lama, Akkaya, Ilge, Aleman, Florencia Leoni, Almeida, Diogo, Altenschmidt, Janko, Altman, Sam, Anadkat, Shyamal, Avila, Red, and Igor Babuschkin, et al. Gpt-4 technical report, 2023.

[131] OpenAI. Chatgpt: A language model by openai.

[132] Ouyang, Long, Wu, Jeff, Jiang, Xu, Almeida, Diogo, Wainwright, Carroll L., Mishkin, Pamela, Zhang, Chong, Agarwal, Sandhini, Slama, Katarina, Ray, Alex, Schulman, John, Hilton, Jacob, Kelton, Fraser, Miller, Luke, Simens, Maddie, Askell, Amanda, Welinder, Peter, Christiano, Paul, Leike, Jan, and Lowe, Ryan. Training language models to follow instructions with human feedback, 2022.

[133] Pan, Alexander, Bhatia, Kush, and Steinhardt, Jacob. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations* (2022).

[134] Pang, Richard Yuanzhe, Parrish, Alicia, Joshi, Nitish, Nangia, Nikita, Phang, Jason, Chen, Angelica, Padmakumar, Vishakh, Ma, Johnny, Thompson, Jana, He, He, and Bowman, Samuel. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Seattle, United States, July 2022), Association for Computational Linguistics, pp. 5336–5358.

[135] Pang, Richard Yuanzhe, Parrish, Alicia, Joshi, Nitish, Nangia, Nikita, Phang, Jason, Chen, Angelica, Padmakumar, Vishakh, Ma, Johnny, Thompson, Jana, He, He, and Bowman, Samuel R. Quality: Question answering with long input texts, yes!, 2021.

[136] Passonneau, Rebecca J., and Litman, Diane J. Discourse segmentation by human and automated means. *Computational Linguistics 23*, 1 (1997), 103–139.

[137] Peng, Bowen, Quesnelle, Jeffrey, Fan, Honglu, and Shippole, Enrico. Yarn: Efficient context window extension of large language models, 2023.

[138] Press, Ofir, Smith, Noah, and Lewis, Mike. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations* (2022).

[139] Press, Ofir, Smith, Noah A., and Lewis, Mike. Shortformer: Better language modeling using shorter inputs, 2020.

[140] Press, Ofir, Smith, Noah A., and Lewis, Mike. Shortformer: Better language modeling using shorter inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Online, Aug. 2021), Association for Computational Linguistics, pp. 5493–5505.

[141] Press, Ofir, Zhang, Muru, Min, Sewon, Schmidt, Ludwig, Smith, Noah A., and Lewis, Mike. Measuring and narrowing the compositionality gap in language models, 2022.

[142] Radford, Alec, and Narasimhan, Karthik. Improving language understanding by generative pre-training.

[143] Radford, Alec, Wu, Jeffrey, Child, Rewon, Luan, David, Amodei, Dario, and Sutskever, Ilya. Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9.

[144] Rae, Jack, and Razavi, Ali. Do transformers need deep long-range memory? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 7524–7529.

[145] Rae, Jack W., Potapenko, Anna, Jayakumar, Siddhant M., Hillier, Chloe, and Lillicrap, Timothy P. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations* (2020).

[146] Rafailov, Rafael, Sharma, Archit, Mitchell, Eric, Ermon, Stefano, Manning, Christopher D., and Finn, Chelsea. Direct preference optimization: Your language model is secretly a reward model, 2023.

[147] Rosenfeld, Roni. A maximum entropy approach to adaptive statistical language modeling.

[148] Roy, Aurko, Saffar, Mohammad, Vaswani, Ashish, and Grangier, David. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics 9* (2021), 53–68.

[149] Roy, Aurko, Saffar, Mohammad, Vaswani, Ashish, and Grangier, David. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics 9* (2021), 53–68.

[150] Saha, Swarnadeep, Zhang, Shiyue, Hase, Peter, and Bansal, Mohit. Summarization programs: Interpretable abstractive summarization with neural modular trees. In *The Eleventh International Conference on Learning Representations* (2023).

[151] Saunders, William, Yeh, Catherine, Wu, Jeff, Bills, Steven, Ouyang, Long, Ward, Jonathan, and Leike, Jan. Self-critiquing models for assisting human evaluators, 2022.

[152] Schank, Roger C, and Abelson, Robert P. Scripts, plans, goals, and understanding: an inquiry into human knowledge structures, 1977.

[153] Schick, Timo, Dwivedi-Yu, Jane, Dessì, Roberto, Raileanu, Roberta, Lomeli, Maria, Zettlemoyer, Luke, Cancedda, Nicola, and Scialom, Thomas. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[154] Schulman, John. Approximating kl divergence. `http://joschu.net/blog/kl-approx.html`, 2020.

[155] Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael I., and Abbeel, Pieter. Trust region policy optimization, 2017.

[156] Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael, and Abbeel, Pieter. High-dimensional continuous control using generalized advantage estimation, 2018.

[157] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms, 2017.

[158] Shaham, Uri, Ivgi, Maor, Efrat, Avia, Berant, Jonathan, and Levy, Omer. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (Singapore, Dec. 2023), Houda Bouamor, Juan Pino, and Kalika Bali, Eds., Association for Computational Linguistics, pp. 7977–7989.

[159] Shaham, Uri, Segal, Elad, Ivgi, Maor, Efrat, Avia, Yoran, Ori, Haviv, Adi, Gupta, Ankit, Xiong, Wenhan, Geva, Mor, Berant, Jonathan, and Levy, Omer. Scrolls: Standardized comparison over long language sequences, 2022.

[160] Shaham, Uri, Segal, Elad, Ivgi, Maor, Efrat, Avia, Yoran, Ori, Haviv, Adi, Gupta, Ankit, Xiong, Wenhan, Geva, Mor, Berant, Jonathan, and Levy, Omer. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (Abu Dhabi, United Arab Emirates, Dec. 2022), Association for Computational Linguistics, pp. 12007–12021.

[161] Sharan, Vatsal, Kakade, Sham, Liang, Percy, and Valiant, Gregory. Prediction with a short memory. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (New York, NY, USA, 2018), STOC 2018, Association for Computing Machinery, p. 1074–1087.

[162] Shen, Sheng, Dong, Zhen, Ye, Jiayu, Ma, Linjian, Yao, Zhewei, Gholami, Amir, Mahoney, Michael W., and Keutzer, Kurt. Q-bert: Hessian based ultra low precision quantization of bert. *Proceedings of the AAAI Conference on Artificial Intelligence 34*, 05 (Apr. 2020), 8815–8821.

[163] Shen, Zhiqiang, Tao, Tianhua, Ma, Liqun, Neiswanger, Willie, Liu, Zhengzhong, Wang, Hongyi, Tan, Bowen, Hestness, Joel, Vassilieva, Natalia, Soboleva, Daria, and Xing, Eric. Slimpajama-dc: Understanding data combinations for llm training, 2023.

[164] Shi, Weijia, Min, Sewon, Yasunaga, Michihiro, Seo, Minjoon, James, Rich, Lewis, Mike, Zettlemoyer, Luke, and tau Yih, Wen. Replug: Retrieval-augmented black-box language models, 2023.

[165] Shinn, Noah, Labash, Beck, and Gopinath, Ashwin. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366* (2023).

[166] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 56 (2014), 1929–1958.

[167] Stelmakh, Ivan, Luan, Yi, Dhingra, Bhuwan, and Chang, Ming-Wei. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (Abu Dhabi, United Arab Emirates, Dec. 2022), Association for Computational Linguistics, pp. 8273–8288.

[168] Stevick, Philip. *The Chapter in Fiction: Theories of Narrative Division*. Syracuse, NY: Syracuse University Press, c1970., 1970.

[169] Stiennon, Nisan, Ouyang, Long, Wu, Jeff, Ziegler, Daniel M., Lowe, Ryan, Voss, Chelsea, Radford, Alec, Amodei, Dario, and Christiano, Paul. Learning to summarize from human feedback, 2022.

[170] Stock, Pierre, Fan, Angela, Graham, Benjamin, Grave, Edouard, Gribonval, Rémi, Jegou, Herve, and Joulin, Armand. Training with quantization noise for extreme model compression. In *International Conference on Learning Representations* (2021).

[171] Su, Dan, Li, Xiaoguang, Zhang, Jindi, Shang, Lifeng, Jiang, Xin, Liu, Qun, and Fung, Pascale. Read before generate! faithful long form question answering with machine reading. In *Findings of the Association for Computational Linguistics: ACL 2022* (Dublin, Ireland, May 2022), Association for Computational Linguistics, pp. 744–756.

[172] Su, Jianlin, Lu, Yu, Pan, Shengfeng, Murtadha, Ahmed, Wen, Bo, and Liu, Yunfeng. Roformer: Enhanced transformer with rotary position embedding, 2023.

[173] Sukhbaatar, Sainbayar, Grave, Edouard, Bojanowski, Piotr, and Joulin, Armand. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 331–335.

[174] Sun, Haitian, Cohen, William, and Salakhutdinov, Ruslan. ConditionalQA: A complex reading comprehension dataset with conditional answers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Dublin, Ireland, May 2022), Association for Computational Linguistics, pp. 3627–3637.

[175] Sun, Haitian, Cohen, William W., and Salakhutdinov, Ruslan. Iterative hierarchical attention for answering complex questions over long documents, 2021.

[176] Sun, Simeng, Krishna, Kalpesh, Mattarella-Micke, Andrew, and Iyyer, Mohit. Do long-range language models actually use long-range context? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (Online and Punta Cana, Dominican Republic, Nov. 2021), Association for Computational Linguistics, pp. 807–822.

[177] Talmor, Alon, Herzig, Jonathan, Lourie, Nicholas, and Berant, Jonathan. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4149–4158.

[178] Taori, Rohan, Gulrajani, Ishaan, Zhang, Tianyi, Dubois, Yann, Li, Xuechen, Guestrin, Carlos, Liang, Percy, and Hashimoto, Tatsunori B. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[179] Tay, Yi, Bahri, Dara, Metzler, Donald, Juan, Da-Cheng, Zhao, Zhe, and Zheng, Che. Synthesizer: Rethinking self-attention for transformer models, 2021.

[180] Tay, Yi, Dehghani, Mostafa, Abnar, Samira, Shen, Yikang, Bahri, Dara, Pham, Philip, Rao, Jinfeng, Yang, Liu, Ruder, Sebastian, and Metzler, Donald. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations* (2021).

[181] Tay, Yi, Dehghani, Mostafa, Bahri, Dara, and Metzler, Donald. Efficient transformers: A survey, 2020.

[182] Touvron, Hugo, Lavril, Thibaut, Izacard, Gautier, Martinet, Xavier, Lachaux, Marie-Anne, Lacroix, Timothée, Rozière, Baptiste, Goyal, Naman, Hambro, Eric, Azhar, Faisal, Rodriguez, Aurelien, Joulin, Armand, Grave, Edouard, and Lample, Guillaume. Llama: Open and efficient foundation language models, 2023.

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

[184] Tworkowski, Szymon, Staniszewski, Konrad, Pacek, Mikołaj, Wu, Yuhuai, Michalewski, Henryk, and Miłoś, Piotr. Focused transformer: Contrastive training for context scaling, 2023.

[185] Vaithilingam, Priyan, Zhang, Tianyi, and Glassman, Elena L. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2022), CHI EA '22, Association for Computing Machinery.

[186] van den Oord, Aäron, Li, Yazhe, and Vinyals, Oriol. Representation learning with contrastive predictive coding. *ArXiv abs/1807.03748* (2018).

[187] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Ł ukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[188] Vu, Tu, Barua, Aditya, Lester, Brian, Cer, Daniel, Iyyer, Mohit, and Constant, Noah. Overcoming catastrophic forgetting in zero-shot cross-lingual generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (Abu Dhabi, United Arab Emirates, Dec. 2022), Association for Computational Linguistics, pp. 9279–9300.

[189] Walker, Marilyn, and Whittaker, Steve. Mixed initiative in dialogue: An investigation into discourse segmentation. In *28th Annual Meeting of the Association for Computational Linguistics* (Pittsburgh, Pennsylvania, USA, June 1990), Association for Computational Linguistics, pp. 70–78.

[190] Wang, Alex, Cho, Kyunghyun, and Lewis, Mike. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 5008–5020.

[191] Wang, Chaojun, and Sennrich, Rico. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 3544–3552.

[192] Wang, Lei, Xu, Wanyu, Lan, Yihuai, Hu, Zhiqiang, Lan, Yunshi, Lee, Roy Ka-Wei, and Lim, Ee-Peng. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023.

[193] Wang, Sinong, Li, Belinda Z., Khabsa, Madian, Fang, Han, and Ma, Hao. Linformer: Self-attention with linear complexity, 2020.

[194] Wang, Tian, and Cho, Kyunghyun. Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 1319–1329.

[195] Wang, Xuezhi, Wei, Jason, Schuurmans, Dale, Le, Quoc V, Chi, Ed H., Narang, Sharan, Chowdhery, Aakanksha, and Zhou, Denny. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations* (2023).

[196] Wang, Yizhong, Ivison, Hamish, Dasigi, Pradeep, Hessel, Jack, Khot, Tushar, Chandu, Khyathi Raghavi, Wadden, David, MacMillan, Kelsey, Smith, Noah A., Beltagy, Iz, and Hajishirzi, Hannaneh. How far can camels go? exploring the state of instruction tuning on open resources, 2023.

[197] Wang, Yizhong, Kordi, Yeganeh, Mishra, Swaroop, Liu, Alisa, Smith, Noah A., Khashabi, Daniel, and Hajishirzi, Hannaneh. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Toronto, Canada, July 2023), Association for Computational Linguistics, pp. 13484–13508.

[198] Wang, Yizhong, Li, Sujian, and Yang, Jingfeng. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 962–967.

[199] WEBBER, B., EGG, M., and KORDONI, V. Discourse structure and language technology. *Natural Language Engineering 18*, 4 (2012), 437–490.

[200] Wei, Jason, Wang, Xuezhi, Schuurmans, Dale, Bosma, Maarten, brian ichter, Xia, Fei, Chi, Ed H., Le, Quoc V, and Zhou, Denny. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems* (2022), Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, Eds.

[201] Wei, Xiang, Cui, Xingyu, Cheng, Ning, Wang, Xiaobin, Zhang, Xin, Huang, Shen, Xie, Pengjun, Xu, Jinan, Chen, Yufeng, Zhang, Meishan, Jiang, Yong, and Han, Wenjuan. Zero-shot information extraction via chatting with chatgpt, 2023.

[202] Welleck, Sean, Kulikov, Ilia, Roller, Stephen, Dinan, Emily, Cho, Kyunghyun, and Weston, Jason. Neural text generation with unlikelihood training. In *International Conference on Learning Representations* (2020).

[203] Weng, Lilian. Policy gradient. `https://lilianweng.github.io/posts/2018-04-08-policy-gradient/`, 2018.

[204] Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks, 2015.

[205] Wu, Yuhuai, Rabe, Markus N., Hutchins, DeLesley S., and Szegedy, Christian. Memorizing transformers. *ArXiv abs/2203.08913* (2022).

[206] Wu, Zeqiu, Hu, Yushi, Shi, Weijia, Dziri, Nouha, Suhr, Alane, Ammanabrolu, Prithviraj, Smith, Noah A., Ostendorf, Mari, and Hajishirzi, Hannaneh. Fine-grained human feedback gives better rewards for language model training, 2023.

[207] Wu, Zhanghao, Liu, Zhijian, Lin, Ji, Lin, Yujun, and Han, Song. Lite transformer with long-short range attention, 2020.

[208] Xiao, Guangxuan, Tian, Yuandong, Chen, Beidi, Han, Song, and Lewis, Mike. Efficient streaming language models with attention sinks, 2023.

[209] Xiong, Wenhan, Liu, Jingyu, Molybog, Igor, Zhang, Hejia, Bhargava, Prajjwal, Hou, Rui, Martin, Louis, Rungta, Rashi, Sankararaman, Karthik Abinav, Oguz, Barlas, Khabsa, Madian, Fang, Han, Mehdad, Yashar, Narang, Sharan, Malik, Kshitiz, Fan, Angela, Bhosale, Shruti, Edunov, Sergey, Lewis, Mike, Wang, Sinong, and Ma, Hao. Effective long-context scaling of foundation models, 2023.

[210] Xu, Fangyuan, Li, Junyi Jessy, and Choi, Eunsol. How do we answer complex questions: Discourse structure of long-form answers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Dublin, Ireland, May 2022), Association for Computational Linguistics, pp. 3556–3572.

[211] Xu, Peng, Ping, Wei, Wu, Xianchao, McAfee, Lawrence, Zhu, Chen, Liu, Zihan, Subramanian, Sandeep, Bakhturina, Evelina, Shoeybi, Mohammad, and Catanzaro, Bryan. Retrieval meets long context large language models, 2023.

[212] Yao, Shunyu, Yu, Dian, Zhao, Jeffrey, Shafran, Izhak, Griffiths, Thomas L., Cao, Yuan, and Narasimhan, Karthik. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[213] Yao, Shunyu, Zhao, Jeffrey, Yu, Dian, Du, Nan, Shafran, Izhak, Narasimhan, Karthik R, and Cao, Yuan. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations* (2023).

[214] Ye, Hongbin, Liu, Tong, Zhang, Aijia, Hua, Wei, and Jia, Weiqiang. Cognitive mirage: A review of hallucinations in large language models, 2023.

[215] Ye, Junjie, Chen, Xuanting, Xu, Nuo, Zu, Can, Shao, Zekai, Liu, Shichun, Cui, Yuhan, Zhou, Zeyang, Gong, Chao, Shen, Yang, Zhou, Jie, Chen, Siming, Gui, Tao, Zhang, Qi, and Huang, Xuanjing. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models, 2023.

[216] Yogatama, Dani, de Masson d'Autume, Cyprien, and Kong, Lingpeng. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics 9* (2021), 362–373.

[217] Yoran, Ori, Wolfson, Tomer, Bogin, Ben, Katz, Uri, Deutch, Daniel, and Berant, Jonathan. Answering questions by meta-reasoning over multiple chains of thought. *arXiv preprint arXiv:2304.13007* (2023).

[218] Zaheer, Manzil, Guruganesh, Guru, Dubey, Kumar Avinava, Ainslie, Joshua, Alberti, Chris, Ontanon, Santiago, Pham, Philip, Ravula, Anirudh, Wang, Qifan, Yang, Li, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems 33* (2020).

[219] Zellers, Rowan, Bisk, Yonatan, Schwartz, Roy, and Choi, Yejin. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 93–104.

[220] Zhang, Jiacheng, Luan, Huanbo, Sun, Maosong, Zhai, Feifei, Xu, Jingfang, Zhang, Min, and Liu, Yang. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 533–542.

[221] Zhang, Pei, Chen, Boxing, Ge, Niyu, and Fan, Kai. Long-short term masking transformer: A simple but effective baseline for document-level neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Online, Nov. 2020), Association for Computational Linguistics, pp. 1081–1087.

[222] Zhang, Shengyu, Dong, Linfeng, Li, Xiaoya, Zhang, Sen, Sun, Xiaofei, Wang, Shuhe, Li, Jiwei, Hu, Runyi, Zhang, Tianwei, Wu, Fei, and Wang, Guoyin. Instruction tuning for large language models: A survey, 2023.

[223] Zhang, Shun, Chen, Zhenfang, Shen, Yikang, Ding, Mingyu, Tenenbaum, Joshua B., and Gan, Chuang. Planning with large language models for code generation, 2023.

[224] Zhang, Susan, Roller, Stephen, Goyal, Naman, Artetxe, Mikel, Chen, Moya, Chen, Shuohui, Dewan, Christopher, Diab, Mona, Li, Xian, Lin, Xi Victoria, Mihaylov, Todor, Ott, Myle, Shleifer, Sam, Shuster, Kurt, Simig, Daniel, Koura, Punit Singh, Sridhar, Anjali, Wang, Tianlu, and Zettlemoyer, Luke. Opt: Open pre-trained transformer language models, 2022.

[225] Zheng, Lianmin, Chiang, Wei-Lin, Sheng, Ying, Zhuang, Siyuan, Wu, Zhanghao, Zhuang, Yonghao, Lin, Zi, Li, Zhuohan, Li, Dacheng, Xing, Eric, Zhang, Hao, Gonzalez, Joseph E., and Stoica, Ion. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track* (2023).

[226] Zhou, Chunting, Liu, Pengfei, Xu, Puxin, Iyer, Srini, Sun, Jiao, Mao, Yuning, Ma, Xuezhe, Efrat, Avia, Yu, Ping, Yu, Lili, Zhang, Susan, Ghosh, Gargi, Lewis, Mike, Zettlemoyer, Luke, and Levy, Omer. Lima: Less is more for alignment, 2023.

[227] Zhou, Denny, Schärli, Nathanael, Hou, Le, Wei, Jason, Scales, Nathan, Wang, Xuezhi, Schuurmans, Dale, Cui, Claire, Bousquet, Olivier, Le, Quoc V, and Chi, Ed H. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations* (2023).

[228] Zhou, Hattie, Bradley, Arwen, Littwin, Etai, Razin, Noam, Saremi, Omid, Susskind, Josh, Bengio, Samy, and Nakkiran, Preetum. What algorithms can transformers learn? a study in length generalization, 2023.

[229] Ziegler, Daniel M., Stiennon, Nisan, Wu, Jeffrey, Brown, Tom B., Radford, Alec, Amodei, Dario, Christiano, Paul, and Irving, Geoffrey. Fine-tuning language models from human preferences, 2020.

APPENDIX A

APPENDIX FOR CHAPTER 2

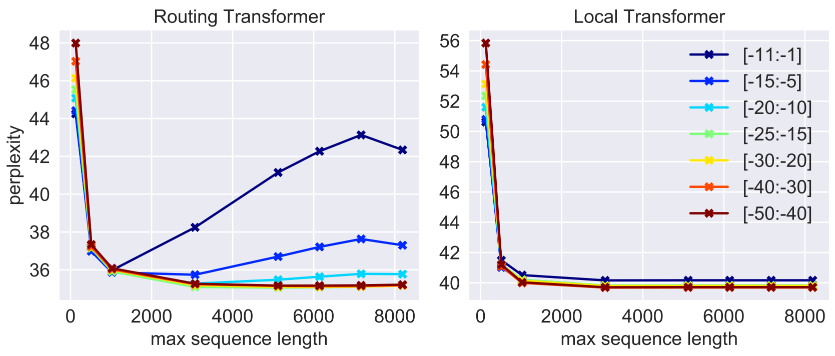


Figure A.1: Perplexity of target chunk of 10 tokens long near the end of sequence. The legend indicates how far away the evaluated chunk is from the end of sequence (e.g., [-15:-5] means evaluating the last 15 tokens to the last 5 tokens). The left plot shows the clustering mechanism in the Routing Transformer assigns higher perplexity to around the last 15 tokens in a sequence. Due to this artifacts, in our analysis, we avoid tokens within this range to make comparable comparison with the Local Transformer. In the main text, our analysis are conducted over the last 50 to last 40 tokens, which are not affected by this artifact.

A.1 Routing Transformer and end-sequence degradation

In our analysis, instead of picking the last 10 tokens in a sequence, we chose the last 50 to last 40 tokens due to an artifact introduced by the clustering heads in the RT model. We find that in general the last 20 tokens in a sequence tend to have increasing perplexity as we evaluate on longer and longer sequence lengths. As shown in Figure A.1, this phenomenon is only native to the RT model and disappears when the clustering heads are replaced with local attentions. Therefore, to make the RT and the LT comparable, we select the tokens from the range that is not affected by the end-sequence issue. Although it's not the last 10 tokens, this short target chunk is still located near the end of a sequence, preceded by enough long context.

A.2 Effect of longer context

In section 2.3 we discussed that books that are fictional and continuous benefit more from the long-range context. We also annotated the validation set by the authorship (i.e., whether a book is written by single author or various authors). Out of 49 books, 11 are written by various authors, 10 of which are non-fictions. Due to this overlap, we only show results of fic/non-fic in the main text.

In this section , we also further break down all targets to the three types of tokens we examined in section 2.3, and display the results by book types. Perplexity of infrequent tokens (Figure A.2), tokens inside subword clusters (Figure A.3), and tokens whose last occurrence is more than 2K tokens away (Figure A.4). In general, for the small set of tokens whose perplexity keep decreasing as adding in more context, the major source of improvements are from the continuous and fictional books.

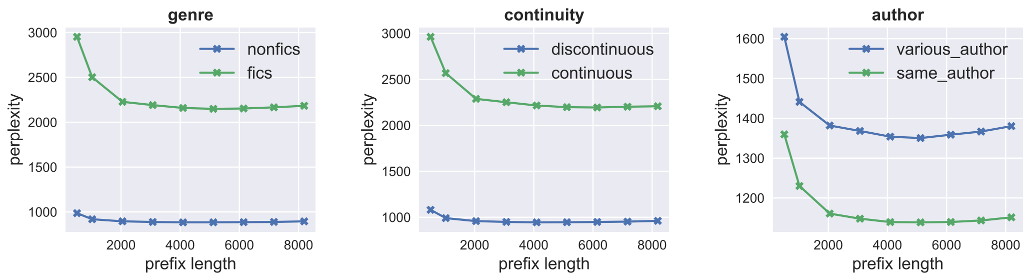


Figure A.2: Perplexity of infrequent target tokens, broken down by genre (**left**), continuity (**middle**), and authorship (**right**). Perplexity of infrequent tokens in fictional, continuous and single-authored books decreases as the context length increases to around 5K. On the other hand, the rest types of books rely on more local context while predicting the infrequent tokens.

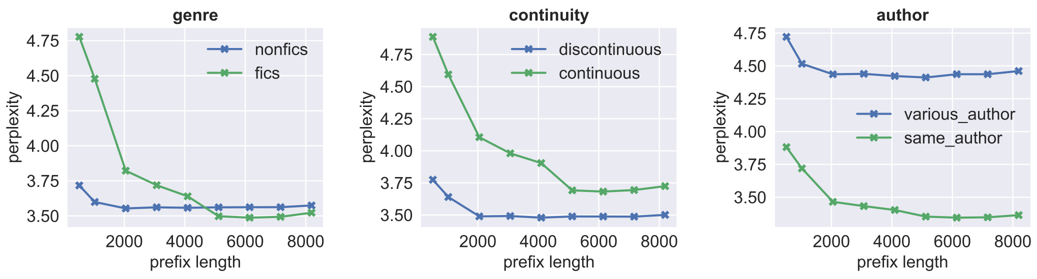


Figure A.3: Perplexity of target tokens inside subword clusters (i.e., excluding the first subword in each cluster), broken down by genre (**left**), continuity (**middle**), and authorship (**right**). Perplexity of these tokens in fictional and continuous books improves as the length increases up to around 4K, whereas nonfictional and discontinuous books are not taking any advantage of long-range context at all.

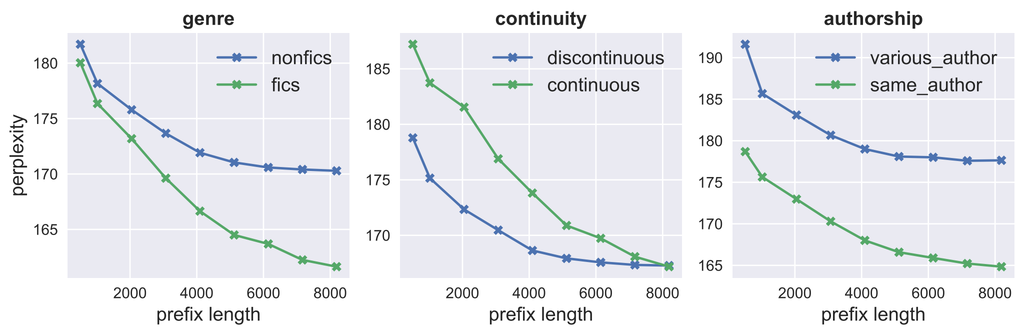


Figure A.4: Perplexity of target tokens that can only be found in the distant context, when evaluated with the Routing Transformer on subset of PG-19 validation set, broken down by genre (**left**), continuity (**middle**), and authorship (**right**). Fictional, continuous, and single-authored books continue to have improved perplexity for this type of tokens as the prefix length increases up to 8K. The single-authored books also contain discontinuous books which require less modeling of long-range context. The decreasing curve indicates the model might have acquired author specific token statistics from incorporating longer context.

	infreq	in-subword	distant
infreq	1.	0.09	0.08
in-subword	0.36	1.	0.1
distant	0.07	0.02	1.

Table A.1: Ratio of overlapped target tokens among different categorizations. **infreq** are infrequent tokens, **in-subword** are tokens within a subword cluster (i.e., excluding the first word in a subword cluster), **distant** are tokens only appear in the distant context (more than 2K away). Row 1 column 3 the number 0.08 indicates around 8% of the infrequent tokens are those that can only be found in the distant context.

A.3 Token overlaps

We have shown in section 2.3 that infrequent tokens, tokens inside subword clusters, and tokens that can only be copied from distant context benefit from context longer than 2K tokens. It is possible that these improvements come from the same set of tokens shared across these three types of tokens. To verify if there are significant overlaps among those three types of tokens, we compute the overlapped ratio in Table A.1. Except for in-subword and infrequent tokens, the overlapped ratios are all below 0.1.

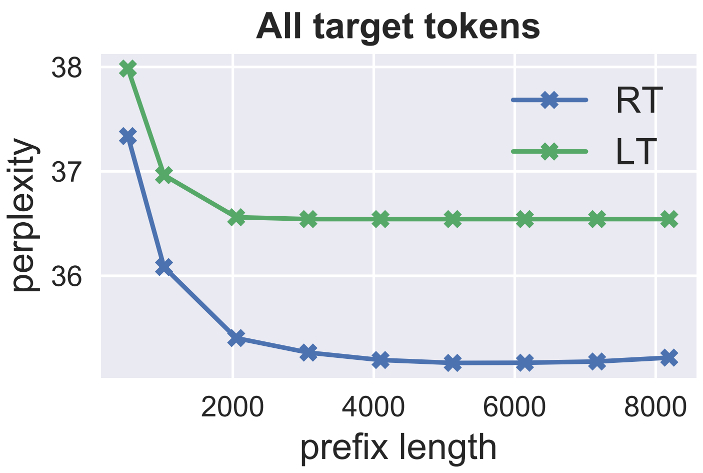


Figure A.5: The perplexity of all target tokens plateaus after 2K prefix tokens for both Routing Transformer and Local Transformer. The LT model is the one released in 2021 summer.

A.4 Perturbation

Perplexity with perturbed distant prefix when evaluated with Local Transformer is shown in Figure A.7. Perplexity hardly changes when perturbing up to around 6K tokens. Because LT doesn't properly use long-range context, we only present the results of Routing Transformer in the main text.

A.5 Suffix Identification

In the main text, we present the suffix identification results with 128-token long suffix. Here, we provide results when evaluate the accuracy with suffix of different length. Interestingly, the accuracy of distinguishing a gold suffix first increases with the suffix length, reaching the best when the suffix length is around 10 to 20, then decreases as the suffix becomes longer. This is likely because the sequence becomes more probable as incorporating more local context (part of the suffix).

In Table A.2 and Table A.3, we present a complete example of the suffix identification task. The prefix contains 1024 tokens, and each of the suffixes contains 128 tokens. Lower perplexity of obviously wrong suffix (e.g. negative 1,3,4) indicates current RT model is not properly taking advantage of distant context to make sequence-level predictions.

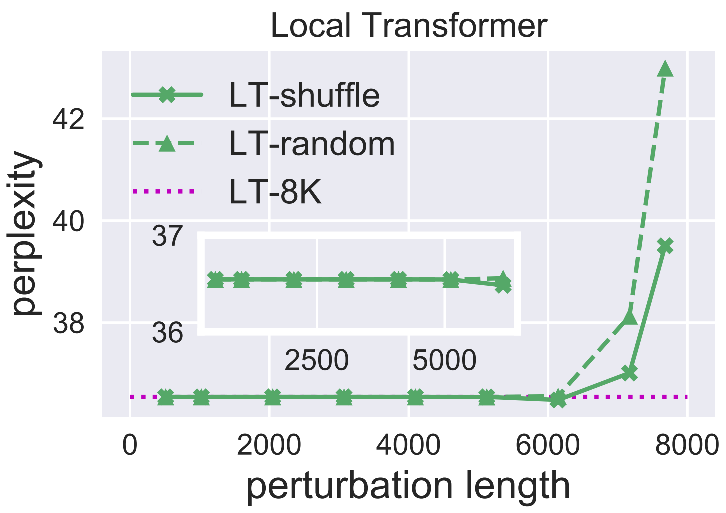


Figure A.6: Perplexity of all target tokens when evaluated with Local Transformer released in 2021 summer.

A.6 Local Transformer checkpoint results

In the main text, we analyzed both the RT checkpoint and an LT model derived from the same RT checkpoint by replacing the clustering heads with local attention heads. After the submission deadline and before the camera ready deadline, the author of the Routing Transformer released a new LT checkpoint, which has 24 layers in total[1]. To make sure the behavior of our former LT is the same with an LT trained from scratch, we also conducted all our analysis again on this new checkpoint. Overall, we find the new LT checkpoint performs slightly better on token-level tasks but is inferior to the previous LT checkpoint on suffix identification, a sequence-level task. Since the trends are the same, no conclusion needs to be changed.

The Effect of Longer Context In Figure A.5 are the perplexities in aggregate over all target tokens of both the RT and the new LT checkpoints. The target tokens are the same ones we used in the main text. The new LT has better perplexity than the one presented in the main text (36.5 vs. 40 as the prefix length extends to 8K).

Perturbation of long-range context Similar to the former LT (Figure A.7), the newly released LT checkpoint(Figure A.6) is not sensitive at all to the perturbation further than local 2K tokens. Both models are impacted by local random replacement more than shuffling, however, the new LT checkpoint has overall better perplexity than the RT-derived LT.

[1]Both the RT model and the *former* LT model have 22 layers.

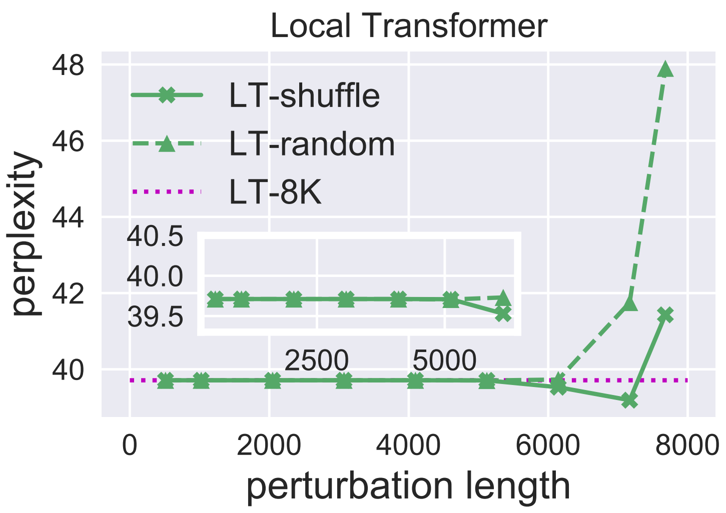


Figure A.7: Perplexity of all target tokens when evaluated with Local Transformer derived from the RT checkpoint.

Sequence-level tasks Figure A.9 shows the performance of both RT and the released LT checkpoint on sequence-level tasks. Compared to the results in the main text, the new LT checkpoint performs better at sequence-copying task, however, the trend remains the same – the order of tokens beyond 2K tokens is not properly encoded. On the other hand, the new LT checkpoint is slightly worse in suffix identification while the former RT-derived LT has almost identical performance as the RT. This implies even though the clustering heads are removed from the previous LT, useful information is preserved by the local attention heads.

Overall, we find the released LT checkpoint has better token-level performance but performs worse on suffix identification. Each plot shares the same trend with its corresponding one in the main text, thus no conclusion needs to be modified.

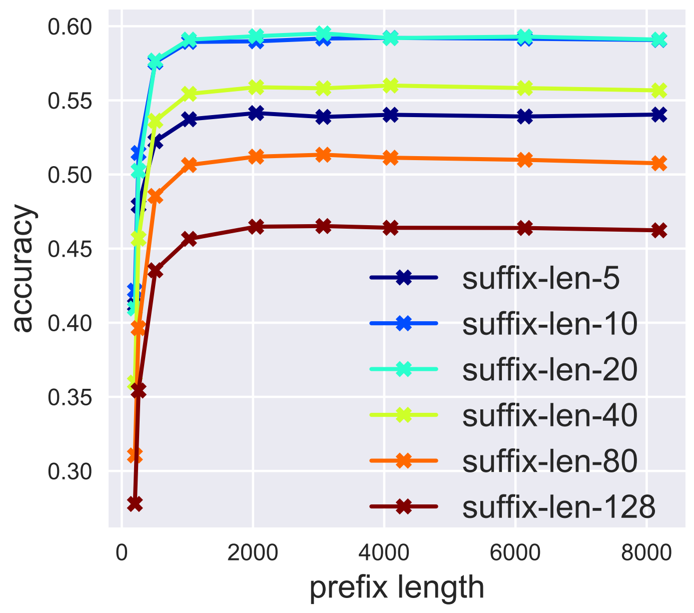


Figure A.8: Suffix identification accuracy when evaluated with different prefix length.

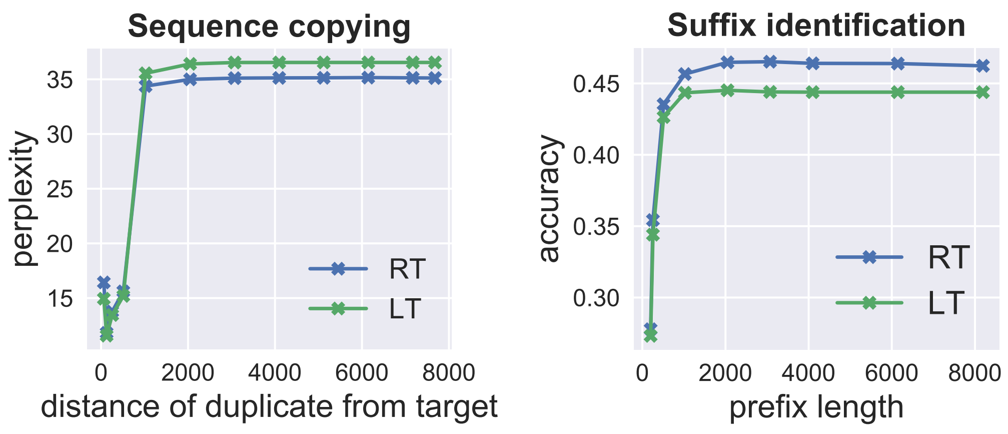


Figure A.9: **Left:** Both models assign low perplexity if a duplicate sequence appears within previous 512 tokens. **Right:** Adding context beyond 2K tokens does not improve performance of suffix identification. Moreover, the recently released LT performs worse than the one derived from the RT checkpoint.

Prefix	Not a bit, sir.Out with it!I have faced death too often to flinch from it now, though I saw it as near me as you are.""Well, well, we must go by averages of course.Shall we say two years?I should think that you have a full two years before you.""In two years your pension would bring you in L1,600.Now I will do my very best for you, Admiral!I will advance you L2,000, and you can make over to me your pension for your life.It is pure speculation on my part.If you die to-morrow I lose my money.If the doctor's prophecy is correct I shall still be out of pocket.If you live a little longer, then I may see my money again.It is the very best I can do for you.""Then you wish to buy my pension?""Yes, for two thousand down.""And if I live for twenty years?""Oh, in that case of course my speculation would be more successful.But you have heard the doctor's opinion.""Would you advance the money instantly?""You should have a thousand at once.The other thousand I should expect you to take in furniture.""In furniture?""Yes, Admiral.We shall do you a beautiful houseful at that sum.It is the custom of my clients to take half in furniture."The Admiral sat in dire perplexity.He had come out to get money, and to go back without any, to be powerless to help when his boy needed every shilling to save him from disaster, that would be very bitter to him.On the other hand, it was so much that he surrendered, and so little that he received.Little, and yet something.Would it not be better than going back empty-handed?He saw the yellow backed chequebook upon the table.The moneylender opened it and dipped his pen into the ink."Shall I fill it up?"said he."I think, Admiral," remarked Westmacott, "that we had better have a little walk and some luncheon before we settle this matter.""Oh, we may as well do it at once.It would be absurd to postpone it now," Metaxa spoke with some heat, and his eyes glinted angrily from between his narrow lids at the imperturbable Charles.The Admiral was simple in money matters, but he had seen much of men and had learned to read them.He saw that venomous glance, and saw too that intense eagerness was peeping out from beneath the careless air which the agent had assumed."You're quite right, Westmacott," said he."We'll have a little walk before we settle it.""But I may not be here this afternoon.""Then we must choose another day.""But why not settle it now?""Because I prefer not," said the Admiral shortly."Very well.But remember that my offer is only for to-day.It is off unless you take it at once.""Let it be off, then.""There's my fee," cried the doctor."How much?""A guinea."The Admiral threw a pound and a shilling upon the table."Come, Westmacott," said he, and they walked together from the room."I don't like it," said Charles, when they found themselves in the street once more; "I don't profess to be a very sharp chap, but this is a trifle too thin.What did he want to go out and speak to the doctor for?And how very convenient this tale of a weak heart was!I believe they are a couple of rogues, and in league with each other.""A shark and a pilot fish," said the Admiral."I'll tell you what I propose, sir.There's a lawyer named McAdam who does my aunt's business.He is a very honest fellow, and lives at the other side of Poultry.We'll go over to him together and have his opinion about the whole matter.""How far is it to his place?""Oh, a mile at least.We can have a cab.""A mile?
Gold suffix (ppl=74.4)	Then we shall see if there is any truth in what that swab of a doctor said.Come, my boy, and clap on all sail, and see who can stay the longest."Then the sober denizens of the heart of business London saw a singular sight as they returned from their luncheons.Down the roadway, dodging among cabs and carts, ran a weather-stained elderly man, with wide flapping black hat, and homely suit of tweeds.With elbows braced back, hands clenched near his armpits, and chest protruded, he scudded along, while close at his heels lumbered a large-limbed, heavy, yellow mustached young man, who seemed to feel the exercise a good deal more than his senior.On they dashed, helter-skelter, until they pulled up panting at the office where the lawyer of the

Table A.2: Suffix identification example, extracted from the book *Beyond the City* by Arthur Conan Doyle. For each suffix, we show the perplexity of the suffix evaluated with the Routing Transformer. RT assigns a lot lower perplexity to examples (e.g. negative 1,3,4) that are obviously wrong given the prefix.

Negative 1 (ppl=34.6):	If I can see Mr.Elberry to-day we may let you have a cheque to-morrow.Try another pinch.No?Well, good-bye.I am very happy to have been of service."Mr.McAdam bowed them out, for he was a very busy man, and they found themselves in the street once more with lighter hearts than when they had left it."Well, Westmacott, I am sure I am very much obliged to you," said the Admiral."You have stood by me when I was the better for a little help, for I'm clean out of my soundings among these city sharks.But I've something to do now which is more in my own line, and I need not trouble you any more.""Oh, it is no trouble.I have nothing
Negative 2 (ppl=71.69):	It was not until he had signed and sent it off that the full significance of all that he had done broke upon him.He had sacrificed everything.His pension was gone.He had nothing save only what he could earn.But the stout old heart never quailed.He waited eagerly for a letter from the Saint Lawrence Shipping Company, and in the meanwhile he gave his landlord a quarter's notice.Hundred pound a year houses would in future be a luxury which he could not aspire to.A small lodging in some inexpensive part of London must be the substitute for his breezy Norwood villa.So be it, then!Better that a thousand fold than that his name should be associated with failure and disgrace.On that morning Harold Denver was to meet the creditors of the firm, and to explain the situation to them.It was a hateful task
Negative 3 (ppl=33.54):	He hurried in, fearing that she might have taken some turn for the worse, but he was reassured to find her sitting up in her bed, with Clara and Ida Walker in attendance upon her.She had removed the handkerchief, and had put on a little cap with pink ribbons, and a maroon dressing-jacket, daintily fulled at the neck and sleeves."My dear friend," said she as he entered, "I wish to make a last few remarks to you.No, no," she continued, laughing, as she saw a look of dismay upon his face."I shall not dream of dying for at least another thirty years.A woman should be ashamed to die before she is seventy.I wish, Clara, that you would ask your father to step up.And you, Ida, just pass me my cigarettes, and
Negative 4 (ppl=34.92):	look!!!"Her voice had fallen suddenly to a quivering whisper and she was pointing to the Westmacotts' house.Her sister gave a gasp of horror, and stood with a clutch at Monica's arm, staring in the same direction.There was a light in the front room, a slight, wavering light such as would be given by a small candle or taper.The blind was down, but the light shone dimly through.Outside in the garden, with his figure outlined against the luminous square, there stood a man, his back to the road, his two hands upon the window ledge, and his body rather bent as though he were trying to peep in past the blind.So absolutely still and motionless was he that in spite of the moon they might well have overlooked him were it not for that tell-tale light behind.
Negative 5 (ppl=47.66):	Again the Admiral burst out cheering."There remains, therefore, about L3,200 which has to be found within ten days.No man shall lose by me.I gave them my word in the room that if I worked my soul out of my body every one of them should be paid.I shall not spend a penny upon myself until it is done.But some of them can't wait.They are poor men themselves, and must have their money.They have issued a warrant for Pearson's arrest.But they think that he has got away to the States.""These men shall have their money," said the Admiral."Dad!""Yes, my boy, you don't know the resources of the family.One never does know until one tries.What have you yourself now?""I have about a

Table A.3: Suffix identification example, extracted from the book *Beyond the City* by Arthur Conan Doyle. For each suffix, we show the perplexity of the suffix evaluated with the Routing Transformer. RT assigns a lot lower perplexity to examples (e.g. negative 1,3,4) that are obviously wrong given the prefix.

APPENDIX B

APPENDIX FOR CHAPTER 3

B.1 Dataset statistics

We collected 13,682 fanfictions from an online dump of stories posted on Archive of Our Own (AO3) by filtering works written in English language, rated General Audience by the author and contains at least 10K words and more than 10 chapters. For each chapter, we remove the text within the range of "`**Notes for the Chapter:**`", "`**Summary for the Chapter:**`" and "`**Author's Note:**`". The meta-comments inserted into the main text by the authors are not removed. The statistics of this long-fic dataset are included in Table B.1. We do not apply other profanity filters to the fictions, therefore there may still be inappropriate content for general audience as the rating is self-labeled by each author. Besides chapter breaks introduced in the main text, we also collected two other discourse boundaries, cause and dialogue, as comparisons to the chapter boundary examples. We present the statistics each type of examples in Table B.2.

- **Cause**: The beginning of the suffix contains words or phrases 'because', 'due to', 'owing to'. According to [3], human readers reactivate memory of global context for comprehending statements following causes or goals.
- **Dialogue**: The gold suffix in this category starts with a quotation mark. This often happens in dialogues where the continuation of one interlocutor depends heavily on the immediately preceding utterance. We conjecture this is the type where the prediction relies more on the local rather than the global context.

	mean	min	max
#chapters	21.5	11	589
#words	41,513.2	10,000	636,468

Table B.1: Statistics of long fanfictions collected from AO3 story dump.

	AO3		PG19	
Suffix Type	#works	#examples	#works	#examples
cause	965	8,133	45	506
dialogue	979	8,724	46	3,165
chapter breaks	1202	7,355	17	241

Table B.2: Data statistics of CHAPTERBREAK as well as another two discourse boundary examples.

B.2 Baselines

Bigbird [218] To reduce the quadratic complexity of self-attention in the standard Transformer, the Bigbird model employs a mixture of global, random and local attention mechanisms, which successfully reduce the complexity to linear. The idea is to insert each sequence $O(1)$ global tokens, which attend to all other tokens. The rest tokens attend to their neighbor tokens, random tokens in the sequence as well as the inserted global tokens. A very similar idea is developed concurrently in the Longformer [13]. The Bigbird model we fine-tuned is the decoder part of the released checkpoint. We fine-tune the model with causal LM objective on 14K books of PG-19 with peak learning rate 0.0001 for 100K steps. We set attention type to be "original_full" instead of using "block_sparse" during fine-tuning. Training is completed on a single RTX8000 GPU for around 6 days.

Local Transformer Rather than implementing all three types of sparse attention in Bigbird, the Local Transformer relies only on the local attention, i.e., each token attends to neighbors within a local window. The maximum attainable sequence length scales linearly with the number of layers, e.g., with window size k, the token representation at layer l theoretically covers information in a range of $k \times l$ tokens.

Routing Transformer [148] Different from previously described models which use *position*-based sparse attention, the Routing Transformer employs *content*-based sparse attention. Namely, each token are routed to clusters and the attention is performed only within each cluster. The clustering operation effectively reduces the quadratic complexity in length L to $O(L^{1.5})$. Both the RT and LT checkpoint we used were trained on PG-19 [145]. For both RT and LT, we evaluate on single RTX8000 GPU.

GPT-2/3 The GPT models have a lot shorter maximum input length than the rest models we evaluated. While GPT-2 model does not use sparse attentions at all, GPT-3 model adopts alternated layers of sparse and dense self-attention. We use the GPT-2 large model, which was pre-trained on data scraped from the Internet. The GPT-3 model was pre-trained on a mixture of filtered CommonCrawl, WebText2, Books1, Books2, and Wikipedia.

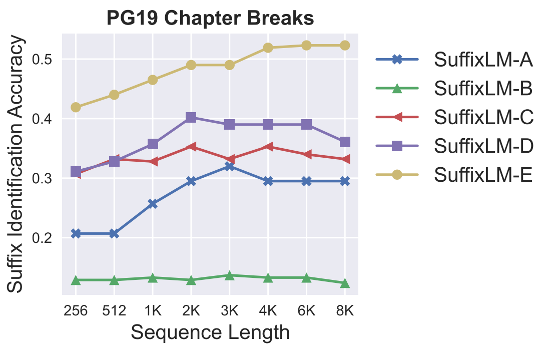


Figure B.1: Performance of each SuffixLM variant. Detailed information about each variant is included in Appendix B.3.

B.3 Finding the best SuffixLM

As there are no prior long-range segment-level LM architectures that we can borrow from, we experiment multiple design choices and report the result of only the best performing one in the main text. For all variants, we use RoBERTa-base [113] as the encoder to obtain the encoded segment representation. This is done by extracting the representation of the $[CLS]$ token prepended at the beginning of each sequence. We describe five variants below.

- **SuffixLM-A** This variant contains a frozen RoBERTa-base encoder and a SuffixLM using a 6-layer Transformer as the base architecture.
- **SuffixLM-B** This variant contains a frozen RoBERTa-base encoder and a SuffixLM using a 6-layer average-attention Transformer as the backbone. The motivation of using uniform distribution for attention weights is to encourage the model to get more information from the distant context rather than rely too much on local context.
- **SuffixLM-C** This variant is essentically SuffixLM-A but during training we perform "segdrop" – stochastically dropping prefix segments with probability 0.2[1] when performing self-attention. When the local segments are dropped, the model has to predict the next segments with only the distant context, which also encourages learning better long-range prefix representations.
- **SuffixLM-D** Instead of freezing the encoder, this variant fine-tunes part of the encoder and the rest is the same as SuffixLM-A. Due to limited memory capacity, we only fine-tune the last two layers of the RoBERTa-base.
- **SuffixLM-E** This model is the same as SuffixLM-D except that we truncate the encoder to just the two tunable layers and train all parameters in the encoder including the embedding parameters.

[1]Tried {0.1, 0.2, 0.4}, 0.2 works the best.

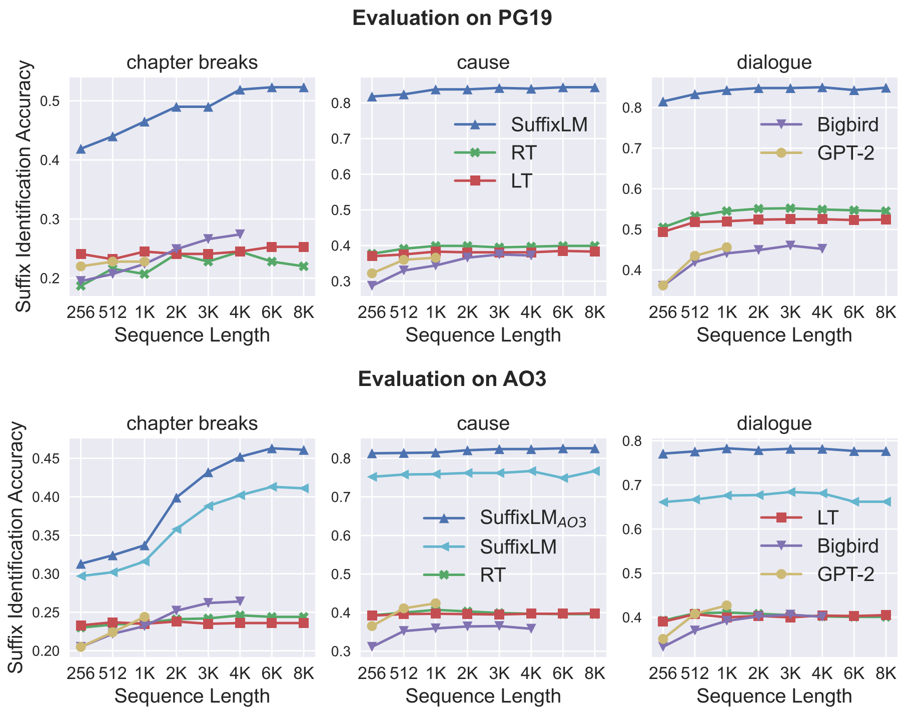


Figure B.2: Evaluation results on both CHAPTERBREAK$_{PG19}$ and CHAPTERBREAK$_{AO3}$.

All SuffixLMs with frozen encoders are trained with average sequence length of 10240 tokens for up to 60k steps, and the one with trainable encoder is trained for max 120k steps. The dimension of the model is 768, hidden dimension 2048,attention heads 8. The peak learning rate is 0.0001 with warm up steps 4000. We train SuffixLM on entire PG-19 dataset and evaluate the best checkpoint selected by dev loss. We use segment size 128 in all SuffixLMs we trained. Each segment starts from a new sentence, if not reaching 128 tokens, we pad with a special '<pad>' token. For very long sentences, the part exceeding 128 tokens overflows to the next segment. We plot the suffix identification accuracy of each variant on CHAPTERBREAK while feeding in prefixes of increasing length. As shown in Figure B.1, SuffixLM-E outperforms all other variants across various prefix lengths. Therefore in the main text, all SuffixLM refers to the SuffixLM-E variant. Note that one limitation of SuffixLM is it exclusively models on segment-level, which prohibits it from performing token-by-token generation and thus impossible for us to evaluate perplexity.

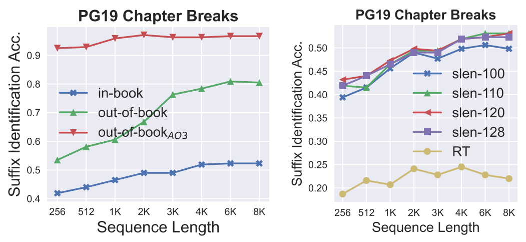


Figure B.3: **Left:** In-book vs. out-of-book. **Right:** SuffixLM performance when evaluated with different suffix length. The variation in suffix length does not explain the large gap between SuffixLM and token-level LMs.

B.4 Suffix perplexity

Although the task of ChapterBreak is to identify gold suffix from negatives, we also present the gold suffix perplexity of next-token prediction LMs. Note that all models were trained or fine-tuned on PG-19 except for GPT-2/3. As these models use different tokenizers, the 128-token suffix may cover different number of words, to make the results comparable, we convert the subword-level perplexity to word-level by multiplying a constant to the log probability value of each model. For RT/LT, we multiple by 1.248 as used in the official repository. We multiply the value by 1.30 for GPT-2, and 1.22 for Bigbird. These values are estimated via the subword/word ratio on validation set of PG-19. Our fine-tuned Bigbird model achieves the lowest perplexity on PG-19, even better than Routing Transformer or Local Transformer. This implies that context from long-range is not necessary for achieving low perplexity since the maximum input length of Bigbird is half that of RT/LT.

B.5 In-book vs. Out-of-book

This section is better read after reading through § 3.3. In this analysis experiment, we show why it is better that the negatives are from the same narrative as the gold suffix. We evaluate our upper bound model SuffixLM on PG-19 set when the negatives are out-of-book suffixes, and plot the suffix identification accuracy in Figure B.3. When evaluate against out-of-book negatives, this suffix identification task is almost solved by our SuffixLM, especially when the out-of-book examples are from another split in ChapterBreak. The extremely high accuracy under out-of-book setup suggests the segment representation from different books are easy for SuffixLM to distinguish, thus we adopt a harder setup where the negatives are from the same book. Besides, in-book negatives may contain the same re-occurring named entities or rare words, which require solid understanding of the prefix to differentiate the gold from the distractors.

B.6 Various Discourse Relationships

In addition to chapter breaks, we also evaluate the other two types of discourse boundary examples introduced in Appendix B.1. As shown in Figure B.2, for all suffix types other than chapter breaks, the evaluated models stop improving as the sequence length grows to more than 2K tokens long. However, there is a significant increasing trend in chapter breaks for SuffixLM. For the rest models, the performance is either flat or not improving. On the AO3 split, the accuracy of SuffixLM improves for $\sim 15\%$ as the sequence length increases from 256 to 8K, whereas the improvement of RT is only $\sim 1.4\%$. This is in contrast with SuffixLM's $\sim 1.5\%$ and RT's $\sim 0.3\%$ improvement for the 'cause' examples. We draw two conclusions from these observations: (1) the chapter breaks examples form a special case where longer prefix is preferred in order to pick the correct continuation. (2) By comparing the relative improvement, the token-level LMs fall far behind the SuffixLM, which is, besides the absolute performance gap, another evidence that current LRLMs do not effectively leverage long-range context for sequence tasks requiring discourse-level understanding.

B.7 Tackle difference in Tokenizers

As the models we evaluated use different tokenizers, there are small variations in term of suffix length, i.e., the 128-token suffix may cover different number of words. To understand how the difference in length impacts validity of evaluation, we evaluate SuffixLM with various suffix lengths. Figure B.3 (right) indicates even though there are small variances when the suffixes are of different lengths, the large gap between SuffixLM and Routing Transformer still remains, thus the difference in suffix length does not explain the large performance gap.

B.8 Error analysis

Models struggle with location and event shifts: Among the 300 examples we annotated in Section 3.2, 89 examples were wrongly predicted by all models we have evaluated. By breaking the incorrectly predicted examples into category as presented in Table 3.1, we find that models tend to make wrong prediction when there is a shift in location or event, and when plots are continuous in timeline.[2]

[2]Detailed numbers are included in Appendix B.8.

Category	**Definition**	**Ratio**
Events	Previous event ends and new event starts	0.74
	Previous event continues into next chapter	0.26
Actors	Change of perspective or character in focus	0.43
	No change in POV or main character	0.57
Locations	Change of location	0.64
	No change in location	0.36
Continuity	Discontinuous but chronological	0.24
	Continuous	0.62
	Analepsis	0.03
	Parallel	0.11

Table B.3: Human annotation on 89 examples sampled from ChapterBreak$_{AO3}$where all models make the wrong prediction. 74% errors come from the examples where new event starts from the new chapter and 64% errors from the change of location.

APPENDIX C

APPENDIX FOR CHAPTER 4

C.1 Evaluation setup

C.2 Supplementary details of analysis

C.3 GPT-4 Multiple-choice setup performance

While our primary focus is on the generative QA setup in the main text, we provide GPT-4's performance under the standard multiple-choice setup here in the Appendix. On the entire QuALITY dev set, GPT-4 achieves an accuracy of 84.4%. For the 1000 challenging question set, GPT-4 reaches an accuracy of 78.7%, nearly 10 points higher than the GPT-4 zero-shot generative baseline. This result suggests that there is still room for improvement in GPT-4's generative answers. We also observe that GPT-4 is sensitive to the ordering of the provided options. We further evaluate GPT-4 with three shuffled versions of the options (swap A and D, B and C; swap A and C, B and D; swap A and B, C and D). While the overall accuracy of these versions remains similar, the questions that are consistently answered correctly across all four option orderings drop to 68.7%. This result raises the question of whether GPT-4 truly "understands" the question and further motivates the generative QA setup.

Human annot. category	**# of plans**
Unnecessary steps	15
Steps can be merged	2
Plan misses information	3
Plan may lead to incorrect answer	4
Plan needs slight edit	7

Table C.1: human freeform feedback aggregation

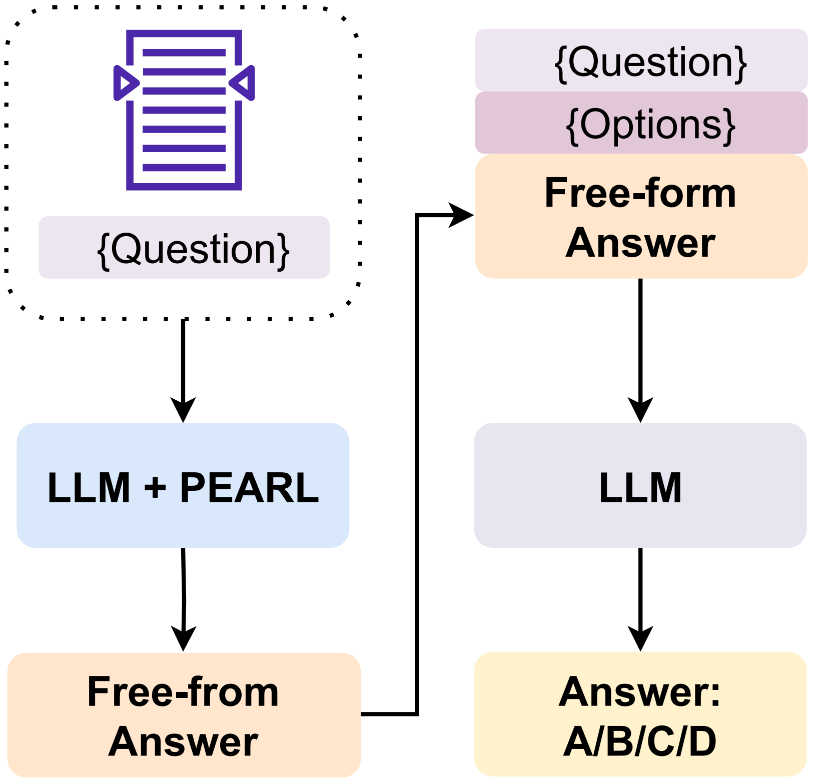


Figure C.1: Generic illustration of our evaluation setup. Given the article and question, we prompt an LLM with PEARL to generate a long-form answer, which is later mapped to one of QuALITY's multiple-choice options by the LLM itself.

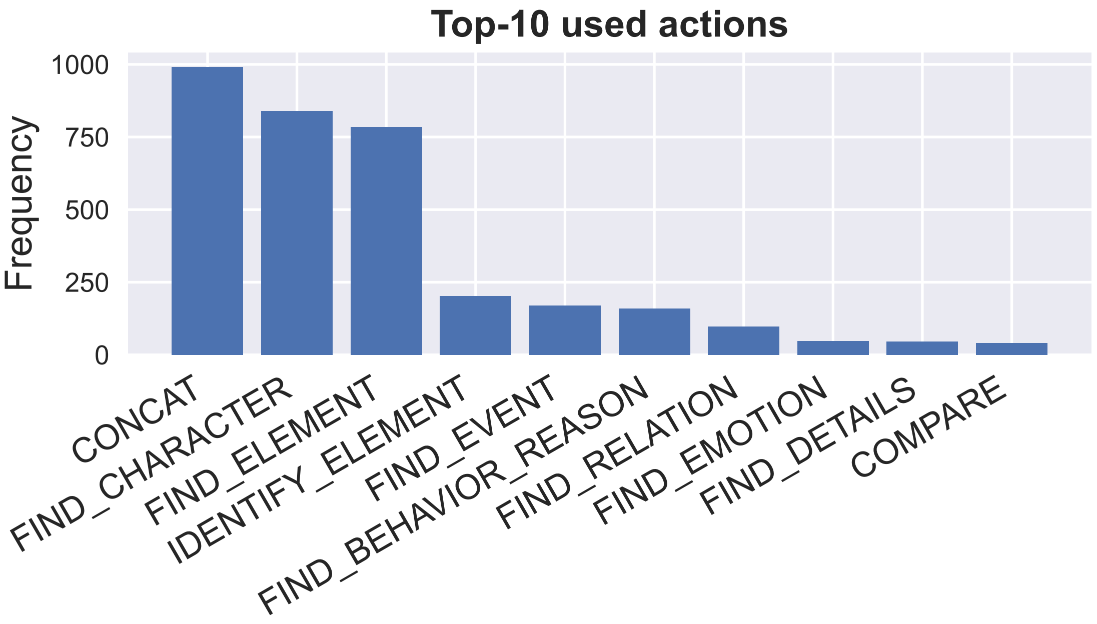


Figure C.2: Top-10 most frequently used actions by PEARL.

	Count	**GPT-4 PEARL**	**GPT-4 zero-shot**
Description	320	0.73	0.73
Why/reason	316	0.79*	0.71*
Symbolism/interpretation	262	0.73	0.70
Person	216	0.75*	0.66*
Event	199	0.69	0.68
Not/except	118	0.70*	0.53*
How/method	100	0.74	0.73
Relation	89	0.71	0.65
Entity	74	0.64	0.68
Numeric	49	0.67	0.78
Location	32	0.59	0.59
What if	21	0.71	0.76
Object	14	0.64	0.64
Duration	18	0.78	0.89
Finish the sentence	10	0.9	0.8

Table C.2: Accuracy by reasoning types. * denotes statistically significant improvement with p-val < 0.005.

Err. Category	Question	Model Generated Plan or Answer	Explanation
True Negative - Error in Plan (17.5%)	Does the tone of the passage shift at all, and if it does, how does it shift?	(Plan) ... 3. tone_shift = COMPARE(CTX, tone_initial, tone_final, "tone") : Compare the initial and final tones of the passage to determine if there is a shift...	Since the plan only compares the initial and final tone, the final answer fails to capture the changes in between, thus leads to an incorrect answer.
True Negative - Error in Exec. (55%)	How many adult characters have speaking roles?	(Answer) In the input article, there are 3 adult characters with speaking roles...	The correct answer involves two characters, whereas PEARL's response mistakenly includes an additional name. The plan for this question is reasonable, but the problem stems from the execution of individual steps.
False Negative (12.5%)	Does the story have a good ending? (Answer: Unclear, the story ends as Evelyn enters a dangerous situation)	(Answer) ...However, the ending of the story is somewhat ambiguous and leaves several questions unanswered. For instance, it is unclear whether Evelyn will be able to successfully complete her mission ...	In this example, the model output is correct, but is mapped to an incorrect distractor option, which contains direct contradictions with the model output.
Other (15%)	Who would most likely enjoy this story, of the following options?	(Answer) The target audience of the input article is science fiction enthusiasts, particularly those who enjoy stories about space exploration, alien encounters...	The model output is not necessarily wrong in the absence of options. However, when provided with options during mapping stage, one of the other options is clearly better.

Table C.3: Examples of errors exhibited by PEARL answers.

C.4 Verify Accuracy of Answer Mapping

As demonstrated in Section 4.6, the mapping stage is not always reliable. To understand the frequency of mapping errors, we conduct a small-scale human answer mapping study. We recruit three professionals on Upwork. We randomly select 50 questions and ask annotators to read PEARL output and then map it to one of the provided options. On average, annotators agree with $\sim$83% of GPT-4 mappings, with inter-annotator agreement on four-class settings of $\kappa = 0.677$. For questions where annotators disagree with each other or do not concur with GPT-4, they tend to be those that can be mapped to than one option or none of the options. We believe this level of accuracy is decent enough to let GPT-4 perform the mapping step for evaluation.

C.5 Can PEARL benefit from more human-written examples?

While we have employed self-refinement and executed the model-generated plan to ensure the quality of ICL demonstrations, it is natural to ask if we can further improve PEARL by incorporating more quality-assured human-written examples. Therefore, we evaluate an alternative version of PEARL in which the in-context examples for plan generation are replaced with 11 human-written examples. This variant achieves 70.3, 76.8, and 72.3 on the long split, the short split, and the total evaluation data, respectively. These results suggest that additional human input may not be necessary to achieve strong results.

C.6 Prompts and templates used in PEARL

C.7 Human feedbacks on model-generated plan

Prompt for Action Mining

[Actions]
- CONCAT(S1, S2, ...) : Concatenate the input S1, S2, ...
- EXTRACT(CTX, X) : Extract the exact wording that X is referring to from input CTX.
- FIND_X(CTX, X): Find and summarize all relevant information about X in the input CTX.
- FIND_REASON(CTX, X) : Find and summarize the cause or reason of X given input CTX.
- FIND_MORAL(CTX) : Find the intended lesson or moral of the input CTX.
- SUMMARIZE(CTX): Provides a general summary about the given CTX.
- SUMMARIZE_X(CTX, X) : Provides a summary about X given the provided input CTX.

[Instructions]
Suppose you are given a question about an article as well as a list of actions that you can execute to solve the question (shown above). You can imagine the actions as functions in a program, where you have input arguments and output. The output of an action can be fed as input to another action. The output of the final action will be the answer to the given question. Suppose you haven't read the article yet, please present a sequence of actions that you would use to answer the question.

Here are a few examples:

Question:
What is the "space cafard" that Si describes?

My new actions:
- COMPREHEND(CTX, X) : Provide a detailed comprehension of X given the input CTX.

My sequence of actions:
1. snippet = EXTRACT(CTX, "space cafard") : Extract the exact wording regarding "space cafard" from the input CTX.
2. ans = COMPREHEND(CTX, X) : Provide a detailed comprehension of the input X given the input CTX.

Question:
Why did the author write the article?

My new actions:
- None

My sequence of actions:
1. moral = FIND_MORAL(CTX) : Find the intended lesson or moral of the input CTX.

Your answer must follow the following rules: 1. The present sequence should be minimal, i.e., no unnecessary actions. 2. The sequence of actions should be specific and cover every detail about the question. 3. The sequence of actions should use as many as existing actions as possible. 4. It is fine to create new actions, however, the created new actions should be maximally reusable and generalizable to other reading comprehension questions. 5. The arguments should cover all the details of the given question.

[Question]
{Question}

[Answer]
Now please provide the plan for the above question.
Your answer should follow the format:

My new actions (if any):
- my_new_action_1(here goes the arguments) : [one-sentence explanation]
- my_new_action_2(here goes the arguments) : [one-sentence explanation]
...

My sequence of actions:
1. output_1 = action_1(here goes the arguments) : [one-sentence explanation]
2. output_2 = action_2(here goes the arguments) : [one-sentence explanation]
...

Table C.4: Prompt for action mining. {Question} indicates the placeholder for filling in training set question. In this stage, we only care about the new actions proposed by the model.

Mined Actions after reducing number of actions with LLM
ANALYZE(CTX, X, Y) # Analyze the relationship, attitude, or feelings between X and Y, or the character, language, tone, or symbolism of X given the input CTX.
COMPARE(CTX, X, Y, Z) # Compare X and Y in the context of Z, considering aspects such as abilities, assets, attractiveness, behavior, concerns, contributions, cultures, events, experiences, feelings, ...
COMPREHEND(CTX, X) # Provide a detailed comprehension of X given the input CTX.
CONCAT(S1, S2, ...)
DEFINE(CTX, X) # Provide the definition of X given the input CTX.
DESCRIBE(CTX, X, Y) # Provide a description of X in terms of Y, such as character, genre, or introduction given the input CTX.
EVALUATE(CTX, X, Y) # Evaluate aspects such as feeling, outcome, performance, personalities, risk, or truth of X in relation to Y given the input CTX.
EXCEPT(CTX, LIST) # Find the item that is not mentioned in the input CTX but is present in the given..
EXPLAIN_PROCESS(CTX, X) # Provide a detailed explanation of the process X given the input CTX.
FIND_BARRIERS_CAUSES(CTX, X) # Find and summarize the remaining barriers or causes related to X given the input CTX.
FIND_BEHAVIOR_REASON(CTX, X) # Find the reason behind the behavior X given the input CTX.
FIND_BENEFIT(CTX, X) # Find the direct benefit of X given the input CTX.
FIND_BEST(CTX, X, Y) # Find the best X in the context of Y given the input CTX.
FIND_CHARACTER(CTX, X) # Find and summarize the character traits, transformation, and changes of X given the input CTX.
FIND_COMMON(CTX, X, Y, Z) # Find the common ground, characteristics, or commonalities between X, Y, and Z given the input CTX.
FIND_CONDITION(CTX, X, Y) # Find the condition, outcome, or consequences related to X and Y given the input CTX.
FIND_CONFLICT_CONCERN(CTX, X, Y) # Find the conflict, concern, or disagreement between X and Y given the input CTX.
FIND_CONSISTENCY(CTX, X) # Determine if X is consistent throughout the input CTX.
FIND_DECISION(CTX, X) # Find the decision, factor, or event that influenced X's decision in the input CTX.
FIND_DESCRIPTION(CTX, X) # Find all descriptions, characteristics, or words that describe X given the input CTX.
FIND_DETAILS(CTX) # Find all the details about a topic (e.g., contract, city-state) discussed in the input CTX.
FIND_DIALOGUE(CTX, X, Y) # Find the dialogue between X and Y in the input CTX.
FIND_DIFFICULTY_DANGER(CTX, X) # Find the most difficult aspect, challenge, or danger faced by X in the given input CTX.
FIND_ELEMENT(CTX, X, Y) # Find the element X related to Y given the input CTX. This function can cover message, method, metrics, mismatch, mission, mistake, most likely, motif, motivation, nationalities, negative critique, negative effect, next event, normal, objective, obstacles, ...
FIND_EMOTION(CTX, X, Y) # Find the emotion or feeling X feels towards Y given the input CTX.
FIND_ENDING(CTX, X) # Find the ending or conclusion of X's story or the input CTX.
FIND_EVENT(CTX, X) # Find the event involving X in the input CTX (e.g., betrayal, change, climax).
FIND_EVIDENCE_EXAMPLE(CTX, X) # Find evidence or an example supporting X given the input CTX.
FIND_EXCEPTION(CTX, X, Y, Z) # Find the exception or characteristic that is not common among X, Y, and Z given the input CTX.
FIND_EXPECTATION(CTX, X) # Find the expectation, assumption, or impact about X given the input CTX.
FIND_EXPLANATION(CTX, X) # Find the most likely explanation, critique, or doubt for X given the input CTX.
FIND_FACT_FALSE(CTX, X) # Find a definite fact or false statement about X given the input CTX.
FIND_FEARS_DISTRACTIONS(CTX, X) # Find the fears, concerns, or distractions of X given the input CTX.
FIND_FEATURES(CTX, X) # Find all the features that X cares about given the input CTX.
FIND_FIRST_INSTANCE(CTX, X) # Find the first instance of X happening in the input CTX.
FIND_FLAW(CTX, X) # Find the greatest flaw of X given the input CTX.
FIND_FOCUS(CTX, X) # Find the person or object that is focused on the most in the input CTX, given a list of X.
FIND_FORESHADOW(CTX, X, Y) # Find the instance where X foreshadows Y in the input CTX.
FIND_FUTURE(CTX, X) # Find the future, predicted outcome, or action of X given the input CTX.
FIND_GRIEVANCE(CTX, X) # Find and summarize the grievance X has against something or someone in the input CTX.
FIND_HALO_EFFECT(CTX, X) # Find and summarize one halo effect of X given the input CTX.
FIND_HUMBLENESS(CTX, X) # Find the instances of humbleness presented by X in the input CTX.
FIND_HYPOTHETICAL(CTX, X) # Find the hypothetical outcome or consequence of X given input CTX.
FIND_IMAGINATION(CTX, X) # Find and summarize how X imagines something in the input CTX.
FIND_IMPACT(CTX, X, Y) # Find the event or experience that had the strongest impact on X's Y given the input CTX.
...

Table C.5: A subset of mined actions from training set questions.

Prompt for Generating Plan

[Actions]
ANALYZE(CTX, X, Y) # Analyze the relationship, attitude, or feelings between X and Y, or the character, language, tone, or symbolism of X given the input CTX.
COMPARE(CTX, X, Y, Z) # Compare X and Y in the context of Z, considering aspects such as abilities, assets, attractiveness, behavior, concerns, contributions, cultures, events, experiences, feelings, focus, intelligence, irony, nationalities, performance, praise, reactions, reviews, secretiveness, time periods, treatment, truth, or worlds given the input CTX.
COMPREHEND(CTX, X) # Provide a detailed comprehension of X given the input CTX.
CONCAT(S1, S2, ...)
DEFINE(CTX, X) # Provide the definition of X given the input CTX.
DESCRIBE(CTX, X, Y) # Provide a description of X in terms of Y, such as character, genre, or introduction given the input CTX.
EVALUATE(CTX, X, Y) # Evaluate aspects such as feeling, outcome, performance, personalities, risk, or truth of X in relation to Y given the input CTX.
...
{List of Actions as shown in Table C.5}

[Instructions]
Suppose you are given a question about an article, as well as a list of potential actions (shown above) that you can execute to solve the question . You can imagine the actions as functions in a program, where you have input arguments and output. The output of an action can be fed as input to another action. Please present a sequence of actions that you would use to answer the question after you read the article. The sequence of actions should be specific and cover all the details about the question. Please prioritize using the actions presented in the list above. If you need to add new actions, please follow the format below. Please assign the output of each action with a distinct name, which can be passed into other actions as argument. Think twice before you provide your answer. Make sure your answer is valid, clear, and easy to understand. Keep the answer simple and remove any unnecessary steps. Do not use list comprehension or dictionary comprehension. Keep each action minimally simple. If a question is unanswerable (e.g., requires options), collect as much information as possible from the input such that it will be answerable when provided with options. Your answer should follow the format:
”’
New actions:
- new_action_1(arguments) : [one-sentence general explanation] or "-None" if there no need to add new actions
- new_action_2(arguments) : [one-sentence general explanation] or "-None" if there no need to add new actions

1. output_1 = action_1(here goes arguments) : [one-sentence explanation]
2. output_2 = action_2(here goes arguments) : [one-sentence explanation]
...
”’

The following are a few examples

Question: "How do Ross and Mehta view Brown’s acquisition of the magazine?"

Answer:
New actions:
- FIND_OPINION(CTX, X, Y) : Find the opinion of X about Y given the input CTX

1. ross = FIND_CHARACTER(CTX, "Ross") : Identify who Ross is in the input article
2. mehta = FIND_CHARACTER(CTX, "Mehta") : Identify who Mehta is in the input article
3. brown = FIND_CHARACTER(CTX, "Brown") : Identify who Brown is in the input article
4. magazine_acquisition = FIND_EVENT(CTX, "Brown’s acquisition of the magazine") : Find the event of Brown’s acquisition of the magazine in the input article
5. ross_opinion = FIND_OPINION(CTX, ross, magazine_acquisition) : Find the opinion of Ross about Brown’s acquisition of the magazine
6. mehta_opinion = FIND_OPINION(CTX, mehta, magazine_acquisition) : Find the opinion of Mehta about Brown’s acquisition of the magazine
7. ans = CONCAT(ross_opinion, mehta_opinion) : Combine the opinions of Ross and Mehta on Brown’s acquisition of the magazine to form the final answer
... {more few-shot examples} ...

[Question]
Now you are given a question about an article:
{question}
Please provide a plan (sequence of actions) that can arrive to the answer after reading the article. As the corresponding options are not provided for the question, when the question is not answerable without the options, simply collect as much information as possible from the input such that it will be answerable with the options. Make sure the plan you generate is valid and faithful to the question.

[Answer]

Table C.6: Prompt for generating plan given a question, which is filled in the placeholder {question}.

Prompt for Executing Single Step of the Plan
Article {Long document} End of Article — Please read the above text first, and then follow the instructions below. [Instructions] {Mined action and corresponding definition of current step. Example shown below.} FIND_EMOTION(CTX, X, Y) # Find the emotion or feeling X feels towards Y given the input CTX. {Current step in the plan generated in the previous stage. Example shown below.} kolin_opinion = FIND_EMOTION(CTX, kolin, "becoming a tree") {Value assignment of input argument(s)} X = "In the story, Kolin is a steward from the Planetary State of Haurtoz who is part of a scouting party sent to explore a planet after their ship, the Peace State, is damaged. Kolin is unhappy with the oppressive regime on Haurtoz and dreams of escaping it. While exploring the planet, he encounters a tree named Ashlew, which is actually a man who has transformed into a tree. Ashlew tells Kolin about the Life, a powerful entity on the planet that can help individuals change their form...{Output from previous step.}" Y = "becoming a tree" [Answer] {A brief description of current step.} (Find the emotion or feeling Kolin has towards becoming a tree himself in the input article)

Table C.7: Prompt for executing a step in a plan. Prompt of this step is a template with placeholders which will be filled with the output from previous step(s).

Prompt for Answer Mapping

Relevant information for answering the question:
 	
{Generative answer output by PEARL.}
 	
Question: {question}
{Options}
 	
Read the relevant information about the article and answer the question by selecting the best option above. Only one of them is correct.
 	
Answer (select from A, B, C, D):

Table C.8: Prompt for mapping generative answer to provided options

Question & Options	Plan	Human feedback
What was Mrs. Jamieson's biggest problem in the story? A: She did not understand the Stinson Effect B: She had to raise her son alone C: She was just able to make ends meet D: She had to hide her scar	1. mrs_jamieson = IDENTIFY_ELEMENT(CTX, "Mrs. Jamieson") : Identify who Mrs. Jamieson is in the input article 2. mrs_jamieson_problems = FIND_ELEMENT (CTX, "problems", mrs_jamieson) : Find and list all the problems faced by Mrs. Jamieson in the story 3. biggest_problem = FIND_BEST(CTX, "problem", mrs_jamieson_problems) : Find the biggest problem faced by Mrs. Jamieson in the story 4. ans = CONCAT(mrs_jamieson, biggest _problem) : Combine Mrs. Jamieson and her biggest problem to form the final answer	Steps 2 & 3 can be combined into one. mrs_jamieson_biggest _problems = FIND_ELEMENT (CTX, "biggest problem", mrs_jamieson): Finds the biggest problem faced by Mrs. Jamieson in the story
Which class was the least intense? A: Kung Fu B: Tai Chi C: Aikido D: Brazilian Jujitsu	1. classes = FIND_DETAILS(CTX) : Find all the details about the classes and their intensity levels mentioned in the input article 2. least_intense_class = FIND_LEAST_DESCRIBING_WORD (classes, "intense") : Find the class that is least intense in the given input CTX 3. ans = CONCAT (least_intense_class, classes) : Combine the least intense class and the intensity levels of all classes to form the final answer	Adding the details for the other classes is unnecessary since the question is looking for a single answer, the least intensive class.
Do the Terrans ever come close to winning the battle within the story? A: No, they continually lose B: They win the whole battle with less casualties C: Yes, by the surprise squadron Evelyn leads D: Yes, by Evelyn cloning soldiers into battle	1. terrans = IDENTIFY_ELEMENT(CTX, "Terrans") : Identify who the Terrans are in the input article 2. battle = FIND_EVENT(CTX, "battle") : Find and summarize the battle event within the story in the input article 3. terrans_battle = FIND_RELATION(CTX, terrans, battle) : Find and summarize the Terrans' involvement in the battle from the input article 4. terrans_close_win = FIND_CONDITION (CTX, "Terrans", "close to winning") : Find the condition or events where the Terrans come close to winning the battle in the input article 5. ans = CONCAT(terrans_battle, terrans_close_win) : Combine the Terrans' involvement in the battle and the events where they come close to winning to form the final answer	Step 2 and 3 can be combined: Find and summarize the Terrans' battle event within the story in the input article
What level of depth does the author provide on the subjects they use to make their case? A: Language is really the only thing covered in any depth B: A broad, but not very deep assessment C: They provide the reader with deeper arguments about the monetary system and striking tendencies than anything else D: They provide deep, explanatory statistics to most arguments	1. author = IDENTIFY_ELEMENT(CTX, "author") : Identify the author of the article 2. subjects = FIND_ELEMENT(CTX, "subjects", author) : Find and list all the subjects the author uses to make their case in the input article 3. depth_analysis = ANALYZE(CTX, subjects, author) : Analyze the level of depth the author provides on the subjects they use to make their case in the input article 4. ans = CONCAT(subjects, depth_analysis) : Combine the subjects and the depth analysis to form the final answer for comparing with the options	Very good plan.

Table C.9: Example human feedback from annotators on PEARL-generated plans.