Karlsruhe Institute of Technology

Evaluating and Training Long-Context Large Language Models for Question Answering on Scientific Papers

Master's Thesis of

Lukas Hilgert

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institut for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Jan Niehues
Second reviewer:	Prof. Dr.-Ing. Tamim Asfour
Advisor:	M.Sc. Danni Liu

01. October 2023 – 27. March 2024

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 27. March 2024

. . . .

(Lukas Hilgert)

Abstract

With the number of scientific papers published every year growing and current large language models (LLMs) showing state-of-the-art performance on natural language processing (NLP) tasks, we ask the question if LLMs could be utilized to answer questions on scientific papers to accelerate the work of scientists and improve science communication. To investigate this question, we experiment with efficient, open-source LLMs that are based on the popular LLaMA 2 model. As scientific papers are often thousands of words long, we select special versions whose ability to process long context was enhanced. To evaluate and improve these models, we use the Qasper dataset which contains 5,049 questions on 1,585 NLP papers. It also provides answers written by annotators. We analyze how well the LLMs handle longer papers and questions that can only be answered by accessing information from far out paragraphs. During our experiments, we see that the performance of these LLMs drops with growing length and position of relevant information. Using recent compute- and memory-efficient training techniques, we are able to fine-tune the considered models on long contexts such as papers. We design different prompts for the LLMs to solve the question answering task: From simple prompts to chain-of-thought prompts that first extract the relevant paragraphs from the paper and then answer the question. We also try splitting the paper into shorter, easier to process parts and modifying the attention algorithm to ignore less important parts of the context. While we still observe a performance loss with increased context length, our measures reduce the effects of this flaw, and we can achieve an F_1 score of 52.73 after fine-tuning with our best approach compared to a score of 31.07 of the unmodified model. This best score on a subset of the Qasper test set used in the long-context benchmark ZeroSCROLLS is 2.03 F_1 points better than GPT-4 and only 4.17 points worse than the best model.

Zusammenfassung

Da jedes Jahr die Anzahl veröffentlichter wissenschaftlicher Artikel steigt und große Sprachmodelle (LLMs) den aktuellen Stand der Forschung in der Verarbeitung natürlicher Sprache (NLP) darstellen, stellen wir die Frage, ob man eben jene LLMs nutzen könnte, um die Arbeit von Forschenden in der Wissenschaft zu beschleunigen und die Wissenschaftskommunikation zu verbessern. Um dieser Frage nachzugehen, experimentieren wir mit effizienten, öffentlich zugänglichen LLMs, die auf dem populären Modell LLaMA 2 basieren. Weil wissenschaftliche Artikel oft mehrere tausend Wörter lang sind, wählen wir Versionen deren Fähigkeit zur Verarbeitung von langem Kontext verbessert wurde. Um diese Modelle zu evaluieren und zu verbessern, nutzen wir den Qasper-Datensatz, der 5 049 Fragen zu 1 585 NLP-Artikeln enthält. Außerdem enthält es Antworten von Annotatoren. Wir analysieren, wie gut die LLMs lange Artikel und Fragen handhaben, die nur beantwortet werden können, indem auf weit entfernte Absätze zugegriffen wird, sehen wir, dass die Leistung der LLMs mit wachsender Länge und Position der relevanten Information abfällt. Durch das Verwenden von rechenleistungs- und speichereffizienten Techniken sind wir in der Lage die betrachteten Modelle auf langem Kontext wie Artikeln feinzujustieren. Wir entwickeln verschiedene Anweisungen für die LLMs, um die Aufgabe des Fragebeantwortens zu lösen: Von einfach Anweisungen bis zu Gedankengang-Anweisungen, die zuerst die relevanten Absätze extrahieren und dann die Frage beantworten. Wir versuchen außerdem, die Artikel in kürzere, einfacher zu verarbeitende Abschnitte aufzuteilen und den Attention-Algorithmus zu modifizieren, damit er weniger wichtige Teile des Kontexts ignoriert. Während wir immer noch eine Abnahme der Leistung mit längerem Kontext beobachten, reduzieren unsere Maßnahmen die Auswirkungen dieser Schwäche und wir können ein F_1-Maß von 52.73 nach Feinjustierung mit unserem besten Ansatz im Vergleich zu einem Wert von 31.07 des unveränderten Modells erreichen. Dieser beste Wert auf einer Teilmenge des Qasper-Testteils, der im Vergleichstest ZeroSCROLLS für langen Kontext benutzt wird, ist 2.03 F_1-Punkte besser als GPT-4 und nur 4.17 Punkte schlechter als das beste Modell.

Contents

Abstract i

Zusammenfassung iii

1. Introduction 1
- 1.1. Problem Statements & Research Questions 1
- 1.2. Thesis Outline 2

2. Background and Related Work 3
- 2.1. Sequence Representation in NLP 3
  - 2.1.1. Language Modeling Task 3
  - 2.1.2. Transformers 4
  - 2.1.3. Pre-trained Models 5
- 2.2. Large Language Models 7
  - 2.2.1. Impact of Scaling 7
  - 2.2.2. Instruction Tuning and RLHF 8
  - 2.2.3. Representative LLMs 8
  - 2.2.4. Efficient Implementations for LLMs 10
- 2.3. Question Answering Task 13
  - 2.3.1. Challenges and Approaches 13
  - 2.3.2. Datasets 15

3. Approach 17
- 3.1. Task-agnostic Techniques 17
- 3.2. Task-specific Approaches 18

4. Experiments and Results 21
- 4.1. Experimental Setup 21
  - 4.1.1. Dataset 21
  - 4.1.2. Data Preprocessing 22
  - 4.1.3. Hard- and Software 23
  - 4.1.4. Hyperparameters 23
  - 4.1.5. Models 24
- 4.2. Evaluation Metrics 24
  - 4.2.1. Analysis by Context Length / Position 24
- 4.3. Results and Discussion 25
  - 4.3.1. Zero-shot Prompt 25
  - 4.3.2. QLoRA Fine-tuned 27

4.3.3. Evidence only 29
4.3.4. Two-step Prompts 31
4.3.5. Split Prompt 35
4.3.6. Modify Attention Algorithm 36
4.3.7. Final Comparison against Baselines 36

5. Conclusion **39**
5.1. Answers to Research Questions 39
5.2. Future Work 40

Bibliography **41**

A. Appendix **51**
A.1. Prompts 51
A.1.1. ZeroSCROLLS 51
A.1.2. LongBench (our version) 51
A.1.3. Evidence only 51
A.1.4. Two-turn 52
A.1.5. Two-turn, Advanced (Prefix for Evidence, Question Repeated) . . 52
A.1.6. Split 53
A.2. Answers from Models 53
A.2.1. LongChat, Unanswerable 53
A.3. Results 54
A.3.1. Two-step Prompt, r = 16 54

List of Figures

2.1. Original Transformer architecture 4
2.2. (Q)LoRA 10
2.3. FlashAttention 12
2.4. Positional Interpolation 14

List of Tables

4.1. Qasper dataset statistics, question / evidence types 21
4.2. Qasper dataset statistics, length / evidence position 22
4.3. All models, LongBench prompt, zero-shot 26
4.4. All models, LongBench prompt, zero-shot, analysis 26
4.5. LongChat-32k, LongBench prompt, QLoRA fine-tuned 27
4.6. Vicuna-16k, LongBench prompt, QLoRA fine-tuned 27
4.7. Vicuna-4k, LongBench prompt, QLoRA fine-tuned 28
4.8. All models, LongBench prompt, QLoRA fine-tuned, analysis 28
4.9. All models, evidence only, zero-shot 29
4.10. All models, evidence only, zero-shot, analysis 30
4.11. LongChat-32k, evidence only, QLoRA fine-tuned 30
4.12. Comparison: LongChat-32k QLoRA fine-tuned on evidence only and full paper 31
4.13. Comparison: LongChat-32k QLoRA fine-tuned on evidence only and full paper, analysis 31
4.14. LongChat-32k, two-step prompt 32
4.15. LongChat-32k, two-step prompts, analysis 33
4.16. LongChat-32k, two-step prompt, vary temperature 34
4.17. LongChat-32k, advanced two-step prompt 34
4.18. Comparison of evidence extraction on <4k tokens 35
4.19. Comparison of split prompts to each other and other prompts 35
4.20. Vicuna-4k and LongChat-32k, modified attention, analysis 36
4.21. Comparison: Our approaches against baselines from Qasper paper, dev and test set 37
4.22. Comparison: Our approaches against baselines from Qasper paper, dev and test set, evidence 38
4.23. Comparison of our approaches against baselines from ZeroSCROLLS benchmark, ZeroSCROLLS subset 38

A.1. Two-step prompt, r = 16 54

1. Introduction

The number of scientific papers published every year is growing exponentially [26]. This creates a problem for scientists but also the general public to keep up with the developments in science. A natural language processing (NLP) system that can reliably answer questions on scientific papers could help in this situation. Scientists would benefit from such a system as it could provide a quick way to find out if a new paper is relevant to the scientist's work. This could lead to higher productivity as scientists could have a better overview about the current research landscape. Otherwise, they would have to read every paper at least partially which is impractical in terms of time. The alternative is to potentially miss some related research. Also, science communication could be improved as these systems could be made accessible to the general public. If people have a better grasp about the current state in science this could lead to better political decision making in democracies.

Question answering (QA) systems often rely on task-specific machine learning models that can only be used for this purpose. This also applies to rule-based systems, classical machine learning models but also to modern deep learning models. Large Language Models (LLMs) are a newer type of deep learning model trained to be general-purpose models for NLP. Current LLMs are often used in an intuitive, conversational manner as chatbots. There are commercial ones (e.g., OpenAI's ChatGPT, Google's Bard / Gemini) and open-source frameworks (e.g., FastChat with LLMs like LLaMA). They show state-of-the-art (SOTA) NLP performance and even display some reasoning capabilities and would be one contender for the core of a QA system focused on scientific papers. They offer the ability to answer follow-up questions and have an intuitive interface for most users. If the used LLM is reversibly modified for this task, it could still be used in its general-purpose setting.

1.1. Problem Statements & Research Questions

In this thesis we want to find out if we can develop a QA system for scientific papers based on a small open-source LLM using state-of-the-art techniques to improve performance further. The main challenges are the context length, the consequential memory requirements for inference and (possibly) training, and the weaker performance of open-source LLMs compared to commercial ones.

The text part of scientific papers is typically about 3,000 to 10,000 words long [10]. This translates to around 4,000 to 13,333 tokens assuming that one word amounts to around $1.\overline{3}$ tokens [51]. Earlier commercial LLMs like GPT-3 [12] or PaLM [16] have a context window of 2048 (2k) tokens. The context window describes the maximum number of tokens a model can effectively process. The base versions of newer commercial models like GPT-3.5 and GPT-4 have context windows of 4,096 [48] and 8,192 [48] tokens while open-source LLMs like LLaMA 2 [70] offer a 4,096 long context window.

Research Question 1: How well do LLMs process scientific papers – especially considering their length?

As Transformer models which serve as the basis for all previously mentioned LLMs have a quadratic time and space complexity regarding the input length, inference and especially training poses a great challenge. Additionally, the computing resources for university students is limited, therefore this thesis has to rely on a single datacenter GPU at once for most of the experiments. Also, as a question answering system for papers should be available to any interested person, we will focus on small open-source models as they are easier to run. Ideally, after applying methods like prompting and fine-tuning our system should be able to perform at least similar to commercial state-of-the-art LLMs on the task we focus on.

Research Question 2: How much does fine-tuning a small LLM on only a single datacenter GPU increase performance on one specific task compared to bigger and better LLMs?

A question answering system is only useful if the answers are reliably correct in an acceptable number. Therefore, it is important to find out if fine-tuning an LLM to better answer the questions leads only to fitting better to the answer format or if the answers themselves get factually better. Can techniques like prompting lead to better results? How does the system compare to human performance?

Research Question 3: How much can prompting / fine-tuning improve the truthfulness of the answers of the LLM? How high is the human performance?

1.2. Thesis Outline

The rest of this thesis is structured as follows: After this introduction, chapter 2 will cover all concepts and techniques used including the machine learning models and the NLP task. The following chapter 3 focuses on the approaches this thesis proposes to solve the problems and research questions mentioned in the earlier section. After that, we will discuss in chapter 4 our experimental setup and the results. The final chapter 5 will conclude our findings.

2. Background and Related Work

In this chapter, we will introduce all concepts presented in this thesis. The first part will focus on how sequences are represented in natural language processing (NLP, section 2.1). This involves modeling language in general (subsection 2.1.1) and how modern deep learning model work on a basic level (subsection 2.1.2). We will also discuss how they can be pre-trained on unlabeled data (subsection 2.1.3). The second part will cover Large Language Models (LLMs) (section 2.2), how they reached their current sizes (subsection 2.2.1) , and how they are improved further from pre-training (subsection 2.2.2). After that we will describe the specific models used in this thesis (subsection 2.2.3) while also mentioning the techniques employed to allow efficient training (subsection 2.2.4). The final section will explain the underlying question answering (QA) task (section 2.3), the challenges in QA and methods to address these in LLMs (subsection 2.3.1), and the datasets we looked at for this thesis for evaluation and training (subsection 2.3.2).

2.1. Sequence Representation in NLP

Scientific papers consist mostly of text in natural language, they have to be presented in some kind of form to be usable for NLP systems. Sequences of words like sentences or whole bodies of text are often represented as strings of so-called tokens which are the result of the tokenization process which splits the text into machine-readable pieces [47]. This process can produce tokens with different levels of granularity: One extreme is that tokens are characters or bytes leading to long sequences which may cause problems for some machine learning models and also captures almost no semantic meaning. One the other end of the spectrum, it is also possible to use full words as tokens which results in a large vocabulary (meaning individual tokens use more information / space) or many unknown tokens when the same tokenizer is used on unseen data. A compromise are subwords [40, 63] as tokens.

2.1.1. Language Modeling Task

The language modeling task is to "learn the joint probability function of sequences of words in a language" [9]. This function can be described by the following equation with w standing for a sequence of words with the length n (Equation 2.1).

$$p(w) = \prod_{i=1}^{n} p(w_n \mid w_1, \ldots, w_{n-1}) \tag{2.1}$$

The number of possible combinations for a sequence of words grows exponentially with the vocabulary size (typically tens of thousands) as the basis and the context length as the exponent. Not all possible or even all similar sentences will be seen during training. In statistical machine learning, n-gram models generate tables with the conditional probabilities for the next word for a large number of context sequences. With artificial neural networks being universal function approximators [18, 31], they can be trained as language models [9].

2.1.2. Transformers

The Transformer [72] is a type of artificial neural network that utilizes only the attention mechanism (apart from multi-layer perceptron layers) in contrast to previous sequence transduction models which (also) relied on recurrence [5, 66] or convolution [27, 36, 57]. Like in its initial publication where the application was machine translation, Transformer models are primarily used in NLP [23, 58, 72] but also appear in computer vision [24].

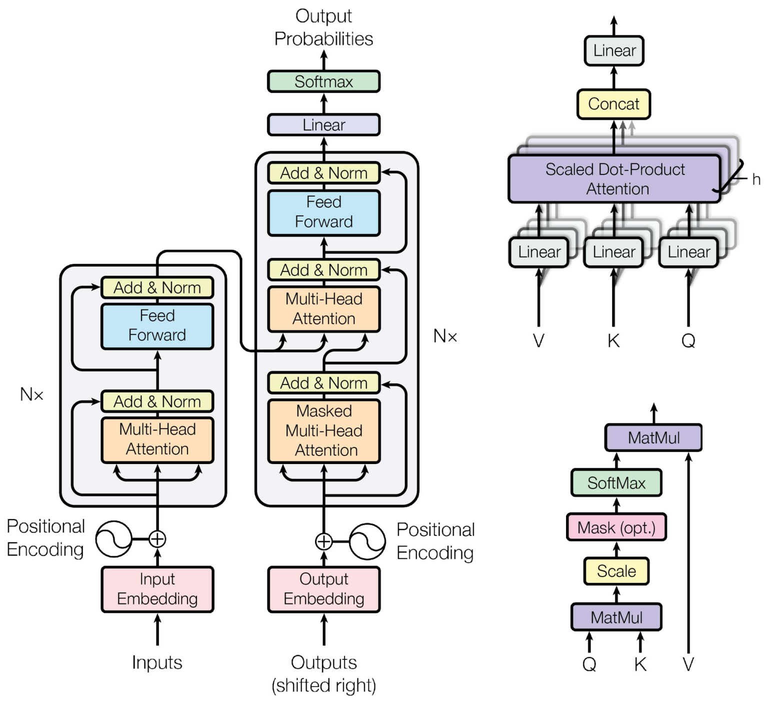

Figure 2.1.: **Left:** The original Transformer architecture [72].
Top right: Structure of Multi-head attention [72].
Bottom right: Structure of Scaled Dot-Product Attention [72].

The original Transformer architecture is split into an encoder and a decoder (Figure 2.1) like preceding deep neural network models [5, 66]. The encoder gets the entire input sequence at once as input while the decoder receives the already generated output sequence as input. Initiated with a start token the decoder generates the output sequence autoregressivly (one token per step) resembling the language modeling task. Both encoder and decoder pre-process their inputs equally: They convert the sequence of tokens (represented as numbers) to vectors via learned embeddings to which they then add positional encoding in the form of sine and cosine functions with different frequencies. This is necessary as the Transformer model has no inductive bias about position.

The encoder and the decoder both consist of multiple blocks which are identical for each part. The neural network layers inside these blocks are a multi-head attention (MHA) and a two-layer feed-forward network (FFN). Each of these modules has a residual connection which adds the module's input to its output. This sum is then layer-normalized although this normalization can also be done at the beginning of each block. The encoder blocks consist of one MHA followed by one FFN while the decoder blocks have another MHA between the first one and the FFN which attends to the final output of the encoder (Figure 2.1). The "normal" self-attentions of the decoder apply a mask to their input to avoid attending to future positions in the sequence (Figure 2.1). Each block gets the output of the previous block as its input. After the final block, a linear ("unembedding") layer processes the output followed by the softmax function to get a probability distribution between 0 and 1 over the vocabulary. This distribution is then used by a decoding strategy to select the next token.

The MHA module is the core of the Transformer model: It allows to process sequences of tokens (may it be text, images, or other modalities) of variable length that are longer than previously possible with Recurrent Neural Networks (RNNs) which suffer from the vanishing gradient problem [8, 54]. All available tokens are linearly projected with different weights resulting in queries Q, keys K and values V. The Attention function computes for each token Q which other tokens K are the most relevant for this step (Equation 2.2). This is done with the dot-product. The result is then scaled by the inverse of $\sqrt{d_k}$ where d_k is the dimension of the keys K to avoid too large values possibly leading to small gradients [72]. After that a normalization with the softmax function converts the attention weights to a pseudo-probability distribution modeling the importance of each token K in the sequence in relation to the currently processed token Q. To get the sequence weighted by their importance the attention weights are multiplied with the values V.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

The authors found it to be beneficial to let the Transformer calculate multiple linear projections Q, K and V (Equation 2.4, Figure 2.1) with smaller dimensions which are then used in multiple attention functions in parallel. The results are concatenated to the original dimension size and fed through a linear layer with weights W^O (Equation 2.3, Figure 2.1).

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.3}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

2.1.3. Pre-trained Models

Two of the first Transformer-based pre-trained LMs were BERT [23] and GPT [58].

BERT The Bidirectional Encoder Representations from Transformers (BERT) model is an encoder-only Transformer model with 110M and 340M parameters, respectively [23].

The number of parameters corresponds to the number of weights of the artificial neural network. The authors pre-trained it with unlabeled data with a sequence length of up to 512 tokens from the BooksCorpus (800M words) and the English Wikipedia (2500M words) in a self-supervised manner: During training, 15% of the tokens are randomly masked and the model has to predict the missing tokens from the remaining tokens as context. In this Masked Language Modeling (MLM) the model has access to the whole input sequence (from left to right and from right to left) except for the mentioned missing tokens. Another training objective was Next Sentence Prediction (NSP): Given two sentences, BERT had to classify the second sentence if it was following the first semantically.

To use the pre-trained BERT for downstream task, they fine-tuned on specific (much smaller) datasets. For classification tasks, the final hidden state of a special classification token put at the beginning of every sentence (also during pre-training) is used and then processed by a single added output layer that was not present during pre-training. For other tasks like sequence tagging the final hidden states of all tokens are fed into an output layer. The authors saw positive transfer learning from this pre-training, scaling the model size improving model performance and BERT advancing the state-of-the-art (SOTA) for eleven NLP tasks.

GPT In the same year, the first version of the Generative Pre-Training (GPT) model was published [58]. It consists of multiple Transformer-decoder-blocks without the cross-attention to the (missing) encoder. The training procedure was similar to BERT: The authors pre-trained the model with the language modeling objective (Equation 2.1) self-supervised on unlabeled data (BooksCorpus). The model had to learn which token out of the vocabulary is the most probable next token given all previous ones (within the context window of 512 tokens).

For each specific task they made minor modifications to the architecture (adding a linear output layer and aggregating the result(s)) and trained GPT on task-specific data. They saw an improvement over RNNs like long short-term memory networks in performance but also in the ability to process longer sequences. They found that including the language model objective as an auxiliary objective during fine-tuning helps the model to generalize (especially for larger datasets). Their method achieved a new SOTA for nine out of twelve tested datasets including natural language inference (NLI), QA, semantic similarity, and text classification.

With GPT-2 the authors showed that SOTA performance on NLP tasks can not only be achieved by fine-tuning pre-trained language models with very small architectural extensions but rather by using pre-trained language models directly in a zero-shot setting [59]. Zero-shot learning is the technique using a deep learning model for a task it was not explicitly trained on. The only information about the task the model gets is an instruction for the task in natural language [12]. They show that performance scales with model size (up to 1.5B parameters) when training on scraped internet data which enables the model to learn different NLP tasks through the language modeling objective. Their dataset WebText consists of 8M documents resulting in 40GB of text from websites that were favorably linked by human users on the social media platform reddit.

On seven out of eight language modeling benchmarks their models (sometimes not even the biggest version) achieve SOTA performance. They investigated the overlap between their training data and the evaluation data and it seemed to be not significantly higher than the overlap within the task specific training and evaluation dataset. On more practical tasks like summarization GPT-2 performs slightly better (with a task hint) than taking three random sentences from the input and falls far behind task-specific models. They did not fine-tune their model but a follow-up work [82] uses reinforcement learning from human preference (roughly similar idea to paragraph 2.2.2) to fine-tune GPT-2 774M on stylistic continuation and summarization which showed significant improvement over zero-shot but still worse performance than the task-specific baselines.

2.2. Large Language Models

The foundation for most current Large Language Models (LLMs) like the Gemini [3], GPT [12, 49], LLaMA [70, 71], Mistral [33, 34] and PaLM [4, 16] families is the same as described previously (subsection 2.1.3). They are pre-trained on Internet-scale data and have a Transformer decoder-only architecture with billions of parameters enabling them to perform tasks they were not explicitly trained on.

2.2.1. Impact of Scaling

Scaling up unsupervised pre-trained Transformer-based models (both decoder-only and full Transformers) to multiple billions of parameters was shown to be possible and that this advanced the SOTA on NLP tasks [60, 61, 65]. Rather theoretical work discovered that the loss of language models scales as a power-law of model and dataset size and the training compute [38]. Inspired by this, the authors of GPT-3 scaled a model similar to GPT-2 up to 175B parameters resulting in strong performance on many NLP benchmarks including some that require reasoning like 3-digit arithmetic [12] enabling them to perform more complex tasks.

As they trained multiple sizes of the same model architecture, they observed large performance gains especially in few-shot settings where the model gets up to hundreds of examples together with its input during inference. The 175B model is also able to generate news articles that humans can only identify in 52% of the cases with 50% being random chance. Yet the authors discover that the language modeling objective may limit the performance of these models on tasks that involve fill-in-the-blank, comparing two pieces of content or re-reading that would benefit from a bidirectional architecture. They also note that the objective weights every token equally and that learning from humans like it was tried out for GPT-2 [82] or later on GPT-3 itself (paragraph 2.2.2) could provide a better training signal and adding modalities like vision could improve understanding of the physical world which was done later on models like GPT-4 [49] and Gemini [3].

2.2.2. Instruction Tuning and RLHF

Even though LLMs offer impressive performance on many NLP tasks, they sometimes make up facts, produce toxic context, or fail to follow instructions [52]. This may be a result of the mismatch between the language modeling objective and the objective of human users that the LLM should "follow the user's instructions helpfully and safely" [52].

Instruction Tuning When given a few examples for a task, LLMs like GPT-3 show impressive NLP capabilities. However, their zero-shot skills are much worse [77]. To improve this, the authors of Finetuned Language Net (FLAN) propose instruction tuning [77]: Fine-tuning a LLM with natural language instructions for twelve task types from 62 NLP datasets. This technique improved the zero-shot performance of their 137B parameter pre-trained LaMDA model [69] such that it outperforms the 175B GPT-3 in zero-shot on 20 out of 25 datasets and even in few-shot on some. The evaluation was done on a set of held-out task types.

Further research on instruction tuning showed that scaling up the number of tasks in the training data (1.8k) and including chain-of-thought (CoT) data dramatically improves performance across several model sizes (from 80M to 540B), different prompting setups (zero-shot, few-shot, CoT) and multiple benchmarks [17]. CoT describes a prompting technique that includes the reasoning steps that lead to the result of the examples in a few-shot setup [76]. The model then mimics these chains of thought which improves the performance on reasoning tasks.

Reinforcement Learning from Human Feedback (RLHF) Reinforcement Learning from Human Feedback (RLHF) tries to further reduce the discrepancy between the language modeling objective and the user's needs [52]: The 1.3B version of the resulting model InstructGPT that is based on GPT-3 is preferred over the full 175B GPT-3 model in human evaluation. The training method of the authors start similar to FLAN: They improve GPT-3 by fine-tuning it with prompt-answer pairs provided by human labelers. This part resembles the supervised fine-tuning (SFT) of FLAN. Following this the authors started a new data collection: The labelers get presented a few sampled model outputs from one prompt which the labelers then rank by preference. These rankings are then used to train a reward model to predict the users' choices. The authors modified a 6B version of GPT-3 with the unembedding layer removed to output a scalar reward value. Now during the reinforcement learning part, the previously trained SFT model (policy) generates an output to a prompt which the reward model rates. Based on this the policy is updated improving the model to follow the preference of users.

2.2.3. Representative LLMs

As we want to experiment on LLMs themselves which includes fine-tuning and modifying them, we utilized available open-source models.

LLaMA Large Language Model Meta AI [1] (LLaMA) is a family of publicly available LLMs ranging from 7 to 70 billion parameters (in version 2) [70, 71]. As other current LLMs [4, 49], LLaMA is a Transformer-based deep learning model trained on next-token prediction / the language modeling objective. Following the work on training-compute-optimal LLMs [30], the authors focus on training smaller models with more data (and more compute) to achieve better inference-compute efficiency.

The architecture is a decoder-only Transformer with some modifications from prior publications: Inspired by GPT-2 [59] and GPT-3 [12] the normalization is done on the input of each sub-layer instead of the output to stabilize training. Like in PaLM [16], they replaced the activation function of the original Transformer ReLU with the SwiGLU activation function. The authors use no absolute positional embeddings in LLaMA, instead they add rotary positional embeddings (RoPE), which were already used for GPT-Neo [11], at each layer.

The used datasets are all publicly available. For the first version of LLaMA, 67% of the training data comes from the CommonCrawl dataset with all non-English data removed. Another 15% are from C4 - a pre-processed CommonCrawl variant. 4.5% each are from Github, Wikipedia (multiple languages) and the book corpora Gutenberg and Books3. Another 2.5% and 2.0% are from the ArXiv preprint server and the question-answer website Stack Exchange respectively. For version 2 the used training data is also publicly available online but the specific mix is not disclosed in the paper [70]. The authors increased the number of tokens to 2T while again up-sampling more reliable data for less hallucinations.

The first version of LLaMA was only pre-trained with the language modeling objective. They also tried a small amount of the advanced instruction tuning described above (paragraph 2.2.2) resulting in a 5.5 percentage point gain on the MMLU benchmark for the 65B instruction-tuned LLaMA-I model surpassing even Flan-PaLM (62B) which is also instruction tuned. For LLaMA 2, the author created Chat-versions which are heavily modified with instruction tuning or supervised fine-tuning (SFT) and RLHF. They started with SFT they already used for LLaMA-I and added high-quality data they collected via annotation vendors. For RLHF, the annotators wrote prompts, get two sampled model responses (different model variants, different temperatures) and then label the responses with their degree of preference which should focus on helpfulness and safety. This data was then used to train the reward model (paragraph 2.2.2).

Vicuna Vicuna is a collection of fine-tuned LLaMA models [15, 81]. For their 13B model, they claim over 90% of GPT-3.5's performance [15] evaluated with their LLM-as-a-judge framework which utilizes a strong LLM (like GPT-4) as an automatic chatbot evaluator [81]. GPT-3.5 [50] is a model related to InstructGPT [52] and serves as a basis for ChatGPT [50]. The fine-tuning data is collected from the ShareGPT website where "users can share their ChatGPT conversations" [81]. They used 125k conversations for training a 7B, 13B and a 33B version based on LLaMA 1. For LLaMa 2, only 7B and 13B versions exist called Vicuna v1.5.[1] The fine-tuning data was 370M tokens long.

On top of these models fine-tuned for better chatbot performance, there are models with longer context windows than the original LLaMA model (version 1: 2k, version 2: 4k

[1] `https://github.com/lm-sys/FastChat/blob/97065ff7caa3ae4ca28c661b7424f7ae4cca539b/docs/vicuna_weights_version.md`

[70]) with up to 16k and 32k tokens using a technique similar to Positional Interpolation developed independently [39, 42].

2.2.4. Efficient Implementations for LLMs

As Transformer models and especially Large Language Models get larger and larger, a need for compute-efficient training (primarily interesting for research groups lacking sufficient compute resources) and inference (primarily interesting for companies providing services) of these models arises. For example, the biggest Transformer model whose size is known has 1.571 trillion parameters [25], high-performing commercial LLMs have a few hundred billion parameters [12, 16], and even publicly available LLMs like LLaMA 2 have versions with up to 70 billion parameters [70].

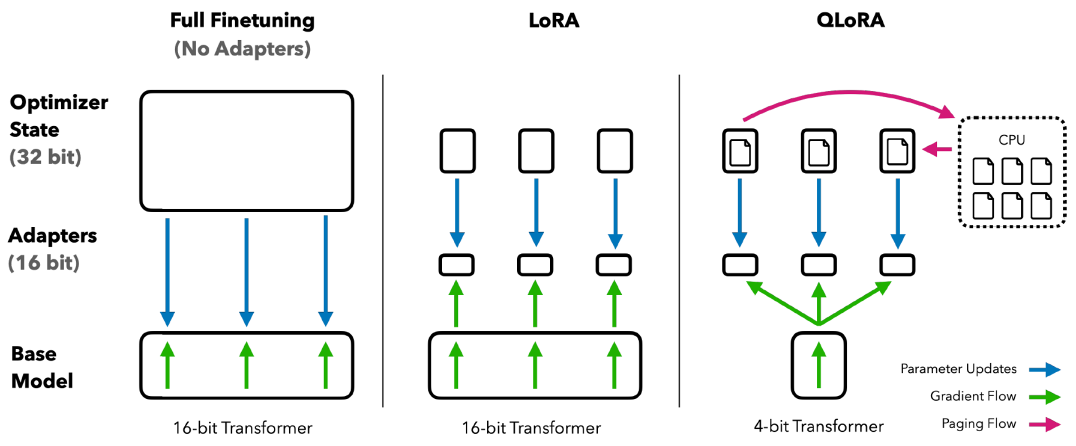

Figure 2.2.: Comparison between full fine-tuning, LoRa fine-tuning and QLoRA fine-tuning [22].

(Q)LoRA Low-rank adaption (LoRA) [32] is a method to reduce the number of parameters changed when fine-tuning LLMs and therefore decreasing the training time and memory requirements. The weight matrices are split into the pre-trained weights $W_0 \in \mathbb{R}^{d \times k}$ which are frozen during training and the adapter weights $\Delta W \in \mathbb{R}^{d \times k} = BA$. When calculating the result of a layer h both weight matrices are used (Equation 2.5). W_0 consists of $B \in \mathbb{R}^{d \times r}$ (random Gaussian initialization) and $A \in \mathbb{R}^{r \times k}$ (initialization with zero) where r is the LoRA rank - the hyperparameter determining how many parameters are changed. The authors only adapt the attention layers and note that training all weight matrices inside the attention block leads to better results than only training a subset of them. The other layers (e.g., the FFN block) were not investigated.

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{2.5}$$

When using LoRA to train [32] GPT-2 Medium (~355M parameters) and Large (~774M parameters) on the E2E NLG Challenge, the LoRA-trained model version achieved the best results almost all the time while modifying less parameters (0.1% of original parameters) than full fine-tuning and any other adaption method tested. All methods were trained for the same number of epochs. For GPT-3 175B the picture is similar: LoRA matches or outperforms full fine-tuning with only ~3% of the parameters trained. With a higher rank which results in the adapter having ~22% of the parameters the adapted model outperforms the fully fine-tuned one in every case. Especially GPT-3 175B shows the practical benefits of LoRA. The training sees a speedup of 25% while the memory requirements are reduced by about 2/3 from 1.2TB to 350GB.

There are several other fundamental advantages of LoRA over other parameter-efficient adaptations: As the parameters trained when using LoRA are added to the pre-trained weights, the total number of parameters of the model stays the same. Even if the number of "normal" adapters is small, as they have to be computed sequentially to the other weights, they can add a noticeable latency during inference [32]. When a LoRA-trained model is deployed its adaptation can easily be swapped by subtracting the adapter weights and adding different adapter weights. In this scenario multiple adapted models can be efficiently stored by having only the pre-trained model and the much smaller adapters saved. A benefit compared to methods like prefix tuning or prompt engineering is that the available sequence length is not reduced due to added tokens in front of the input. The authors also note that LoRA is relatively stable concerning the number of trained parameters of GPT-3 175B, while other methods need more careful hyperparameter tuning [32].

To further reduce the compute and memory requirements of fine-tuning, the authors of Quantized LoRA (QLoRA) [22] propose quantizing the pre-trained model that is frozen during LoRA training to 4-bit data types. When the quantized weights are used for matrix calculations, they get dequanitized to 16-bit types temporarily. To achieve this, they use the 4-bit NormalFloat (NF4) data type which they claim is information theoretically optimal for this application. During the quantization process, floating-point numbers called quantization constants are calculated. To reduce their impact on memory consumption, they quantize these numbers too with a process called Double Quantization resulting in 0.37 bits per parameter less on average. To avoid memory spikes during gradient checkpointing that results in out-of-memory issues they use the NVIDIA unified memory feature and allocate paged memory for the optimizer states that then get swapped into the main (CPU) memory when the GPU memory is full and gets swapped back when needed. These improvements reduce the average memory requirements of LLaMA 65B from >780GB to <48GB enabling fine-tuning on a single professional GPU enabling teams with less resources to fine-tune models which is essential to turn pre-trained LLMs into e.g., useful chatbots. Standard LoRA only adapts the attention weights, replicating this with QLoRA leads to worse performance than full fine-tuning. The author found that the number of LoRA adapters is the most critical hyperparameter (opposed to the total number of adapted parameters). Therefore, they also adapt all linear layer blocks. To demonstrate QLoRA they fine-tune LLaMA from sizes 7B to 65B on two instruction following datasets (Alpaca and FLAN v2) with full fine-tuning and QLoRA. When comparing their 5-shot MMLU accuracy the mean score for the QLoRA-trained models is 0.1 percentage points better than full fine-tuning.

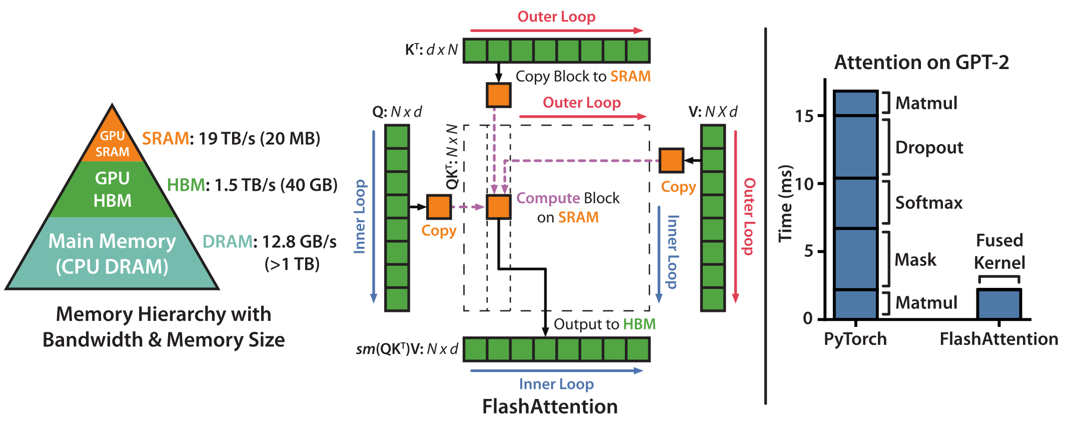

Figure 2.3.: Memory hierarchy of GPUs, the FlashAttention algorithm and the runtime behavior of “normal” attention vs. FlashAttention [20].

FlashAttention Despite all their strengths, Transformers have the weakness of scaling quadratically in sequence length. In general, models that use approximate attention to reduce this effect have a reduced quality or offer no wall-clock speed-up [20]. FlashAttention [20] proposes an IO-aware variant of attention: Moving the data in a non-naive way inside the memory hierarchy. For training and running inference of LLMs like GPT [12] or LLaMA [70, 71] high-performance GPUs are used. They have a memory hierarchy similar to the traditional one with CPU cache made of static random-access memory (SRAM) and main memory made of dynamic random-access memory (DRAM). GPUs feature very fast but small SRAM cache (megabytes range) and slower but still fast much bigger High Bandwidth Memory (HBM) memory (tens of gigabytes) (Figure 2.3). When running the attention algorithm (Equation 2.2) the data in form of vectors has to be moved from the memory to the cache, used for calculations and moved back to the memory. During standard attention, the product of the query Q and keys K and the result of the softmax function produce intermediate matrices taking up $O(N^2)$ (N is the sequence length) not directly needed for the final results but for the backward pass. The authors of FlashAttention propose the idea of splitting the inputs Q and K into blocks (tiling) and applying softmax per block rather than the whole matrix while storing some statistic values for recomputing the attention matrices during the backward pass (Figure 2.3). Now only $O(N)$ additional memory is needed. Instead of performing all these operations consecutively with every operation loading from memory to cache and storing from cache to memory, the authors fuse them into one kernel (one load / execute / store GPU operation). With FlashAttention GPT-2 can be trained 1.7 times faster than the then fastest implementation. FlashAttention also eases longer context windows: Even if the context length of GPT-2 is increased by four times the training still runs 30% faster and achieves a 0.7 better perplexity in language modeling on the OpenWebText dataset.

FlashAttention 2 [19] improves this further by addressing some weaknesses the authors found during their analysis of the first version: FlashAttention only achieves around 25-40% of the maximum FLOPs/s that the hardware is theoretically capable of. The authors therefore reduce the (rather small) number of non-matmul FLOPs as GPUs have special units for matrix multiplications that allow an up 16× higher throughput. As GPUs are highly parallel, a program is executed by a high number of threads. These threads are grouped into "warps" consisting of 32 threads each. Multiple warps make up a thread block. The authors discovered a suboptimal work partitioning between these thread blocks and warps which results in unused hardware resources or unnecessary communication between these thread groups. They found that the existing parallelization over the batch size and the number of heads can be a problem for long sequences as in that case the batch size and possibly the number of heads is lower. Therefore, FlashAttention 2 parallelizes also over the sequence length. Regarding the work partitioning between warps, they discovered that the splitting of $QK^\intercal$ leads to unnecessary synchronization, so they only split Q and make K accessible to all warps. These measures and some other minor improvements speed up the attention calculation by 1.3× to 1.5× (depending on sequence length) for the forward pass and by 2× for the backward pass. The end-to-end training throughput for GPT-style models with sizes 1.3B and 2.7B parameters is around 1.3× higher for sequence lengths of 2k and 8k.

2.3. Question Answering Task

Question answering is an NLP task that focuses on answering questions based on related context or knowledge [75]. This task can be grouped into different partially overlapping categories [53]: The question type can be multiple choice (MC) where one or multiple correct answers have to be selected resembling a classification problem. The opposite of this is conversational QA where the system has to freely generate the answer. The type of answer can also be categorized: A factoid answer usually only contains a few words while definition-based QA demands definitions as answers from a knowledge base. There are hybrid forms of this where the QA system has to interpret the question carefully to answer with the correct type. The type of context can also differ as it may be present as knowledge (typically organized as a knowledge graph) or as harder to manage raw text. Modern QA system mostly use deep learning-based models like (fine-tuned) BERT- or GPT-style models. Earlier (classical) machine learning systems use methods [53] like support vector machines [35], decision trees or naive Bayes in a classification setup where the features are pre-defined. It is also possible to design rule-based QA systems with surrogate tasks like semantic class tagging and entity recognition that assign scores given the question and a context piece. The answer is the sentence in the input with the highest score.

2.3.1. Challenges and Approaches

Scientific papers present great challenges as context for QA when using LLMs for multiple reasons: They are often thousands of tokens long exceeding the context window of 4k for models like LLaMA 2. Also, they consist of long unstructured (except sectioning etc.) raw text making it hard to determine which part is important to answer the question. The answer type is also not clear as the question could be about explaining some concepts presented in the paper, simple facts or even yes or no questions, or the question could be unanswerable.

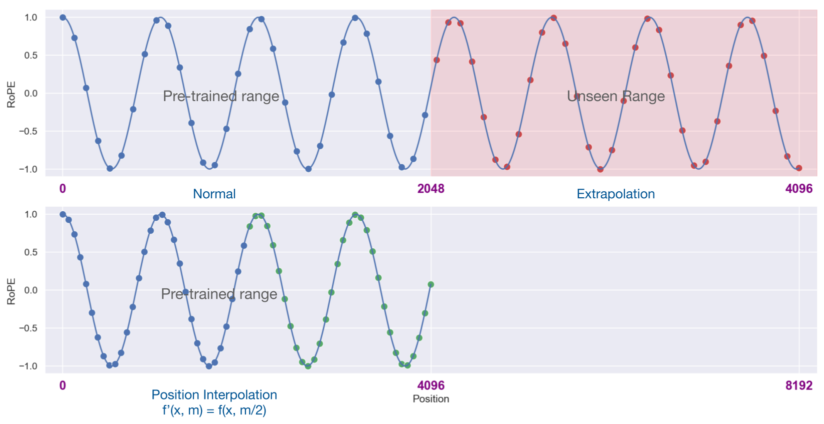

Figure 2.4.: Positional encoding if the RoPe method is extrapolated vs. interpolated [13].

The unstructuredness and length of the context is especially problematic even for long-context LLMs as researchers found [45]: For multi-document QA where the LLM has to select the relevant context part from multiple options, the performance curve has a U-shape with respect to the position of the documents as the ones at the beginning and the end are better retrieved than those in the middle. Some models like GPT-3.5 and MPT [68] even have the same problem [45] on a simpler synthetic task that involves the retrieval of a value from a JSON object given a key.

As most LLMs are Transformer-based, their time complexity and memory consumption scales quadratically in sequence length because of the attention algorithm which makes both the training and inference with long context a problem. If one does not want to use approximate attention as this may introduce inductive bias or reduce performance in general [78], there are techniques to still handle long context.

Positional Interpolation The context length an LLM can handle is determined by the length of the samples of the training data during its pre-training. Fine-tuning extends the context window only very slowly in terms of training steps [13]. Although there are extrapolation techniques that enable inference on longer context than the model was trained on (Figure 2.4), many LLMs use positional encodings (e.g., Rotary Position Embedding (RoPE) like LLaMA [70, 71]) that show weak performance during extrapolation [13]. The authors of Positional Interpolation (PI) propose the idea of stretching the original context window (L) to the new maximum length L' by down-scaling the position indices that are the input to the positional encoding function (Figure 2.4). RoPE uses a vector-valued complex function $f(x, m)$ where x is the embedding vector and m the position index. For PI, m is replaced by $\frac{mL}{L'}$ so the maximum distance between tokens stays the same.

The authors trained all sizes of LLaMA 1 (context window: 2k) on the next token prediction objective for only 1000 steps on longer context and showed for fine-tuned models (no PI, 8k) with up to 33B parameters that they have a higher perplexity which rises as the evaluation context window increases. On the other hand, the perplexity of models with PI decreases or only slightly increases with longer evaluation sequences.

For models with up to 13B parameters, they show similar behavior on context windows of up to 32k tokens. They also note that models with PI have only slight performance degradation within their original context window of 2k tokens.

Retrieval-augmented Generation Another idea is to compress the given context to fit inside the context window by augmenting the LLMs with a retrieval method [41, 78] that extracts only relevant parts from the context. The retrieval method can be included at pre-training [28], fine-tuning [41] or inference [78]. These methods can also enable the model overall to run faster as the sequence processed by the LLMs is shorter.

A retriever model gets a long context and a short query as input and determines the importance of each part of the context [78]. For example, Dragon [44] a state-of-the-art retriever both in supervised and zero-shot usage with a dual-Transformer-encoder architecture takes the query and each part of the context one by one as inputs. The resulting vectors can then be used to compute the most relevant passages by calculating the dot product between the query vector and every passage vector. Some works suggest that retrieval-augmentation enables models with shorter context length (e.g., 4k) to match models with longer context length (e.g., 16k) [78] while others observe no general performance improvement [6]. This may depend on model size as bigger models seems to benefit more from retrieval-augmentation.

2.3.2. Datasets

To evaluate LLMs on question answering on scientific papers, we need data consisting of papers and related questions and answers.

Datasets on Scientific Papers Datasets that cover the topic of scientific papers focus on various aspects. Many [14, 37, 56, 43, 74, 79] focus on the review process which yields different artifacts: The review itself with comments and possibly a rating by the reviewer. For some reviews exist rebuttal letters by the paper author to react to the review. A meta-reviewer summarizes the other reviews into a meta-review together with an acceptance score. These artifacts enable different tasks: (Meta-) Review Generation [43, 74], acceptance prediction / paper rating [37, 79], Argument Pair Extraction from reviews and corresponding rebuttals [14], and Multi-document Summarization on reviews [56].

Qasper As the primary focus of this thesis is the answering of specific questions on scientific papers with LLMs, we need a dataset different to the ones mentioned above. The Question Answering over Scientific Research Papers (Qasper) dataset [21] is a collection of 1585 NLP papers with 5049 questions (split into train / validation / test: 2593 / 1005 / 1451). Each of these questions was formulated by an NLP practitioner who only read the title and abstract of the corresponding paper. The answers were then answered by other NLP practitioners who also selected the paragraphs, figures or tables ("evidence") in the paper that are relevant to answer the question (which may be none). There are four types of questions / answers in this dataset: Extractive (questions can be answered by copying chunks of the relevant paragraph), abstractive (free text answers that are not literally in the paper), yes/no questions and questions that are unanswerable. Like for the SQuAD dataset, the authors chose a span-level F_1 score as their metrics. If there are multiple reference answers, the maximum of the F_1 score will be used. They also provide a baseline model: The challenges of long context and different answer types (arbitrarily long text or simply yes/no / unanswerable) excluded many already existing models like BERT-style models or decoder-only LLMs. The authors chose the Longformer 16k model: The Longformer-Encoder-Decoder (LED) [7] has a similar architecture to the original Transformer with the difference of a local attention window and a global attention at pre-specificed points of interest which enables a linear time complexity in sequence length. With a context window of 16k, Longformer can encode 99% of the dataset's papers.

Datasets focused on Long Context In recent times a lot of new LLMs have been published, many of which offer longer context windows (>4k). To make comparison easier, multiple benchmarks sets have been created to test their abilities across different task types.

ZeroSCROLLS [64] is a benchmark focused on long text understanding in a zero-shot setting. The included task types are summarization, question answering and aggregation. Aggregation tasks involve "contextualizing and aggregating information from different parts of the input" [64] that goes beyond simple summarization or QA. For example, one aggregation task in ZeroSCROLLS is to determine the percentage of positive reviews given hotel reviews without ratings. A 500-item subset of Qasper is part of the question answering section. The authors evaluated open-source models like Flan-T5-xxl [60] and Flan-UL2 [67] (both encoder-decoder, instruction tuned with FLAN method paragraph 2.2.2), closed models like GPT-3 [12, 50] (different variants), GPT-4 [49], and Claude [55] and a task-specific model CoLT5-xl [2]. ZeroSCROLLS itself does not contain any training data. Their results show that GPT-4 is the best model on average, yet on Qasper it does not always follow the answer format allowing the encoder-decoder models to achieve slightly higher F_1 scores.

A similar benchmark called LongBench also includes Qasper [6] but only 200 or 224 (LongBench-E) examples respectively. Opposed to ZeroSCROLLS it is bilingual (English and Chinese) and incorporates more task types: Single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion. The authors evaluated a different set of LLMs: Closed ones like GPT-3.5-Turbo-16k and open-source ones like Llama2-7B-chat-4k, Vicuna-v1.5-7B-16k, and LongChat-v1.5-7B-32k. They find that GPT-3.5 outperforms the open-source models but still struggles on longer contexts and they report substantial improvements on long context understanding on models whose position embeddings were scaled up with techniques like Positional Interpolation (paragraph 2.3.1) and fine-tuned on longer sequences.

3. Approach

To improve the performance of the general-purpose LLMs, we apply different techniques ranging from prompting to fine-tuning. The first kind of techniques we introduce are generally used to positively influence the behavior of LLMs and therefore not specific to question answering on scientific papers. To further optimize the model for our use case, we also employ task-specific techniques.

3.1. Task-agnostic Techniques

As the name “Large Language Models” suggests, LLMs have a high number of parameters spanning from a few billions to trillions, resulting in big file sizes and also in high memory consumption when they are fully loaded into memory to run inference. This is why in production, LLMs are mostly only used in inference and not fine-tuned for specific use cases as training requires even more compute and memory. Another approach to influence the behavior of an LLM is to prompt it with instructions that are optimal for a specific task.[1] The process of finding such optimal prompts is often called “prompt engineering”.

Zero-shot Prompting As the long context posed by scientific papers is one of our main problems, we can not use few-shot prompting as this would require including examples in the input which would only increase the challenge of long context. One example would consist of a whole paper, a question, the instruction, and the corresponding answer. Therefore, we have to resort to zero-shot prompting which only includes the instruction for the model as a kind of learning signal. This is the easiest and computational cheapest option as no parameters are changed and the model is used like during “normal” inference with the difference that the instruction to solve the task is carefully selected for that specific task. We list all our used prompts in section A.1.

Supervised Fine-tuning By combining the previously mentioned efficient methods (subsection 2.2.4) of implementing and training Transformer-based models, we are able to fine-tune a small LLM on long context. We replace the standard attention algorithm with FlashAttention 2 which is faster and more memory-efficient in sequence length which is especially important for the long context in our case. Additionally, we use QLoRA for fine-tuning the model. LoRA only changes a fraction of the parameters of the model which reduces the compute and memory consumed during the backward pass. QLoRA quantizes the model weights that are used during the forward pass reducing the memory requirements even further.

[1] `https://huggingface.co/docs/transformers/main/en/tasks/prompting`

As the parameters learned via (Q)LoRA are added ($W + \Delta W$) to the existing parameters we do not increase the number of parameters. This means after fine-tuning the inference time stays the same. As the addition of the adapters is reversible by subtracting the adapter weights from the original model weights, we can still use the model in a general-purpose setting by switching to other task-specific adapters (e.g., writing abstracts / reviews for scientific papers). Alternatively, the adapter weights can be added during inference (increasing latency) which may be more useful when switching tasks frequently. As only a fraction of the parameters has to be changed by (Q)LoRA to be effective, multiple different adapters can be stored alongside the model without consuming unacceptable amounts of storage.

3.2. Task-specific Approaches

On top of these two more general techniques, we employ more task-specific variations to increase performance and explore the limitations and potentials of the models we looked at.

Evidence-only Prompt As the dataset we use also contains the extracted relevant paragraphs needed to answer the question ("evidence") for each question, we want to find out how our investigated models perform if we provide them with the evidence only – both during inference and training. This should give use an idea of the upper limits of the performance of the models as this task should be easier as the model has to process fewer tokens. Another benefit could be reduced training compute / time. Additionally, we think that a comparison between these fine-tuned models and those that received the full paper during training should indicate how much our fine-tuning improves our goal of long-context understanding and how much it just improves instruction following.

Two-step Prompt Chain-of-thought prompting [76] showed that splitting a task into subtasks can help LLMs to solve them. Therefore, we split the question answering into two tasks: First the model has to find the evidence – all relevant paragraphs to answer the question. After that we prompt it to answer the question based on the extracted paragraphs in the previous step. As the dataset includes the evidence for every question, we can evaluate the performance of the evidence extraction and also train the model on this auxiliary task specifically. As unanswerable questions and some yes/no questions are annotated with no evidence, we have to use a placeholder during inference (included in the prompt) and training. We assume that generating this placeholder provides more information and is easier for the models than generating nothing.

On top of a possible performance increase, extracting the relevant paragraphs is a useful task on its own: It could be useful for the user the see the context of the answer inside the paper to gain further insight. Also, this could improve interpretability as the user can see through this intermediate step how the system arrived at an answer. This could be especially useful if the model fails and produces a wrong answer.

There are also some downsides: We have to run inference twice as this approach requires the model to generate its input for the second step. As the context can be very long, this may result in an unacceptable latency increase between issuing the question to the QA system and getting the answer. Also, as the model generates its own input (apart from the second prompt), this approach may lead to cascading errors.

Similar approaches were investigated for science QA on short context [46, 73, 80], for (Chinese) multi-document QA [29], and in the original publication of the Qasper dataset [21] (but the model was no current LLM).

During training, the most common evidence presented to the model is the placeholder we use for no evidence as almost all other evidence is from different papers and different questions and as a result is a diverse collection of strings. We therefore include a prefix in the training data and a hint in the prompt that every non-empty extracted evidence starts with this prefix. We argue that this helps the model to avoid resorting to generating the "easiest" evidence which is none or the placeholder. Attention Strengthening Question Answering [29] includes a similar technique which inserts hints to predict the indices of the most relevant document in multi-document QA. We also adopt their approach of placing the question before and after the context to achieve a "contextual-aware representation". To further reduce the number of generated empty evidence, we lower the number of examples in our training data where no evidence should be found to push the model into generating non-empty evidence more frequently.

Split Prompt As we discovered that our tested models with extended context windows still show a performance degradation on context exceeding the original context window, we split the context into parts that fit inside this original context window. With a naive rule-based approach, we fuse the resulting answers per split into one final answer. For the two-step prompt, we remove all placeholder strings for empty evidence if the model generated non-empty evidence for at least one part of the context. If there are multiple instances of non-empty evidence, we simply concatenate them together in the order of the split they are from. We also tried this approach with the one-step prompt. Here, we filter out the answer that the question is unanswerable given the split as context. Otherwise, we operate in the same way as for the two-step prompt.

Modify Attention Algorithm During manual investigation of the internal values of the attention function with the Inseq[2] [62] tool to interpret sequence generation models, we found that most of the attention weights are very small numbers. Our hypothesis is that when we mask a portion of these tokens that the model attributes very small attention weights too, this now freed probability mass of the attention matrix can move to the more important tokens, highlighting them and potentially leading to better answers. Our masking works by finding the indices of the k lowest values over the sequence length inside the attention matrix. We set all these attention weights to the lowest possible value of negative infinity. The softmax function after that in the standard attention algorithms sets these values to zero.

[2]`https://inseq.org/en/latest/index.html`

4. Experiments and Results

After describing the setup we used to run our experiments, we will list all experiments we ran and discuss their results.

4.1. Experimental Setup

In the following, we will describe our experimental setup to enable easy reproducibility of our experiments and results. This includes a description of the used dataset, our preprocessing of it, the utilized hard- and software, the hyperparameters we used during inference and training, and the models we experimented with. When we refer to numbers in tokens and use unit prefixes like k we mean the binary prefixes.[1] This means that e.g., 4k tokens mean 4,096 tokens.

4.1.1. Dataset

The Qasper dataset (paragraph 2.3.2) we used to evaluate and train the considered models consists of a total of 1,585 NLP papers with 5,049 questions on these papers. The authors provide the following dataset splits: train (2,593 questions), validation (1,005 questions), and test (1,451 questions). The four types of questions appear in different frequencies (Table 4.1) and the authors evaluated the performance of their model for each question type individually. The dataset website[2] provides an official evaluation script.

Question type	Frequency
Extractive	51.8%
Abstractive	24.2%
Yes/No	13.9%
Unanswerable	10.2%

Evidence type	Frequency
Text	81.6%
Table/Figure	11.6%
None	12.8%

Table 4.1.: Qasper dataset statistics [21]: question / evidence types; the percentages for the evidence types add to over 100% because answers can include multiple evidence types.

As this thesis covers also how well LLMs handle long context, we take a look at the models' performance per paper length (Table 4.2) and absolute position of the related paragraphs ("evidence"). Each question in the dataset is annotated with an evidence field which contains the paragraphs or tables / figures that are needed to answer the question.

[1] `https://en.wikipedia.org/wiki/Binary_prefix`

[2] `https://allenai.org/data/qasper`

For 12.8% questions, this field is empty. This is predominantly the case for unanswerable questions but also for some yes/no (boolean) questions. It is important to note that paper length can be determined for unseen papers while evidence position is unclear.

For the final analysis, we use a subset of the Qasper test split that is part of the Zero-SCROLLS (ZC) benchmark. We saw a similar statistic for this subset as for the (custom) splits we used during development and final analysis. We therefore assume that the ZC subset of Qasper will be representative for the performance of our approaches.

Models	dev-short		dev		test		ZC	
Questions / %	990	100	1,005	100	1,451	100	500	100
Paper length								
0k – 4k	333	34	333	33	511	35	149	30
4k – 8k	593	60	593	59	802	55	312	64
8k –	64	6	79	8	138	10	39	8
Absolute evidence position								
0k – 4k	794	80	799	80	1182	81	405	81
4k – 8k	173	17	180	18	263	18	91	18
8k –	6	1	11	1	18	1	7	1
No evidence	77	8	78	8	99	7	37	7

Table 4.2.: Qasper dataset statistics we created for our research questions: paper length and absolute evidence position; the numbers for absolute evidence position exceed the total number of questions because the evidence for a question can be from multiple paragraphs.

4.1.2. Data Preprocessing

During the development, we used the development / validation split of Qasper to evaluate our approaches and choose the best one. Five of the papers lead to out-of-memory errors during inference. We therefore exclude these five papers from our results and call the resulting split "dev-short". As these five only account for around 1.8% of the 281 papers in the dev split, we assume that this does not skew our view of the quality of the models. Also, the distribution of the length / position bins is not changed much (Table 4.2). All removed papers have more than 16k tokens.

As our tested models have text as their only modality, it cannot process the figures and tables provided with the dataset. We therefore remove all questions from the training data that mention figures or tables in their evidence field. This type of evidence starts with the string `FLOAT SELECTED` and can therefore be reliably removed.

As training data, we use the training split of the Qasper dataset. As input, we use a prompt template from the LongBench benchmark dataset [6] where the paper text and the question are inserted the same way as for zero-shot prompting. The target is the answer from the dataset. Many questions are annotated with multiple possible answers. We assume that most of them are equally good as training data as in general multiple answers to a question can be correct. In some cases, they clearly heavily disagree with each other e.g., one possible answer is "Unanswerable" and the other is "Yes" or "No". We remove these cases. We also have to limit the training data to texts with a maximum of 8,192 tokens as longer inputs cause out-of-memory errors even with techniques to increase memory efficiency like QLoRA and FlashAttention.

4.1.3. Hard- and Software

For evaluation and training of the tested models we need high-performance GPUs. Therefore, we use the bwUniCluster 2.0[3] for our experiments. Depending on availability, we use the NVIDIA A100 with 80 GB of accelerator memory or the NVIDIA H100 with 94 GB. The bwUniCluster 2.0 allows the use of NVIDIA Enroot[4] which enables running Docker[5] containers on the computing cluster. We use the PyTorch container[6] by NVIDIA to train the models in our experiments. FlashAttention is only implemented per GPU type at the moment and comes pre-installed with this container.

We run all our experiments (inference and training) with the FastChat[7] [81] framework which is an open-source platform for "training, serving, and evaluating large language model based chatbots". It is developed by the Large Model Systems Organization (LMSYS Org).[8] The LMSYS Org also operates the LMSYS Chatbot Arena[9] [81] which tries to compare the performance of current LLMs against each other in a chatbot setting. FastChat provides code to easily run models, feed them with input data, and store their answers. Besides regular fine-tuning it also provides a (Q)LoRA implementation that can utilize FlashAttention. This script is run with the DeepSpeed[10] library.

4.1.4. Hyperparameters

All following stated hyperparameters are the same on all experiments if not stated differently per experiment.

During inference, we run the models with a temperature of 0.0 which equates to greedy decoding.[11] FastChat code also uses a temperature of 0.0 for tasks like extraction and reasoning.[12] This fits our requirements as we want the most accurate and truthful answer. Also, we saw a degradation in performance when raising the temperature. We let the models generate up to 1,024 tokens.

Our training configuration is the same as the example from FastChat: We use a LoRA rank r of 8 and a LoRA Alpha of 16. Rank $r = 8$ results in 4,194,304 trainable parameters out of 6,742,609,920 for LLaMA 2 7B based models. The dropout is 0.05 and we apply no

[3]https://wiki.bwhpc.de/e/Main_Page
[4]https://github.com/NVIDIA/enroot
[5]https://docs.docker.com/
[6]https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch
[7]https://github.com/lm-sys/FastChat
[8]https://lmsys.org/
[9]https://chat.lmsys.org/
[10]https://github.com/microsoft/DeepSpeed
[11]https://huggingface.co/blog/how-to-generate
[12]https://github.com/lm-sys/FastChat/blob/085c2c37dca426059f023e2a080c45717c742fd1/fastchat/llm_judge/common.py

weight decay. The learning rate is initialized with 2e-5 with a warm-up ratio of 0.03 and a cosine learning rate scheduling. We do no extensive hyperparameter search because of time constraints regarding compute and because the authors of QLoRA already noted that the most important "hyperparameter" is the location of the adapted parameters inside the model. We train each model for 5 epochs on the training split after our preprocessing. We chose this duration as it could be done within a few hours on a single GPU, and we saw performance saturation within this training duration.

4.1.5. Models

We use three models with different context window lengths in our experiments. The creators of FastChat (LMSYS Org) provide the Vicuna family (paragraph 2.2.3) of LLMs. We only test the smallest available models with around 7 billion parameters for compute and memory efficient experiments and as this is the only model size that has a LongChat version. This version has a context window of 32k tokens. Vicuna 7B-4k has the same as LLaMA 2 (4k) and Vicuna 7B-16k's was extended to 16k. We use the models of version v1.5 which indicates that they are based on LLaMA 2 instead of LLaMA 1 like the previous versions. We omit the parameter count in the following from the models' names as they are the same of every model we tested.

4.2. Evaluation Metrics

We use different metrics to evaluate our approaches regarding our specific focus but also to ensure comparability to other work.

F_1 As the paper proposing Qasper and work using Qasper as a benchmark, we use the F_1 score to evaluate our models. The F_1 score is the harmonic mean of precision and recall:

$$F_1 = 2\,\frac{\text{precision} + \text{recall}}{\text{precision} \cdot \text{recall}} \tag{4.1}$$

While we use the F_1 score for all question types, the example of unanswerable questions highlights its usefulness: It tells us how many unanswerable questions the model recognizes as unanswerable (recall) and how many of the claimed unanswerable questions are actually unanswerable (precision).

For Qasper, the official evaluation script calculates scores on a span-level. When there are multiple reference answers for a question, the script uses the maximum F_1 score from all possible ones. We also use this script to see how our approaches perform for the different question types. The script and our tables refer to yes/no questions as "boolean" and to unanswerable questions as "none".

4.2.1. Analysis by Context Length / Position

Additionally to the F_1 score used by the official evaluation script from the Qasper dataset, we evaluate the models by splitting the evaluation data into (partially) overlapping groups. We want to evaluate per paper length and absolute evidence position. During our experiments, we also investigated the relative position. However, we saw no U-shape (in terms of performance per evidence position from paper beginning to end) of the performance. This corresponds to prior work [45] which found this strong primary and recency bias only in large (>7B) models. We measure length / distance in tokens as this enables a comparison to the (declared) context window lengths of the models we investigated.

To bin the evaluation sets, we tokenize the whole paper texts from the JSON files that the dataset website provides. We used the tokenizer from Vicuna-16k, but the number of tokens should be the same for the other tokenizers. For the evidence positions, we determine the position of the first token of the evidence paragraphs inside the paper text. We assume this is representative as most evidence is at most a few sentences long. If there are multiple evidence paragraphs that are not continuous in the dataset, we bin potentially one paper multiple times.

Paper Length As the context windows of some of tested models were extended beyond their initial length, we want to find out if this enables the models to process longer context as well as context within the original context windows or if the performance differs per paper length. Here, we bin per paper as the length is the same for all associated questions. We count the number of tokens after tokenizing the concatenated text fields from the JSON files of a paper.

Evidence Position It is also important to find out if the position of the relevant information (evidence) inside the paper or the total length of the text does affect performance more. We will take a look at the absolute toke position of the evidence. For “Unanswerable” and some yes/no questions there is no evidence in the dataset. We put these questions into a separate bin (“No evidence”). A model tasked to extract the evidence should output no paragraphs here. Instead, they should generate the string “No relevant paragraphs found” which we include in the prompts and filter out of the answers before calculating the F_1 score. In contrast to the length binning, we group the evaluation data per question as the evidence positions differ in general per question and not per paper.

4.3. Results and Discussion

We start our experiments with all available small (7B parameters) models from LMSYS Org with varying context window lengths: Vicuna-4k, Vicuna-16k, and LongChat-32k.

4.3.1. Zero-shot Prompt

First, we run all tested models with a zero-shot prompt (Table 4.3). We use the template from LongBench [6] (subsection A.1.2). We saw similar performance for the models with extended context windows while the one with the original window of 4k tokens performed a lot worse. While LongChat is able to answer the “normal” questions better, it seems to be unable to handle unanswerable questions. Also, its ability to answer yes/no questions is more limited than Vicuna-16k. Qualitative analysis showed that LongChat almost never outputs “Unanswerable” and even if it does, the answer is a whole sentence which ignores the instruction in the prompt (examples: subsection A.2.1). We assume that this difference between Vicuna-16k and LongChat-32k is a result of slightly different fine-tuning resulting in worse instruction following for LongChat.

Models	Vicuna-4k	Vicuna-16k	LongChat-32k
Answer F_1	8.82	24.79	24.19
Answer F_1 by type			
extractive	6.55	22.07	26.51
abstractive	4.81	15.98	20.78
boolean	25.16	59.66	36.79
none	13.28	19.75	0.04

Table 4.3.: Models we tested, dev-short set, **LongBench prompt** [6], zero-shot.

Our deeper analysis (Table 4.4) shows that the three models perform similar for papers that are 4k tokens long or shorter. After this threshold, the “normal” model is not able to perform any kind of position extrapolation. The other two models show similar performance while the 32k version is able to better answer questions whose relevant paragraphs are at higher token positions. The overall performance differs not that much because LongChat struggles with questions that require no evidence (most of them are unanswerable). We assume that the lower F_1 score of LongChat on papers with more than 8k tokens is a result of this weakness and not a general property. We conclude from this experiments that Positional Interpolation (or similar techniques) improves the long-context understanding of LLMs in the domain of scientific papers. But we also see that their performance drops with longer papers and especially questions whose evidence is outside of the original context window are harder. At least this is the case for our simple zero-shot prompt.

Models	Bin count	Vicuna-4k	Vicuna-16k	LongChat-32k
Answer F_1 per paper length				
0k – 4k	333	25.53	27.20	25.47
4k – 8k	593	0.40	24.01	24.08
8k –	64	0.00	19.55	18.51
Answer F_1 per absolute evidence position				
0k – 4k	794	9.79	25.82	26.73
4k – 8k	173	0.38	18.02	23.35
8k –	6	0.00	3.78	15.06
No evidence	77	11.80	23.38	1.06

Table 4.4.: Analysis of the models we tested, dev-short set, **LongBench prompt** [6], zero-shot.

4.3.2. QLoRA Fine-tuned

As especially LongChat in our opinion showed insufficient instruction following, we now want to see how much fine-tuning can increase the performance of the tested models.

LongChat-32k	Zero-shot	1 epoch	3 epochs	5 epochs
Answer F_1	24.19	41.13	44.56	47.02
Answer F_1 by type				
extractive	26.51	41.80	45.18	48.21
abstractive	20.78	12.59	16.59	20.10
boolean	36.79	70.49	80.33	76.47
none	0.04	69.57	66.67	68.54

Table 4.5.: LongChat-32k, dev-short set, **LongBench prompt** [6], fine-tuned with QLoRA.

Vicuna-16k	Zero-shot	1 epoch	3 epochs	5 epochs
Answer F_1	24.79	40.05	43.84	45.23
Answer F_1 by type				
extractive	22.07	38.61	42.26	44.86
abstractive	15.98	11.38	14.57	15.49
boolean	59.66	72.36	78.86	76.67
none	19.75	79.17	78.72	80.43

Table 4.6.: Vicuna-16k, dev-short set, **LongBench prompt** [6], fine-tuned with QLoRA.

The impact of QLoRA fine-tuning on LongChat-32k (Table 4.5) and Vicuna-16k (Table 4.6) is fairly similar: Extractive, boolean and unanswerable questions get a lot better after just one epoch and except for the unanswerable questions they improve with longer training. We assume that the F_1 scores for unanswerable questions reach the highest scores possible with this model size and pre-training and fine-tuning procedure. Here, the model has to do a trade-off between generating answers with more information (extractive, abstractive) or classify the question as unanswerable. For both models, the answers to abstractive questions see an inital quality degradation and only converge back to their inital level late in training. We think that this is a result of the training data forcing the model to fit to the answer style for around 75% of the questions in Qasper: extracting word for word and short answers. With more epochs of fine-tuning, the model re-learns the more complex task of abstractive QA.

We see a similar general behavior for Vicuna-4k but with reduced overall scores: Extractive, boolean and unanswerable questions are answered better with more training. The answer F_1 score after 5 epochs of 24.34 only reaches the zero-shot performance of the other models (24.19 and 24.79). The questions that require longer answers only reach about half or even a third / quarter of the performance: 17.86 / 4.75 opposed to 48.21 / 20.10 and 44.86 / 15.49. But the performance on boolean (57.98) and unanswerable (65.68) questions comes close to the other models. When analyzing the answers of Vicuna-4k manually, we see that 385 out of 990 answers are empty strings indicating that the model

Vicuna-4k	Zero-shot	1 epoch	3 epochs	5 epochs
Answer F_1	8.82	20.49	23.78	24.34
Answer F_1 by type				
extractive	6.55	13.34	16.29	17.86
abstractive	4.81	4.24	4.05	4.75
boolean	25.16	57.89	64.17	57.98
none	13.28	55.45	63.30	65.68

Table 4.7.: Vicuna-4k, dev-short set, **LongBench prompt** [6], fine-tuned with QLoRA.

struggles to generate any answer. Therefore, it should be easier for the fine-tuned model to generate one-word answers. These results fall in line with prior work [13] that found that naive fine-tuning of LLMs with longer context does improve long-context performance only very slowly.

Models	Bin count	Vicuna-4k	Vicuna-16k	LongChat-32k
Answer F_1 per paper length				
0k – 4k	333	38.89	50.26	52.15
4k – 8k	593	18.53	43.02	44.45
8k –	64	2.48	39.55	44.09
Answer F_1 per absolute evidence position				
0k – 4k	794	23.99	43.54	46.28
4k – 8k	173	9.33	35.23	37.74
8k –	6	0.00	64.37	67.94
No evidence	77	52.81	75.32	64.94

Table 4.8.: Models we tested, dev-short set, **LongBench prompt** [6], fine-tuned with QLoRA for 5 epochs.

While we only train with sequences of up to 8k tokens, we see an improvement across all analyzed paper lengths and the performance loss after 8k tokens reduces for Vicuna-16k slightly from 4.46 to 3.47 (difference in F_1 score) and for LongChat-32k it almost disappears going from 5.57 to 0.36 (compare Table 4.4 to Table 4.8). However, we still see consistently reduced performance for papers that exceed LLaMA 2's original context window length of 4k and especially for questions where the evidence is further out than 4k. Vicuna-4k does show improvements during training yet still falls way short of the models with longer context windows (Table 4.8).

Deeper analysis (Table 4.8) shows that while training does increase Vicuna-4k's performance on longer papers it still significantly underperforms the other models' ability to handle long context. LongChat's F_1 scores are slightly better than Vicuna-16k's for every bin per length and per evidence position. Yet it still struggles with questions where the dataset annotators found no evidence which we also saw for zero-shot prompting. Fine-tuning also improves the answers for questions with an evidence position beyond 8k tokens from 3.78 and 15.06 F_1 (Table 4.4) to 64.37 and 67.94 (Table 4.8). Only one of these six questions is a boolean question. We are not sure what causes this increased performance but as this bin only contains six questions, we assume that this does not reflect the general performance of the models on this bin as papers with more than 8k tokens see a continued loss in answer quality.

4.3.3. Evidence only

Our previous experiments showed that even the models whose context window was extended with a technique similar to Positonal Interpolation (paragraph 2.3.1) struggle with papers that exceed the original context length of LLaMA 2 of 4k tokens – especially if the evidence lies outside of that range. The question now is if these questions or at least some of them are per se harder to answer. As the Qasper dataset also provides the extracted evidence for each question we evaluate if the performance varies in our analysis if the context given to the model is the evidence only instead of the full paper.

On this task, the performance of the models is quite similar to each other (Table 4.9). Even Vicuna 4k is able to produce comparable results. LongChat's problem with short answers is also reduced with this setup which is probably a result of unanswerable questions being trivial as the provided context is empty. However, LongChat still shows problems with short answers. During qualitative analysis, we saw the same picture: It answers with more than a single word when it should be, ignoring the prompt.

Models	Vicuna-4k	Vicuna-16k	LongChat-32k
Answer F_1	37.33	38.99	36.16
Answer F_1 by type			
extractive	34.57	42.91	37.58
abstractive	18.01	20.46	21.80
boolean	57.26	51.86	47.96
none	78.89	48.72	53.48

Table 4.9.: All models we tested, **evidence only** dev-short set, zero-shot.

Further analysis shows no significant performance variations or at least no clear trends across different paper lengths (Table 4.10). We see lower F_1 scores on evidence positions above 8k tokens. But again, as this bin only contains 6 papers and the previous bin showed only minor performance decrease, we assume that no safe conclusions can be drawn from this result, and this may be an outlier.

When fine-tuning LongChat with QLoRA on the evidence only, we see much quicker better results that exceed those before (Table 4.11). Also, we see a performance improvement immediately for all question types. Only the two shorter types suffer a little the longer the training gets. This could be the same case as we saw before for unanswerable questions that the performance is hitting an upper limit leading to trade-offs. Also, there is no initial quality loss for abstractive answers. We think that this effect disappears because the learning of the mapping between evidence and correct answer is quicker than between full paper and answer. As the performance saturated faster, we only trained for 3 epochs.

After training LongChat on the evidence only, we compare its performance directly against the model that we trained on full papers (Table 4.12). The performance of the

Models	Bin count	Vicuna-4k	Vicuna-16k	LongChat-32k
Answer F_1 per paper length				
0k – 4k	333	36.49	41.26	36.30
4k – 8k	593	36.10	37.58	35.79
8k –	64	36.49	40.23	38.86
Answer F_1 per absolute evidence position				
0k – 4k	794	34.81	37.97	34.95
4k – 8k	173	34.04	36.27	33.65
8k –	6	6.98	26.35	17.71
No evidence	77	66.23	47.01	52.10

Table 4.10.: All models we tested, analysis, **evidence only** dev-short set, zero-shot.

LongChat-32k	Zero-shot	1 epoch	2 epochs	3 epochs
Answer F_1	36.16	55.65	56.97	57.22
Answer F_1 by type				
extractive	37.58	61.04	61.41	62.19
abstractive	21.80	25.20	25.60	27.01
boolean	47.96	72.27	80.99	79.83
none	53.48	84.00	83.33	80.56

Table 4.11.: LongChat-32k, **evidence only** dev-short set, fine-tuned with QLoRA.

context-length-specific model is better on all sub scores. When evaluating the evidence only model on full papers we made an interesting observation: This model has equal or better F_1 scores on all question types except for unanswerable questions. The score for this type of question is probably so low as the model only learned to map the absence of evidence or the presence of a placeholder to the question being unanswerable. We think that the extractive score is about the same as the full paper model because it learned to follow the instruction to answer by copying short excerpts from the text. The abstractive answers could be better because the previously mentioned "re-training" of this type happens quicker. The good performance on boolean questions is probably again a case of increased instruction following. We assume that this result together with less than 8k tokens long training data improving performance on more than 8k tokens long evaluation data means that training the model mostly improves instruction following and does not promote better long-context understanding. But we also note that in order for the model to learn if a question is unanswerable it has to explicitly learn the mapping of no evidence in the whole paper to the question being unanswerable.

During fine-grained analysis (Table 4.13), we see that the model that we trained on evidence only shows no performance decrease with increased paper length but also its performance for shorter papers is worse than those models that were trained on full papers. When binning the results per evidence position, the evidence only model shows only worse performance at questions whose relevant paragraphs are at token positions above 8k and with no relevant paragraphs. Even though the bin for far out evidence is very small, it seems like an indication that the evidence only model struggles with these cases because

LongChat-32k	3 epochs	3 epochs	3 epochs	5 epochs
Train split	evidence only		full paper	
Evaluation split	evidence only	full paper		
Answer F_1	57.22	41.66	44.56	47.02
Answer F_1 by type				
extractive	62.19	47.01	45.18	48.21
abstractive	27.01	24.15	16.59	20.10
boolean	79.83	76.67	80.33	76.47
none	80.56	2.70	66.67	68.54

Table 4.12.: Compare LongChat-32k, fine-tuned with QLoRA on **evidence only** or full paper.

the models that we trained on full papers have similar F_1 scores on this bin. This also shows that training on the full papers is useful as it dramatically improves performance for questions where no evidence is contained in the paper text. It is important to note that we did these experiments on the model with the longest context window: To answer questions on long context, we need a model that is capable of effectively processing long context and also has good instruction following.

LongChat-32k	Bin count	3 epochs	3 epochs	3 epochs	5 epochs
Train split		evidence only		full paper	
Evaluation split		evidence only	full paper		
Answer F_1 per paper length					
0k – 4k	333	57.93	42.68	50.22	52.15
4k – 8k	593	56.86	40.46	41.91	44.45
8k –	64	56.94	42.68	39.75	44.09
Answer F_1 per absolute evidence position					
0k – 4k	794	54.27	44.48	43.94	46.28
4k – 8k	173	50.86	37.46	34.80	37.74
8k –	6	63.61	43.71	64.76	67.94
No evidence	77	93.51	16.88	61.04	64.94

Table 4.13.: Analysis, compare LongChat-32k, fine-tuned with QLoRA on **evidence only** or full paper.

4.3.4. Two-step Prompts

Inspired by the results of using only the evidence as context to answer the questions, we argue that a chain-of-thought prompt could increase performance: The model has to extract the relevant paragraphs first and then answer the questions based on the found evidence. We still expect some performance drop for longer context as the paragraphs' extraction should also become harder with increased paper length.

We only proceeded with LongChat to reduce the number of experiments (especially training) we have to run. We chose LongChat as it showed the highest gains during training, better scores on the question types with high amounts of information (extractive, abstractive) and better performance on longer papers.

LongChat-32k	Zero-shot	1 epoch	3 epochs	5 epochs
Answer F_1	24.94	19.93	34.52	39.08
Answer F_1 by type				
extractive	23.19	8.27	29.98	37.50
abstractive	17.35	1.52	10.28	14.92
boolean	57.96	23.76	52.34	49.51
none	11.84	98.45	89.57	89.09
Evidence F_1	12.73	25.74	34.32	38.45

Table 4.14.: LongChat, dev-short set, **two-step prompt**, fine-tuned with QLoRA.

During training, we see an initial drop in performance for all question types that can be answered with the paper as context (Table 4.14). When looking at the evidence score and during qualitative analysis, we see that the model does not extract the correct paragraphs leading to an inability to answer most of the questions. After the first epoch, the model outputs “No relevant paragraphs found” in 900 of 990 cases. This improves with each epoch as the paragraphs’ extraction improves. After five epochs, for 535 out of 990 questions the model finds evidence. Still after the same number of epochs as the one-step prompt model, this model still performs worse.

Even for longer papers and more difficult to reach evidence, the two-step prompt does not improve performance as the evidence extraction also suffers on longer context and also does not help even inside the original context window (Table 4.15). Out of 990 questions, the fine-tuned model still finds no evidence for 455 questions. When analyzing the performance based on the extracted evidence (Table 4.15), we see that the model’s performance also does not improve when finding at least some evidence: When evidence is found, the F_1 score of 46.45 is still worse than the best one-step model with 47.02 on dev-short. While we see better performance on questions with no evidence (81.82 vs. 64.94), the problem is that this is not known a priori like paper length meaning we can not reliably fuse the answers for these questions of the two-step prompt model with the others from the one-step prompt model.

We also did one training run with the LoRA rank r increased to 16 which doubles the number of parameters affected by training. As the model has to handle two-tasks for the two-step prompt, changing more parameters may improve performance. But like it was noted in the publication of QLoRA [22], as long as LoRA is applied to all layers and building blocks of the model, which is the case for QLoRA, r is not very important for fine-tuning performance. Therefore, the results were very similar to the two-step prompt with $r = 8$ (compare Table 4.14 to Table A.1). We exhibit the detailed results in subsection A.3.1.

We also tested if changing the temperature increases performance (Table 4.16): Our rationale is that the most probable evidence is none as the placeholder string for this is always the same and occurs more often during training than any other evidence string.

LongChat-32k	Bin count	Zero-shot	5 QLoRA epochs	Zero-shot	5 QLoRA epochs
Variation		Two-step	Two-step	Advanced two-step	Advanced two-step
Answer F_1 per paper length					
0k – 4k	333	25.68	41.57	19.26	44.78
4k – 8k	593	24.85	37.77	17.50	39.66
8k –	64	21.97	38.23	13.83	36.44
Evidence F_1 per paper length					
0k – 4k	333	12.91	38.09	0.00	12.01
4k – 8k	593	12.31	39.05	0.00	10.62
8k –	64	15.62	34.64	0.00	20.31
Answer F_1 per absolute evidence position					
0k – 4k	794	26.84	36.00	18.89	40.54
4k – 8k	173	23.40	30.31	16.64	34.97
8k –	6	28.96	56.19	2.75	39.78
No evidence	77	6.69	81.82	9.61	57.14
Evidence F_1 per absolute evidence position					
0k – 4k	794	9.07	35.17	0.00	7.81
4k – 8k	173	5.78	28.75	0.00	5.78
8k –	6	0.00	16.67	0.00	16.67
No evidence	77	61.04	87.01	0.00	61.04
Answer F_1 per extracted evidence					
none	(varies)	24.03	30.40	0.00	47.93
some	(varies)	25.52	46.45	17.85	39.28
Evidence F_1 per extracted evidence					
none	(varies)	32.90	40.44	0.00	53.46
some	(varies)	0.00	36.75	0.00	0.00

Table 4.15.: LongChat, dev-short set, **two-step prompts**, compare initial and advanced prompt, zero-shot vs. fine-tuned with QLoRA.

Also, it is not that important if the found paragraphs are perfectly correct (e.g., not too long): It just has to be useful to answer the question. Yet, increasing the temperature monotonously decreases both the evidence and answer F_1 scores. On top of reduced quality, the percentage of empty evidence rises from ~46% (0.0) to ~66% (1.0).

LongChat-32k	0.0	0.2	0.4	0.6	0.8	1.0
Answer F_1	39.08	37.61	35.05	33.15	30.57	29.60
Evidence F_1	38.45	37.20	35.31	33.54	31.85	29.16

Table 4.16.: LongChat, dev-short set, **two-step prompt**, fine-tuned 5 epochs with QLoRA, varying temperatures.

We now employ all small techniques we presented in the section before to improve the two-step prompt: Prefix for evidence extraction, question repetition and changing the ratio of the examples in the training data that feature no found evidence. In the training data (<8k tokens), only around 16% of the questions are annotated with no evidence. However, the model that we fine-tuned on the "standard" two-step prompt generates no extracted evidence for around 40% of the questions which is 2.5 times as often. We assume a linear dependency between percentage of training answers without evidence and the percentage of generated answers without evidence. We lower the ratio of questions with no evidence in the training data to around 6% to arrive at 16% of generated empty evidence.

LongChat-32k	Zero-shot	1 epoch	3 epochs	5 epochs
Answer F_1	17.85	27.59	41.54	41.18
Answer F_1 by type				
extractive	16.37	22.27	45.00	41.82
abstractive	16.03	10.64	21.06	19.41
boolean	36.75	59.66	68.14	58.10
none	5.33	59.79	40.23	69.23
Evidence F_1	0.00	26.37	31.12	35.13

Table 4.17.: LongChat, dev-short set, **two-step prompt, improved**, fine-tuned with QLoRA.

While the answer F_1 score does improve with this adapted prompt for the fine-tuned model (Table 4.17) when compared to the simpler two-step prompt (Table 4.14), the evidence F_1 is lower even though the percentage of empty evidence drops from around 46% to around 22%. Also, for the zero-shot prompt all question types show worse results and the evidence score even drops to 0.0. Manual investigation shows that the model generated very long paragraphs as evidence in the zero-shot setup which led to this score.

In further analysis (Table 4.15), the advanced two-step prompt shows slightly better results for papers with under 8k tokens and evidence below the same threshold. But the F_1 scores are still below those of the one-step prompt (Table 4.8). Even the answers where the model found evidence are not better than those of the previous two-step prompt. We therefore can only use the evidence extracted by the two-step prompt as an addition to the answers generated with the LongBench prompt (fine-tuned) but can not use it to generate answers.

4.3.5. Split Prompt

To determine how to split the papers, we first have to find out what the optimal split length is. It is possible that the evidence extraction on less than 4k tokens (e.g., 2k) is better. Our investigation (Table 4.18) shows that the two-step prompt is better than no evidence extraction in every case. The difference is even greater when binned per evidence position. We also conclude that the difference between none and extracted evidence for shorter papers does not justify splitting the text further than into parts of 4k tokens or less. When splitting the papers into parts, we try to add a buffer for the prompt meaning that paper text and prompt together should still be shorter than 4k tokens. This leads to 216 papers that are not split even though 333 paper have a length of 4k tokens or less. We try to split them on section borders.

Method	Bin count	No evidence	Two-step prompt
Evidence F_1 per paper length			
0k – 4k	333	28.83	38.09
0k – 2k	18	55.56	59.26
2k – 4k	315	27.30	36.88
Evidence F_1 per absolute evidence position			
0k – 4k	794	18.77	35.17
0k – 2k	531	22.41	34.08
2k – 4k	376	13.83	33.92

Table 4.18.: Comparison of evidence extract on <4k tokens, dev-short set, no evidence vs. evidence extraction via two-step prompt.

We utilize the models we already trained for this use case: We use the model trained on the two-step prompt for evidence extraction and the evidence only model to answer the questions based on the extracted evidence. However, we saw no performance gains compared to the two-step prompt (Table 4.19) as the evidence extraction experienced a quality degradation.

Model	2-step prompt	Split prompt
Answer F_1	39.08	33.32
Answer F_1 by type		
extractive	37.50	31.77
abstractive	14.92	10.62
boolean	49.51	37.86
none	89.09	85.84
Evidence F_1	38.45	23.04

Table 4.19.: LongChat, dev-short set, **split prompt** compared to previous results.

4.3.6. Modify Attention Algorithm

Our modified attention algorithm masks the k lowest attention weights along the sequence length before the softmax function (see Equation 2.2) to focus more attention on the more important tokens. We start our experiments with a "window" size of 4k to see if masking all tokens that exceed the original context window by their number enables the model to perform better.

Our naive implementation requires creating an index matrix with size of the attention matrix minus k on each dimension. This leads to out-of-memory errors even with four GPUs during inference for papers with a length of over 8k tokens. We therefore exclude the affected bins of our results. As the performance for our earlier approaches already drops after the original context window size of 4k, the truncated results should still show if the modified attention improves F_1 scores on longer context.

The modified attention seems to have no positive impact as the performance stays about the same or decreases slightly (Table 4.20).

Models Attention	Bin count	Vicuna-4k unmodified	Vicuna-4k 4k window	LongChat-32k unmodified	LongChat-32k 4k window
Answer F_1 per paper length					
0k – 4k	333	38.89	38.89	52.15	52.05
4k – 8k	593	18.53	17.86	44.45	43.60
Answer F_1 per absolute evidence position					
0k – 4k	794	23.99	23.52	46.28	44.03
4k – 8k	173	9.33	9.85	37.74	36.73

Table 4.20.: **Modified attention**, dev-short set, LongBench prompt [6], fine-tuned with QLoRA for 5 epochs (before modification).

4.3.7. Final Comparison against Baselines

Finally, we want to compare the results of our experiments against task-specific models and strong LLMs.

Qasper Baseline First, we compare our best approach against the baseline model from the original publication of the Qasper dataset [21]. Their model is the Longformer-Encoder-Decoder (LED) [7] in two sizes: base and large. It contains more fine-grained results than the comparison on the ZeroSCROLLS [64] subset of Qasper. The authors of Qasper provide the detailed results per question type for the answer F_1 of the LED-base model and evidence F_1 of both model sizes. Also, they estimate a lower bound for the human performance on the test set by calculating the agreement between different annotator answers for each question. Their best model for question answering is LED-base that receives the full paper as input. One variant includes evidence extraction during training. We chose LongChat-32k trained with the LongBench prompt as our best approach as it had the best results on the dev-short set and showed the best scaling behavior for papers and evidence exceeding LLaMA's original context window length.

Models	LongChat-32k LongBench prompt zero-shot	LongChat-32k LongBench prompt 5 epochs	LED-base without evidence extraction	LED-base with evidence extraction	Human (lower bound)
Dev answer F_1	23.83	46.87	29.05	28.01	–
Dev answer F_1 by type					
extractive	26.21	48.00	26.07	24.62	–
abstractive	20.39	20.34	16.59	13.86	–
boolean	35.87	76.23	67.48	63.64	–
none	0.04	67.78	28.57	38.89	–
Test answer F_1	28.81	55.20	32.80	33.63	60.92
Test answer F_1 by type					
extractive	28.39	54.89	30.96	29.97	58.92
abstractive	20.82	18.79	15.76	15.02	39.71
boolean	56.11	84.68	70.33	68.90	78.98
none	2.14	86.42	26.21	44.97	69.44

Table 4.21.: Comparison of our approaches against baselines from the Qasper paper, full dev and test set.

Our comparison (Table 4.21) shows that LED has a similar distribution of the F_1 scores per type. The extractive score is higher than the abstractive score and the boolean score is the highest or close to it. We can also see a similar behavior of the LED model to the two-step prompt when integrating evidence extraction into the answer generation process: The extractive and abstractive scores suffer while the model detects unanswerable questions better. Also, our best approach performs better on questions with very short answers (yes/no, unanswerable) than the lower bound for human performance. This could be an explanation of our observation that longer training does not improve these scores after they reach a certain level (trade-off: short vs. long answers). However, the quality of the abstractive answers is considerably worse (39.71 vs. 18.79).

For the evidence extraction, our best model is LongChat-32k fine-tuned with the two-step prompt. While the evidence extraction did not improve the answer quality in our case, it can be a useful addition for the user of a QA system to contextualize the answer. Here, the difference between our approach and the Qasper baseline LED-large (Table 4.22) is not as high as for the answer F_1 score but we still see a clear improvement over the baseline.

ZeroSCROLLS Baselines Our second comparison is on the ZeroSCROLLS subset of the Qasper test set which we think is representative enough for the full test set (subsection 4.1.1) to use it for comparison to strong LLMs. We compare our approach to three of the models that the authors of ZeroSCROLLS evaluated: Flan-UL2 which is the strongest open-source and encoder-decoder model on ZeroSCROLLS and particularly on the Qasper subet, GPT-4 which is the strongest overall model on ZeroSCROLLS and one of the highest

Models	LongChat-32k two-step prompt 5 epochs	LED-base	LED-large	Human (lower bound)
Dev evidence F_1	38.27	23.94	31.25	–
Test evidence F_1	42.57	29.85	39.37	71.62

Table 4.22.: Comparison of our approaches against baselines from the Qasper paper, full dev and test set, evidence extraction.

performing current LLMs, and CoLT5 which is a model that the authors fine-tuned per task in ZeroSCROLLS.

The ZeroSCROLLS subset uses a slightly different prompt (subsection A.1.1) for Qasper and does not include the title and abstract in the input. We compare our approaches with both prompts: ZeroSCROLLS (ZC) and LongBench (LB). We included both title and abstract during our experiments because we wanted the model to have as much information from the paper as possible. For the comparison with the other models that the authors of ZeroSCROLLS evaluated we do not include title or abstract.

Models	Flan-UL2	GPT-4	CoLT5	LC-32k 0-shot	LC-32k 0-shot	LC-32k 5e	LC-32k 5e
Training prompt	–	–	ZC	–	–	LB	LB
Inference prompt	ZC	ZC	ZC	ZC	LB	ZC	LB
Answer F_1	56.90	50.70	53.10	25.80	31.07	46.90	52.73

Table 4.23.: Baseline results from ZeroSCROLLS benchmark [64] compared to our results (LongChat-32k (LC-32k), 5 epochs (5e)), ZeroSCROLLS subset of Qasper test set.

With the LongBench prompt used during inference, our best approach exceeds GPT-4's F_1 score on the ZeroSCROLLS subset, comes close to the strongest model, and represents a great improvement over the zero-shot setup (Table 4.23). It is important to note that the ZeroSCROLLS authors mentioned that GPT-4 sometimes struggled more than other models to follow the prompt on Qasper. When we use the same prompt as the other models, both our zero-shot and the fine-tuned model lose more than 5 F_1 points showing how important prompting can be. As the performance drop is almost the same, we assume that for the fine-tuned model this is not a result of the mismatch between the training prompt and the inference prompt. The fine-tuned LongChat-32k model with the LongBench prompt is only able to almost match the task-specific model. We assume that this observation and the fact that Flan-UL2 is the best performing model are a result of these models being full transformers with an encoder and a decoder. The bidirectional encoder that processes the context together with the question and the prompt before generating the answer could help here.

5. Conclusion

After presenting and discussing our results, we want to give answers to our research questions (section 1.1) and discuss how future work could further research LLM-based QA systems for scientific papers.

5.1. Answers to Research Questions

Our experiments resulted in the following answers to the research questions we formulated in the beginning.

Research Question 1: How well do LLMs process scientific papers – especially considering their length?

We observe that the (unmodified) small open-source long-context LLMs we tested are able to process scientific papers with up to about 16k tokens from the Qasper dataset but fall short of commercial LLMs. The performance also drops after the context exceeds the original context window – especially if the relevant information to answer to question lies in that region of the paper. Yet it is important to note that we saw that techniques like Positional Interpolation can increase the long-context understanding of LLMs.

Research Question 2: How much does fine-tuning a small LLM on only a single datacenter GPU increase performance on one specific task compared to bigger and better LLMs?

When we employ the current techniques for efficient training QLoRA and FlashAttention, we can fine-tune the models on papers with a length of up to 8k tokens on a single datacenter GPU that is available to a university student for research. While Qasper (including the training split) also contains a lot of papers with more than 8k tokens, the performance of our fine-tuned model still increases for these longer papers without being trained on these lengths. Experiments with models that we only trained on extracted paragraphs without providing the model the full paper suggest that our training primarily improves instruction following but also improves the models' ability to determine if a question is unanswerable as it has to learn the connection between the absence of relevant information and the unanswerability of the question. When comparing our results against baselines, we saw that our best approach reaches or surpasses the result of GPT-4 which is or was until recently the SOTA for many NLP tasks.

Research Question 3: How much can prompting / fine-tuning improve the truthfulness of the answers of the LLM? How much can prompting / fine-tuning improve the truthfulness of the answers of the LLM? How high is the human performance?

Especially LongChat-32k benefits a lot from fine-tuning as it enables it to follow our instructions and give concise answers to yes/no questions or questions that are not answerable with the given context. For these question types, our modified model even slightly exceeds the lower bound of human performance while there is still a large gap for free form answers. As models like GPT-4 that are in general better in almost every NLP task than the models we tested also achieve better results, we assume that even if the F_1 score can not determine semantic meaning, it indicates that our approaches help the models to generate better answers.

5.2. Future Work

We think there are two main starting points to continue the presented work: Besides from making our naively modified attention algorithm more efficient in the spirit of FlashAttention or similar techniques, the masking could be replaced by a cropping of the sequence. After passing the sequence into the model and generating the hidden states, these states could again be used to determine the importance of the tokens in the sequence from the attention weights. This information could be brought back to the input where the tokens with a low attention weight are deleted. This shortened sequence is then again fed into the model and used to generate an answer.

This idea resembles retrieval-augmented generation where one model determines which small parts of the large context are important and passes them to a second model that uses only these as its context. As this reduces the length of the sequence processed by the second model, this could make it possible to use a bigger model if we assume a fixed compute and memory budget. For example, a retrieval model selects the important paragraphs from the paper and Vicuna-13B or an even bigger model generates the answer.

Bibliography

[1] Meta AI. *Introducing LLaMA: A foundational, 65-billion-parameter large language model.* 2023. URL: https://ai.meta.com/blog/large-language-model-llama-meta-ai/ (visited on 12/11/2023).

[2] Joshua Ainslie et al. "CoLT5: Faster Long-Range Transformers with Conditional Computation". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023.* Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, 2023, pp. 5085–5100. URL: https://aclanthology.org/2023.emnlp-main.309.

[3] Rohan Anil et al. "Gemini: A Family of Highly Capable Multimodal Models". In: *CoRR* abs/2312.11805 (2023). DOI: 10.48550/ARXIV.2312.11805. arXiv: 2312.11805. URL: https://doi.org/10.48550/arXiv.2312.11805.

[4] Rohan Anil et al. "PaLM 2 Technical Report". In: *CoRR* abs/2305.10403 (2023). DOI: 10.48550/ARXIV.2305.10403. arXiv: 2305.10403. URL: https://doi.org/10.48550/arXiv.2305.10403.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.0473.

[6] Yushi Bai et al. "LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding". In: *CoRR* abs/2308.14508 (2023). DOI: 10.48550/ARXIV.2308.14508. arXiv: 2308.14508. URL: https://doi.org/10.48550/arXiv.2308.14508.

[7] Iz Beltagy, Matthew E. Peters, and Arman Cohan. "Longformer: The Long-Document Transformer". In: *CoRR* abs/2004.05150 (2020). arXiv: 2004.05150. URL: https://arxiv.org/abs/2004.05150.

[8] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Trans. Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181. URL: https://doi.org/10.1109/72.279181.

[9] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (2003), pp. 1137–1155. URL: http://jmlr.org/papers/v3/bengio03a.html.

[10] Bo-Christer Björk, Annikki Roos, and Mari Lauri. "Scientific journal publishing: yearly volume and open access availability". In: *Inf. Res.* 14.1 (2009). URL: http://www.informationr.net/ir/14-1/paper391.html.

[11] Sid Black et al. “GPT-NeoX-20B: An Open-Source Autoregressive Language Model”. In: *CoRR* abs/2204.06745 (2022). DOI: 10.48550/ARXIV.2204.06745. arXiv: 2204.06745. URL: https://doi.org/10.48550/arXiv.2204.06745.

[12] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[13] Shouyuan Chen et al. “Extending Context Window of Large Language Models via Positional Interpolation”. In: *CoRR* abs/2306.15595 (2023). DOI: 10.48550/ARXIV.2306.15595. arXiv: 2306.15595. URL: https://doi.org/10.48550/arXiv.2306.15595.

[14] Liying Cheng et al. “APE: Argument Pair Extraction from Peer Review and Rebuttal via Multi-task Learning”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020.* Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 7000–7011. DOI: 10.18653/V1/2020.EMNLP-MAIN.569. URL: https://doi.org/10.18653/v1/2020.emnlp-main.569.

[15] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.* 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/ (visited on 03/04/2024).

[16] Aakanksha Chowdhery et al. “PaLM: Scaling Language Modeling with Pathways”. In: *J. Mach. Learn. Res.* 24 (2023), 240:1–240:113. URL: http://jmlr.org/papers/v24/22-1144.html.

[17] Hyung Won Chung et al. “Scaling Instruction-Finetuned Language Models”. In: *CoRR* abs/2210.11416 (2022). DOI: 10.48550/ARXIV.2210.11416. arXiv: 2210.11416. URL: https://doi.org/10.48550/arXiv.2210.11416.

[18] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Math. Control. Signals Syst.* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

[19] Tri Dao. “FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning”. In: *CoRR* abs/2307.08691 (2023). DOI: 10.48550/ARXIV.2307.08691. arXiv: 2307.08691. URL: https://doi.org/10.48550/arXiv.2307.08691.

[20] Tri Dao et al. “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.* Ed. by Sanmi Koyejo et al. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

[21] Pradeep Dasigi et al. “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, NAACL-HLT 2021, Online, June 6-11, 2021*. Ed. by Kristina Toutanova et al. Association for Computational Linguistics, 2021, pp. 4599–4610. DOI: `10.18653/V1/2021.NAACL-MAIN.365`. URL: `https://doi.org/10.18653/v1/2021.naacl-main.365`.

[22] Tim Dettmers et al. “QLoRA: Efficient Finetuning of Quantized LLMs”. In: *CoRR* abs/2305.14314 (2023). DOI: `10.48550/ARXIV.2305.14314`. arXiv: `2305.14314`. URL: `https://doi.org/10.48550/arXiv.2305.14314`.

[23] Jacob Devlin et al. “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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: `10.18653/V1/N19-1423`. URL: `https://doi.org/10.18653/v1/n19-1423`.

[24] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: `https://openreview.net/forum?id=YicbFdNTTy`.

[25] William Fedus, Barret Zoph, and Noam Shazeer. “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity”. In: *J. Mach. Learn. Res.* 23 (2022), 120:1–120:39. URL: `http://jmlr.org/papers/v23/21-0998.html`.

[26] Michael Fire and Carlos Guestrin. “Over-Optimization of Academic Publishing Metrics: Observing Goodhart’s Law in Action”. In: *CoRR* abs/1809.07841 (2018). arXiv: `1809.07841`. URL: `http://arxiv.org/abs/1809.07841`.

[27] Jonas Gehring et al. “Convolutional Sequence to Sequence Learning”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1243–1252. URL: `http://proceedings.mlr.press/v70/gehring17a.html`.

[28] Kelvin Guu et al. “Retrieval Augmented Language Model Pre-Training”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 3929–3938. URL: `http://proceedings.mlr.press/v119/guu20a.html`.

[29] Junqing He et al. “Never Lost in the Middle: Improving Large Language Models via Attention Strengthening Question Answering”. In: *CoRR* abs/2311.09198 (2023). DOI: `10.48550/ARXIV.2311.09198`. arXiv: `2311.09198`. URL: `https://doi.org/10.48550/arXiv.2311.09198`.

[30] Jordan Hoffmann et al. “Training Compute-Optimal Large Language Models”. In: *CoRR* abs/2203.15556 (2022). DOI: `10.48550/ARXIV.2203.15556`. arXiv: `2203.15556`. URL: `https://doi.org/10.48550/arXiv.2203.15556`.

[31] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: `10.1016/0893-6080(89)90020-8`. URL: `https://doi.org/10.1016/0893-6080(89)90020-8`.

[32] Edward J. Hu et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: `https://openreview.net/forum?id=nZeVKeeFYf9`.

[33] Albert Q. Jiang et al. “Mistral 7B”. In: *CoRR* abs/2310.06825 (2023). DOI: `10.48550/ARXIV.2310.06825`. arXiv: `2310.06825`. URL: `https://doi.org/10.48550/arXiv.2310.06825`.

[34] Albert Q. Jiang et al. “Mixtral of Experts”. In: *CoRR* abs/2401.04088 (2024). DOI: `10.48550/ARXIV.2401.04088`. arXiv: `2401.04088`. URL: `https://doi.org/10.48550/arXiv.2401.04088`.

[35] Thorsten Joachims. *Learning to classify text using support vector machines - methods, theory and algorithms*. Vol. 668. The Kluwer international series in engineering and computer science. Kluwer, 2002. ISBN: 978-0-7923-7679-8.

[36] Nal Kalchbrenner et al. “Neural Machine Translation in Linear Time”. In: *CoRR* abs/1610.10099 (2016). arXiv: `1610.10099`. URL: `http://arxiv.org/abs/1610.10099`.

[37] Dongyeop Kang et al. “A Dataset of Peer Reviews (PeerRead): Collection, Insights and NLP Applications”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, 2018, pp. 1647–1661. DOI: `10.18653/V1/N18-1149`. URL: `https://doi.org/10.18653/v1/n18-1149`.

[38] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *CoRR* abs/2001.08361 (2020). arXiv: `2001.08361`. URL: `https://arxiv.org/abs/2001.08361`.

[39] Kaio Ken. *Extending Context is Hard… but not Impossible*. 2023. URL: `https://kaiokendev.github.io/context` (visited on 03/04/2024).

[40] Taku Kudo and John Richardson. “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Eduardo Blanco and Wei Lu. Association for Computational Linguistics, 2018, pp. 66–71. DOI: `10.18653/V1/D18-2012`. URL: `https://doi.org/10.18653/v1/d18-2012`.

[41] Patrick S. H. Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* Ed. by Hugo Larochelle et al. 2020. URL: `https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html`.

[42] Dacheng Li et al. *How Long Can Open-Source LLMs Truly Promise on Context Length?* 2023. URL: `https://lmsys.org/blog/2023-06-29-longchat` (visited on 03/04/2024).

[43] Jialiang Lin et al. "MOPRD: A multidisciplinary open peer review dataset". In: *Neural Comput. Appl.* 35.34 (2023), pp. 24191–24206. DOI: `10.1007/S00521-023-08891-5`. URL: `https://doi.org/10.1007/s00521-023-08891-5`.

[44] Sheng-Chieh Lin et al. "How to Train Your Dragon: Diverse Augmentation Towards Generalizable Dense Retrieval". In: *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023.* Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, 2023, pp. 6385–6400. URL: `https://aclanthology.org/2023.findings-emnlp.423`.

[45] Nelson F. Liu et al. "Lost in the Middle: How Language Models Use Long Contexts". In: *CoRR* abs/2307.03172 (2023). DOI: `10.48550/ARXIV.2307.03172`. arXiv: `2307.03172`. URL: `https://doi.org/10.48550/arXiv.2307.03172`.

[46] Pan Lu et al. "Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.* Ed. by Sanmi Koyejo et al. 2022. URL: `http://papers.nips.cc/paper%5C_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html`.

[47] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. DOI: `10.1017/CBO9780511809071`. URL: `https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf`.

[48] OpenAI. *GPT-3.5 Turbo.* 2024. URL: `https://platform.openai.com/docs/models/gpt-3-5-turbo` (visited on 03/04/2024).

[49] OpenAI. "GPT-4 Technical Report". In: *CoRR* abs/2303.08774 (2023). DOI: `10.48550/ARXIV.2303.08774`. arXiv: `2303.08774`. URL: `https://doi.org/10.48550/arXiv.2303.08774`.

[50] OpenAI. *Introducing ChatGPT.* 2022. URL: `https://openai.com/blog/chatgpt` (visited on 02/23/2024).

[51] OpenAI. *What are tokens and how to count them?* 2024. URL: `https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them` (visited on 03/04/2024).

[52] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.* Ed. by Sanmi Koyejo et al. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

[53] Hariom A. Pandya and Brijesh S. Bhatt. “Question Answering Survey: Directions, Challenges, Datasets, Evaluation Matrices”. In: *CoRR* abs/2112.03572 (2021). arXiv: 2112.03572. URL: https://arxiv.org/abs/2112.03572.

[54] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013.* Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1310–1318. URL: http://proceedings.mlr.press/v28/pascanu13.html.

[55] Anthropic PBC. *Introducing Claude.* 2023. URL: https://www.anthropic.com/news/introducing-claude (visited on 02/23/2024).

[56] “PeerSum: A Peer Review Dataset for Abstractive Multi-document Summarization”. In: *CoRR* abs/2203.01769 (2022). Withdrawn. DOI: 10.48550/ARXIV.2203.01769. arXiv: 2203.01769. URL: https://doi.org/10.48550/arXiv.2203.01769.

[57] Ngoc-Quan Pham, Germán Kruszewski, and Gemma Boleda. “Convolutional Neural Network Language Models”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016.* Ed. by Jian Su, Xavier Carreras, and Kevin Duh. The Association for Computational Linguistics, 2016, pp. 1153–1162. DOI: 10.18653/V1/D16-1123. URL: https://doi.org/10.18653/v1/d16-1123.

[58] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018). URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[59] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[60] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

[61] Corby Rosset. *Turing-NLG: A 17-billion-parameter language model by Microsoft.* 2020. URL: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/ (visited on 03/04/2024).

[62] Gabriele Sarti et al. “Inseq: An Interpretability Toolkit for Sequence Generation Models”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2023, Toronto, Canada, July 10-12, 2023*. Ed. by Danushka Bollegala, Ruihong Huang, and Alan Ritter. Association for Computational Linguistics, 2023, pp. 421–435. DOI: 10.18653/V1/2023.ACL-DEMO.40. URL: https://doi.org/10.18653/v1/2023.acl-demo.40.

[63] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. DOI: 10.18653/V1/P16-1162. URL: https://doi.org/10.18653/v1/p16-1162.

[64] Uri Shaham et al. “ZeroSCROLLS: A Zero-Shot Benchmark for Long Text Understanding”. In: *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, 2023, pp. 7977–7989. URL: https://aclanthology.org/2023.findings-emnlp.536.

[65] Mohammad Shoeybi et al. “Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism”. In: *CoRR* abs/1909.08053 (2019). arXiv: 1909.08053. URL: http://arxiv.org/abs/1909.08053.

[66] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al. 2014, pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

[67] Yi Tay et al. “UL2: Unifying Language Learning Paradigms”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=6ruVLB727MC.

[68] MosaicML NLP Team. *Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs*. 2023. URL: www.mosaicml.com/blog/mpt-7b (visited on 03/04/2024).

[69] Romal Thoppilan et al. “LaMDA: Language Models for Dialog Applications”. In: *CoRR* abs/2201.08239 (2022). arXiv: 2201.08239. URL: https://arxiv.org/abs/2201.08239.

[70] Hugo Touvron et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: *CoRR* abs/2307.09288 (2023). DOI: 10.48550/ARXIV.2307.09288. arXiv: 2307.09288. URL: https://doi.org/10.48550/arXiv.2307.09288.

[71] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *CoRR* abs/2302.13971 (2023). DOI: 10.48550/ARXIV.2302.13971. arXiv: 2302.13971. URL: https://doi.org/10.48550/arXiv.2302.13971.

[72] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

[73] Lei Wang et al. "T-SciQ: Teaching Multimodal Chain-of-Thought Reasoning via Large Language Model Signals for Science Question Answering". In: *CoRR* abs/2305.03453 (2023). DOI: `10.48550/ARXIV.2305.03453`. arXiv: `2305.03453`. URL: `https://doi.org/10.48550/arXiv.2305.03453`.

[74] Qingyun Wang et al. "ReviewRobot: Explainable Paper Review Generation based on Knowledge Synthesis". In: *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020, Dublin, Ireland, December 15-18, 2020*. Ed. by Brian Davis et al. Association for Computational Linguistics, 2020, pp. 384–397. URL: `https://aclanthology.org/2020.inlg-1.44/`.

[75] Zhen Wang. "Modern Question Answering Datasets and Benchmarks: A Survey". In: *CoRR* abs/2206.15030 (2022). DOI: `10.48550/ARXIV.2206.15030`. arXiv: `2206.15030`. URL: `https://doi.org/10.48550/arXiv.2206.15030`.

[76] Jason Wei et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: `https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html`.

[77] Jason Wei et al. "Finetuned Language Models are Zero-Shot Learners". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: `https://openreview.net/forum?id=gEZrGCozdqR`.

[78] Peng Xu et al. "Retrieval meets Long Context Large Language Models". In: *CoRR* abs/2310.03025 (2023). DOI: `10.48550/ARXIV.2310.03025`. arXiv: `2310.03025`. URL: `https://doi.org/10.48550/arXiv.2310.03025`.

[79] Pengcheng Yang et al. "Automatic Academic Paper Rating Based on Modularized Hierarchical Convolutional Neural Network". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 496–502. DOI: `10.18653/V1/P18-2079`. URL: `https://aclanthology.org/P18-2079/`.

[80] Ori Yoran et al. "Answering Questions by Meta-Reasoning over Multiple Chains of Thought". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, 2023, pp. 5942–5966. URL: `https://aclanthology.org/2023.emnlp-main.364`.

[81] Lianmin Zheng et al. “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: `http://papers.nips.cc/paper%5C_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets%5C_and%5C_Benchmarks.html`.

[82] Daniel M. Ziegler et al. “Fine-Tuning Language Models from Human Preferences”. In: *CoRR* abs/1909.08593 (2019). arXiv: `1909.08593`. URL: `http://arxiv.org/abs/1909.08593`.

A. Appendix

A.1. Prompts

We used the following prompts during our experiments. <CONTEXT> stands for the paper text or a shortened version of it while <QUESTION> is the placeholder for the specific question on the provided context.

A.1.1. ZeroSCROLLS

You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable". Do not provide any explanation.

Article: <CONTEXT>

Question: <QUESTION>

A.1.2. LongBench (our version)

You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write 'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or 'unanswerable'. Do not provide any explanation.
Article: <CONTEXT>
Answer the question based on the above article as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write 'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or 'unanswerable'. Do not provide any explanation.
Question: <QUESTION>

A.1.3. Evidence only

You are given excerpts from a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the excerpts from an article, write 'unanswerable'.

If the question is a yes/no question, answer ’yes’, ’no’, or ’unanswerable’. Do not provide any explanation.
Excerpts from Article: <CONTEXT>
Answer the question based on the above excerpts from an article as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the excerpts from an article, write ’unanswerable’. If the question is a yes/no question, answer ’yes’, ’no’, or ’unanswerable’. Do not provide any explanation.
Question: <QUESTION>

A.1.4. Two-turn

Turn 0:
You are given a scientific article and a question. Extract all paragraphs that are relevant to answer the question. Copy them word by word from the article. If there are no relevant paragraphs answer ’No relevant paragraphs found’. Do not provide any explanation.
Article: <CONTEXT>
Extract all paragraphs that are relevant to answer the question. Copy them word by word from the article. If there are no relevant paragraphs answer ’No relevant paragraphs found’.
Question: <QUESTION>

Turn 1:
Answer the question based on your extracted relevant paragraphs from the above article and answer as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article (your last answer was possibly ’No relevant paragraphs found’), write ’unanswerable’. If the question is a yes/no question, answer ’yes’, ’no’, or ’unanswerable’. Do not provide any explanation.
Question: <QUESTION>

A.1.5. Two-turn, Advanced (Prefix for Evidence, Question Repeated)

Turn 0:
You are given a scientific article and a question. Extract all paragraphs that are relevant to answer the question. Copy them word by word from the article and start with ’Found these relevant paragraphs:’. If there are no relevant paragraphs answer ’No relevant paragraphs found’. Do not provide any explanation.
Question: <QUESTION>
Article: <CONTEXT>
Extract all paragraphs that are relevant to answer the question. Copy them word by word from the article. If there are no relevant paragraphs answer ’No relevant paragraphs found’.
Question: <QUESTION>

Turn 1:
Answer the question based on your extracted relevant paragraphs from the above article and answer as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article (your last answer was possibly 'No relevant paragraphs found'), write 'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or 'unanswerable'. Do not provide any explanation.
Question: <QUESTION>

A.1.6. Split

Evidence extraction:
You are given a few sections from a scientific article and a question. Extract all paragraphs that are relevant to answer the question. Copy them word by word from the provided sections from the article. If there are no relevant paragraphs answer 'No relevant paragraphs found'. Do not provide any explanation.
Sections from an article: <CONTEXT>
Extract all paragraphs that are relevant to answer the question. Copy them word by word from the provided sections from the article. If there are no relevant paragraphs answer 'No relevant paragraphs found'.
Question: <QUESTION>

Generated evidence only: You are given extracted relevant paragraphs from a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the relevant paragraphs from an article, write 'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or 'unanswerable'. Do not provide any explanation.
Relevant paragraphs from Article: <CONTEXT>
Answer the question based on the above relevant paragraphs from an article as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the relevant paragraphs from an article, write 'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or 'unanswerable'. Do not provide any explanation.
Question: <QUESTION>

A.2. Answers from Models

Here, we show full answers from models to highlight certain aspects of their answering style.

A.2.1. LongChat, Unanswerable

We **highlight** the part of the answer that determines the question as unanswerable.

- Examples 1
  - Question: “How do their results compare to state-of-the-art?”
  - Answer: “The results of the current study, which focuses on detecting offensive language in English tweets, are compared to state-of-the-art methods. The authors explore various approaches, including Google universal sentence encoder, Fasttext-based embedding, Dynamic Mode Decomposition (DMD) based features, and Random Kitchen Sink (RKS) explicit mapping. The study finds that the RKS approach improves the evaluation scores, achieving competitive results. However, the article does not provide specific comparisons to state-of-the-art methods, **leaving the answer to the question ’unanswerable’**”.
- Example 2
  - Question: “How large are the textual descriptions of entities?”
  - Answer: “The textual descriptions of entities in the article range from a few words to several sentences, depending on the context and the specific entity being described. **The article does not provide specific length constraints for these descriptions**, but it is mentioned that the text encoders used in the study can handle variable-length sentences.”

A.3. Results

Some experiments we ran provided not enough new information to show them in the main text, but we list them here.

A.3.1. Two-step Prompt, r = 16

LongChat-32k	Zero-shot	1 epoch	3 epochs	5 epochs
Answer F_1	24.94	20.44	32.31	38.50
Answer F_1 by type				
extractive	23.19	8.95	26.27	36.28
abstractive	17.35	1.66	8.12	13.74
boolean	57.96	24.75	53.27	50.49
none	11.84	98.45	89.74	90.18
Evidence F_1	12.73	26.19	32.99	36.73

Table A.1.: LongChat, dev-short set, **two-step prompt**, fine-tuned with QLoRA, rank r set to 16 (instead of 8).