InstructRetro: Instruction Tuning post Retrieval-Augmented Pretraining

Boxin Wang [1] **Wei Ping** [1] **Lawrence McAfee** [1] **Peng Xu** [1] **Bo Li** [2] **Mohammad Shoeybi** [1] **Bryan Catanzaro** [1]

Abstract

Pretraining auto-regressive large language models (LLMs) with retrieval demonstrates better perplexity and factual accuracy by leveraging external databases. However, the size of existing pretrained retrieval-augmented LLM is still limited (e.g., *Retro* has 7.5B parameters), which limits the effectiveness of instruction tuning and zero-shot generalization. In this work, we introduce *Retro* 48B, the largest LLM pretrained with retrieval. Specifically, we continue to pretrain a 43B GPT model on additional 100 billion tokens using the Retro augmentation method by retrieving from 1.2 trillion tokens. Notably, the obtained foundation model, Retro 48B, largely outperforms the counterpart GPT 43B trained on 1.2T tokens in terms of perplexity with only 2.58% additional GPU hours, demonstrating the significant scaling potential of the method. After instruction tuning on Retro, *InstructRetro* demonstrates significant improvement over the instruction tuned GPT on a wide range of zero-shot tasks. Specifically, the average improvement of InstructRetro is 7% over its GPT counterpart across 8 short-form QA and reading comprehension tasks, 10% over GPT across 4 challenging long-form QA tasks, and 16% over GPT across 3 summarization tasks. Surprisingly, we find that one can ablate the encoder from InstructRetro architecture and directly use its decoder backbone, while achieving comparable results. Our results highlight the promising direction to obtain a better GPT decoder through continued pretraining with retrieval before instruction tuning. Our code and checkpoints are publicly available at: https://huggingface.co/nvidia/retro-48b-instruct-4k.

[1]NVIDIA [2]UIUC. Correspondence to: Boxin Wang <boxinw@nvidia.com>, Wei Ping <wping@nvidia.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Retrieval helps large language models (LLM) to handle current events, detailed knowledge, proprietary information not in pretraining, and to improve factual grounding (e.g., Nakano et al., 2021; Thoppilan et al., 2022; Borgeaud et al., 2022). In the previous study, pretraining auto-regressive language model with retrieval (i.e., *Retro*) demonstrates successes in reducing perplexity (Borgeaud et al., 2022) and improving factual accuracy (Wang et al., 2023a).

In the past year, the decoder-only auto-regressive LLMs have demonstrated remarkable successes (e.g., OpenAI, 2022; 2023), because *i)* LLMs have been scaled to hundreds of billion parameters (Brown et al., 2020a; Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022), *ii)* pretraining corpus has been scaled up to trillions of tokens (Hoffmann et al., 2022; Touvron et al., 2023a;b), and *iii)* instruction tuning (Wei et al., 2022a; Chung et al., 2022) and reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022) recipes have been applied on these pretrained LLMs.

In contrast, the pretrained retrieval-augmented language models still have a relatively small number of parameters trained with a limited number of tokens. For example, the auto-regressive *Retro* has 7.5B parameters and is trained on 600B tokens (Borgeaud et al., 2022), *Retro++* has 9.5B parameters and is trained on 330B tokens (Wang et al., 2023a), and T5-based *Atlas* has 11B parameters and is trained with retrieval on maximum 327M tokens (Izacard et al., 2022b). In addition, none of previous models have been applied with instruction tuning and RLHF to enhance usability. The lack of scaling could also limit the effectiveness of instruction tuning (Wei et al., 2022a) and other intriguing properties that exist in large language models (Wei et al., 2022b).

In this work, we scale up *Retro* up to 48B parameters, trained on 1.2T tokens in total, i.e., 1.1T tokens for pretraining its GPT backbone, 100B tokens for continued retrieval-augmented pretraining while retrieving from 1.2T tokens. As a result, we can mitigate the zero-shot generalization gap on a wide range of tasks after applying instruction tuning.

Specifically, we make the following contributions:

1. We introduce *Retro* 48B, the largest LLM pretrained with retrieval. To save the computation budget, we continue to pretrain a 43B parameter GPT

model (originally trained on 1.1T tokens) on additional 100B tokens by retrieving from 1.2T tokens. In contrast to *Retro-fitting* (Borgeaud et al., 2022), that freezes pretrained decoder weights, we unfreeze the decoder, jointly train all the parameters and find better perplexity. [1] Notably, with only 2.58% additional GPU hours, the perplexity improvement of Retro 48B over its GPT 43B counterpart is still significant even at this scale, demonstrating that the value of retrieval does not diminish with scaling model size.

2. After instruction tuning, *InstructRetro* 48B demonstrates strong zero-shot capability to incorporate context for various downstream tasks, and significantly outperforms instruction-tuned GPT with retrieval-augmented generation (RAG). The training pipeline of InstructRetro is shown in Figure 1.
3. Perhaps surprisingly, we find that one can directly ablate the encoder from *InstructRetro* 48B. The obtained decoder-only *InstructRetro* 43B still achieves very comparable results on downstream tasks. This highlights the promising direction of obtaining better decoder-only LLMs through continued pretraining with retrieval before instruction tuning.

We organize the rest of the paper as follows. We discuss related work in § 2. We introduce the continued pretraining of Retro 48B in § 3 and the instruction tuning recipe in § 4. We report results in Section 5 and conclude the paper in § 6.

2. Related Work

Retrieval-augmented language models have been established for open domain question answering for years (Karpukhin et al., 2020; Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2022; Izacard et al., 2022b). In the previous study, language models have been augmented with retrieval at inference (Khandelwal et al., 2020; Yogatama et al., 2021), fine-tuning (Karpukhin et al., 2020; Lewis et al., 2020; Guu et al., 2020; Huang et al., 2023; Shi et al., 2023b), and pretraining (Borgeaud et al., 2022; Izacard et al., 2022b; Wang et al., 2023a; Shi et al., 2023a). Retrieval-augmented pretraining is particularly interesting, as it can largely reduce model perplexity (Borgeaud et al., 2022), enhance factuality (Wang et al., 2023a), and improve downstream task accuracy after task-specific fine-tuning (Izacard et al., 2022b) and reasoning capability (Shi et al., 2023a).

In contrast to the state-of-the-art decoder-only LLMs with hundreds of billion parameters (Brown et al., 2020b; Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022), the sizes of pretrained retrieval-augmented LLMs are still around 10B parameters (Borgeaud et al., 2022; Wang et al., 2023a; Izacard et al., 2022a), which largely limits the zero-shot generalization capability after instruction tuning (Wei et al., 2022a; Ouyang et al., 2022; Chung et al., 2022). For example, Wei et al. (2022a) find instruction tuning to be more effective when the decoder-only LLM has around 50B parameters.

Instruction tuning aims to teach LLMs to follow natural language instructions (Wei et al., 2022a; Ouyang et al., 2022; Sanh et al., 2022b; Mishra et al., 2022), which becomes an indispensable ingredient to build the state-of-the-art LLMs for downstream tasks (OpenAI, 2022; 2023; Touvron et al., 2023b). In the past years, many high-quality instruction tuning datasets have been created, including FLAN (Chung et al., 2022), OpenAssistant (Köpf et al., 2023), Self-Instruct (Wang et al., 2022a), Dolly (Conover et al., 2023), Unnatural Instructions (Honovich et al., 2022). A concurrent work, RA-DIT (Lin et al., 2024), focuses on retrieval-augmented instruction tuning and further augments 20 instruction tuning datasets with retrieval, which supports fine-tuning both LLM and retriever to yield high-quality neighbors. In contrast, our work focuses on retrieval-augmented pretraining, which extends the scale of the retrieval database to trillions of tokens. Although the two work are orthogonal, InstructRetro 43B outperforms RA-DIT 65B on certain benchmarks as shown in Table 1. We leave it as an interesting future direction to apply RA-DIT retrieval-augmented instruction tuning data to the instruction tuning stage of InstructRetro for further performance improvement of retrieval-augmented LLMs.

3. Continued Pretraining of GPT with Retrieval

In this section, we start by introducing the preliminaries of *Retro* (Borgeaud et al., 2022) and highlight some key differences between Retro and GPT. We then go through the pretraining details of how we scale up the size of Retro to 48B, a size that has never been studied before.

3.1. Preliminaries of Retro

Retro (Borgeaud et al., 2022) is an auto-regressive language model pretrained with retrieval augmentation. While Retro shares the backbone of GPT models, Retro differs from GPT by incorporating an additional *Retro encoder*. The Retro encoder is adept at encoding features of retrieved neighbors from *external knowledge bases*. Furthermore, Retro adds *chunk-wise cross-attention* layers within its decoder transformer architecture to integrate retrieved information from the Retro encoder effectively. This design paradigm also makes Retro different from the encoder-decoder architecture (e.g., T5 (Raffel et al., 2020) and Atlas (Izacard et al., 2022a)). The success of scaling decoder-only autoregres-

[1] Note that, it turns out that unfreezing of decoder is an important design not only for better perplexity, and it eventually leads to the interesting finding after instruction tuning.

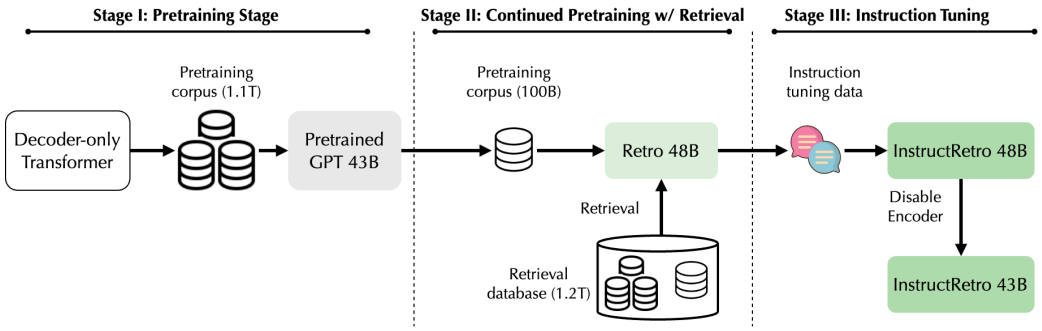


Figure 1. Training pipeline for InstructRetro 48B and InstructRetro 43B.

sive language models (e.g., ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023)) motivates us to further scale up autoregressive Retro and understand the potential benifit of retrieval-augmented pretraining.

Retro encoder is a shallow bidirectional transformer to encode retrieved neighbors from external databases into dense features. Specifically, in this work, we follow Borgeaud et al. (2022) and use a two-layer bidirectional transformer as the Retro encoder with the same hidden dimension as the Retro backbone decoder. Our preliminary results show that increasing the layers of the Retro encoder does not bring better perplexity on the validation set, but only increases the computational overhead and model parameters.

Retrieval database. Borgeaud et al. (2022) demonstrates that retrieval-augmented pretraining can significantly benefit from large-scale retrieval up to trillions of tokens. To build the retrieval database, we utilize the entire pretraining corpus, but holding out 1% as a validation set. This ensures that both Retro and GPT models are pretrained on an equivalent volume of information from the pretraining corpus. Our retrieval database is a key-value database, where values are chunks of tokens split from the pretraining corpus, and the keys are corresponding BERT embeddings (Devlin et al., 2018). The pretraining corpus consists of 1.2 trillion tokens of English corpus. More details of the pretraining corpus can be found in Appendix §A.1. In summary, our retrieval database comprises 19 billion chunks, with each chunk containing 64 tokens.

Chunk-wise cross-attention. Aligning with the chunk-wise design of the retrieval database, Retro splits the input tokens into a sequence of chunks. Specifically, Retro retrieves nearest neighbor chunks using the previous chunk and fuses this information with the context from preceding chunks to guide the generation of the next chunk. Formally, given a input sequence X with n tokens $X = (x_1, ..., x_n)$, Retro splits X into a sequence of l chunks $(C_1, ..., C_l)$ with chunk size $m = \frac{n}{l}$. From a high-level perspective, Retro uses the last $(i-1)$-th chunk C_{i-1} to retrieve k nearest neighbor chunks $\mathcal{N}(C_{i-1})$ from the retrieval database, and fuses the contextual information from the previous chunks $(C_1, ..., C_{i-1})$ and retrieval information from $\mathcal{N}(C_{i-1})$ by cross-attention to guide the generation of the next (i)-th chunk C_i. To avoid breaking the causality, the autoregressive generation of i-th chunk C_i can only use the nearest neighbors of the previous chunk $\mathcal{N}(C_{i-1})$ instead of $\mathcal{N}(C_i)$. In our work, we follow Borgeaud et al. (2022) and retrieve top-$k = 2$ nearest neighbors for each chunk, with chunk size $m = 64$ and the maximum number of tokens $n = 4096$.

3.2. Retro-fitting: continued pretraining with retrieval

There are two main challenges of scaling up Retro: the large-scale retrieval database and the pretraining cost of LLMs. To overcome the challenges, we leverage the *Faiss* index (Johnson et al., 2019) to achieve fast approximate nearest neighbor search and *retro-fitting* technique to reuse the pretrained GPT parameters and save computational cost.

Retrieval index to the large-scale retrieval database. We use the Faiss index (Johnson et al., 2019) as the implementation for the dense retriever to search for approximate nearest neighbors in the BERT embedding space. We configure the Faiss index to cluster the dense embeddings into 2^{22} centroids accelerated with Hierarchical Navigable Small World (HNSW) graphs (Malkov & Yashunin, 2018) to speed up the query. We also encode the embeddings with optimized product quantization (Gray & Neuhoff, 1998; Ge et al., 2014) to compress memory overhead and further improve the query throughput. As a result, we can achieve 4*ms* per query over the whole pretraining corpus averaged for each chunk on a DGX-A100 node. One may find more details in Appendix §B.

Base pretrained GPT. We launch continued pretraining (*i.e.*, GPT-fitting and Retro-fitting) based on pretrained GPT models. Specifically, we pretrain from scratch a set of GPT models with the following parameter sizes: 823M, 2.25B, 8.5B, 22B, and 43B. All of the models are based on Transformer (Vaswani et al., 2017) with different hidden dimen-

sions, number of layers, and attention heads. We adopt the Sentence Piece tokenizer (Kudo & Richardson, 2018) for both GPT and Retro. We pretrain all models with 1.1 trillion tokens of the pretraining corpus. More details of corpus be found in Appendix §A.1.

Unfreezing decoder at Retro-fitting. As Retro shares its backbone decoder with the GPT decoder and only adds around 10% additional parameters for Retro encoder and cross-attention, we can initialize Retro decoder from pretrained GPT models, randomly initialize Retro encoder and cross-attention, and continue pretraining with retrieval, which is named as "*Retro-fitting*". Note that, Borgeaud et al. (2022) freezes the decoder parameters at Retro-fitting. In contrast, we **unfreeze all the decoder parameters** and continue pretraining the entire model. We also conduct an ablation study of Retro-fitting based on a pretraiend GPT of 823M parameters and compare the validation perplexity loss when freezing or unfreezing Retro decoder during pretraining. As shown in Figure 3, given the same training schedules, unfreezing Retro decoder parameters converges faster and demonstrates better validation perplexity, which eventually yields a better Retro decoder to incorporate in-context retrieved evidence, even without a Retro encoder as shown in §5.4. We continue pretraining with retrieval on an additional 100 billion tokens, which is 9% of the pretraining data used for pretrained GPT models. To have a fair comparison, we also continue pretraining GPT foundation models on the same 100 billion tokens, which we name "*GPT-fitting*". In terms of overall pretraining cost, Retro 48B only need 2.58% additional GPU hours than its counterpart GPT trained on 1.2T tokens. More details of continued pretraining are in Appendix A.2 and A.3.

Perplexity evaluation. We evaluate the perplexity of GPT foundation models, GPT-fitting models, and Retro-fitting models of varying parameter sizes in Figure 2. The validation corpus consists of 1% held-out samples from the pretraining corpus, which are not used in the pretraining stage, the continued pretraining stage, and the retrieval database to ensure that there is no validation data leakage. From Figure 2, one can see that after continued pretraining on additional 100 billion tokens, the perplexity of GPT-fitting slightly improves over original pretrained GPT, while Retro significantly outperforms both GPT and GPT-fitting across different parameter sizes in terms of perplexity. Retro achieves even better perplexity than GPT models with 4× larger parameter sizes. Notably the improvement is still significant when the parameter sizes of Retro scale up to 48B, and the gap does not decrease from 8B to 48B. We present more evaluation results in §5.4.

4. Instruction Tuning

Instruction tuning can significantly improve the ability of foundation LLMs to follow instructions, thus improving zero-shot results on downstream tasks (e.g., Wei et al., 2022a; Chung et al., 2022). In this section, we further enhance Retro via instruction tuning.

4.1. Datasets Blending

Existing instruction tuning methods mainly leverage supervised fine-tuning on a blend of instruction following datasets (Wei et al., 2022a; Chung et al., 2022; Sanh et al., 2022a; Wang et al., 2023b).

We use a blend of high-quality instruction tuning datasets to train LLMs to follow instructions in conversational formats, which include: *i)* a high-quality social dialogue dataset SODA (Kim et al., 2022), *ii)* a long-form QA dataset ELI5 that requires elaborate answers (Fan et al., 2019), *iii)* LLM-generated instructions: Self-Instruct (Wang et al., 2022b) and Unnatural Instructions (Honovich et al., 2022), *iv)* FLAN and Chain-of-thought datasets (Chung et al., 2022; Wei et al., 2022c; Longpre et al., 2023), *v*) a private crowdsourced conversational dataset and public human-written conversation datasets OpenAssistant (Köpf et al., 2023) and Dolly (Conover et al., 2023), and *vi*) samples from the pretraining corpus.

The format of all the instruction tuning data is unified in a conversational way with three roles: "system", "assistant", and "user". The "system" role sets up the tone and style of LLM assistants to give helpful, detailed, and polite answers to the user's questions. The "user" and "assistant" role contains the questions and the corresponding answers from the instruction tuning datasets. We show an example format of the instruction data in Appendix C.1. In total, we collect a total of 128K high-quality samples for instruction tuning.

4.2. Training details

For each training sample, we take the multi-turn conversations between the user and the assistant as context and apply the loss mask only to the last response from the assistant. We use the standard language modeling loss with teacher forcing. Since Wei et al. (2022a) suggests that instruction tuning is most effective with *large* language models, we apply instruction tuning to the GPT-fitting 43B model and the Retro 48B model, naming them "InstructGPT$_{\text{RAG}}$ 43B"[2] and "InstructRetro 48B", respectively. We finetune the LLMs by taking the loss only on the answer part with a batch size of 128 and a learning rate of 5e-6 for 1000 steps with a weight decay of 0.01. We use the Adam optimizer (Kingma & Ba,

[2]We distinguish "InstructGPT$_{\text{RAG}}$", which uses supervised fine-tuning and RAG, from "InstructGPT" (Ouyang et al., 2022), which leverage RLHF for instructing tuning.

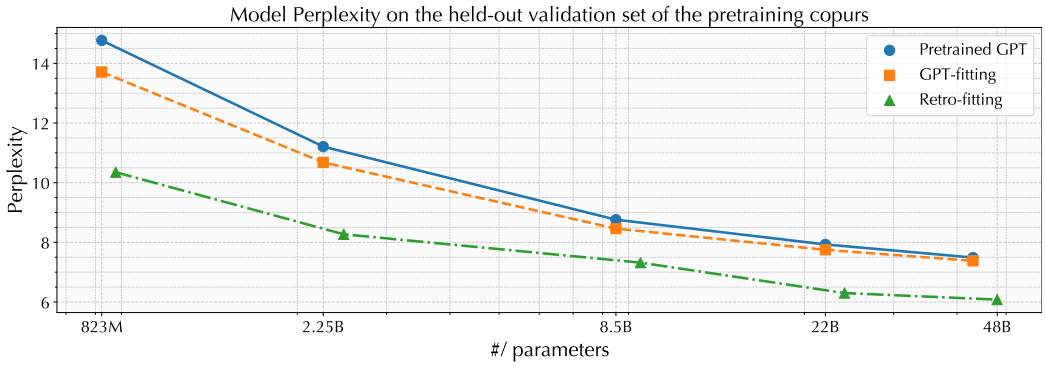


Figure 2. Perplexity evaluation of pretrained GPT models, GPT-fitting, and Retro-fitting models across various parameter sizes on the held-out validation set. In contrast to Borgeaud et al. (2022), we unfreeze all parameters for Retro-fitting. Retro significantly outperforms GPT models, achieving the perplexity comparable to GPT models with 4× larger parameter sizes.

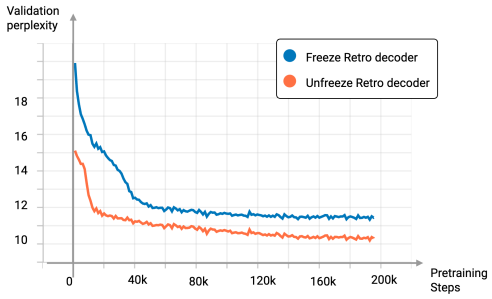


Figure 3. Validation perplexity of Retro-fitting (823M) when we freeze or unfreeze Retro decoder during continued pretraining on 100B tokens.

2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.98$.

Instruction tuning for Retro. Since the Retro backbone largely shares with GPT models, the training objective of Retro is also the same as GPT models. However, one noticeable difference is that Retro requires retrieval of nearest neighbors, which is not available from all the instruction tuning datasets. Since the instruction tuning data is high-quality, retrieval from the pretraining corpus can yield noisy neighbors, thus not helping improve the model capabilities to follow instructions. We instead disable the Retro encoder by skipping the cross-attention connection through a manually-set gated mechanism as detailed in Figure 4, which sets the gate to *zero* when retrieved neighbors are not available. During backpropagation, as the cross-attention module and the connected retro encoder are skipped, their parameters are effectively frozen, and only the weights of the decoder backbone get updated. Such design not only simplifies the instruction tuning and inference but also makes Retro learn to inference with and without retrieval during instruction tuning, potentially improving the generalization of the Retro decoder. We also leave it as an important future direction to construct retrieval-augmented instruction tuning data for retrieval-augmented generation.

5. Experiments

In this section, we conduct comprehensive studies on the zero-shot capabilities of *InstructRetro* and its GPT counterpart with RAG (InstructGPT$_{\text{RAG}}$) across various downstream tasks to unveil the potential of Retro model after instruction tuning.

5.1. Experimental setup

Datasets. To demonstrate the generalization of instruction tuning, we follow FLAN (Wei et al., 2022a) and primarily focus on *zero-shot evaluation* of downstream tasks. Specifically, we consider two categories of open-ended QA tasks as well as text summarization tasks: (1) *short-form QA or reading comprehension*, which expects short answers (e.g., a few tokens) to be generated or extracted from the context, including Natural Question (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), NewsQA (Trischler et al., 2016), SQuAD 1.1 (Rajpurkar et al., 2016), SQuAD 2.0 (Rajpurkar et al., 2018), Quoref (Dasigi et al., 2019), NarrativeQA (Kočiskỳ et al., 2018), DROP (Dua et al., 2019). To compare with baselines, we use the split from KILT benchmark (Petroni et al., 2021) for NQ and TriviaQA. For the other tasks, we use the official splits; (2) *long-form QA*, which expects longer answer spans within a few sentences, including doc2dial (Feng et al., 2020), two proprietary annotated car manual datasets (people ask questions about the particular car models), and another proprietary annotated IT documentation dataset; (3) *summarization*, which expects to summarize a long passage or context within a few sentences, including QMSum (Zhong et al., 2021), SummScreenFd (Chen et al., 2021), and GovReport (Huang et al., 2021).

Retrieval-augmented generations (RAG). At pretraining, we use BERT embeddings to embed the retrieval database and support retrieval from trillions of tokens. For downstream task evaluation, we follow Retro (Borgeaud et al., 2022) and use task-specific corpus and state-of-

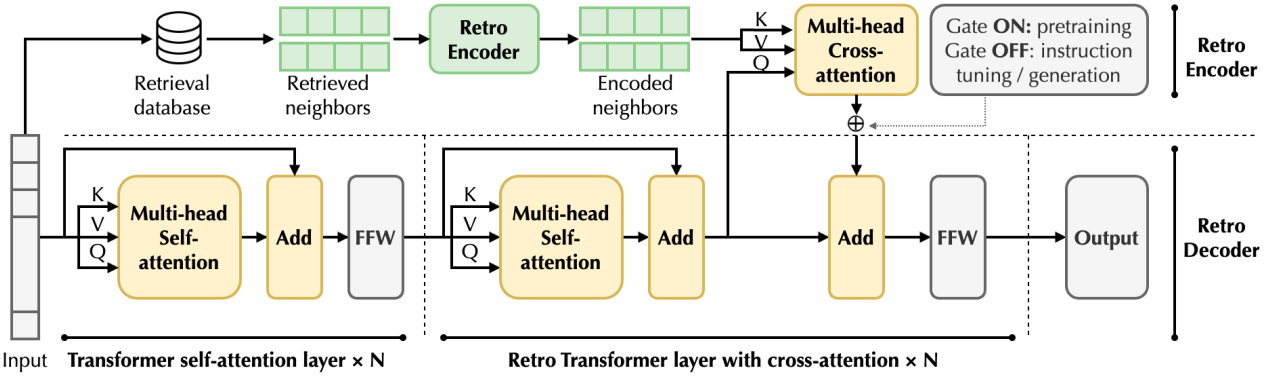


Figure 4. Simplified architecture diagram of InstructRetro. We omit the layer norm, softmax, and embedding layers for simplicity. We add additional 0/1 gates between cross-attention output and the residual connection from self-attention output. During pretraining, we keep the Retro encoder gate ON with gate-value 1. During instruction tuning and inference, we bypass the Retro encoder by turning the Retro encoder gate OFF with gate-value 0 to solely serve as a GPT decoder.

the-art retrievers to retrieve the most relevant and high-quality information for the task. Specifically, for NQ, TriviaQA, doc2dial, and other long-form QA datasets, we use DRAGON+ (Lin et al., 2023) as the retriever. We retrieve the top-$k = 5$ nearest neighbors and concatenate them in the prompt. For the remaining QA and summarization tasks, we use the provided contexts in the datasets. An example of how we format the retrieved neighbors in the prompt is shown in Appendix Table 10.

Models. InstructGPT$_{\text{RAG}}$ 43B is our main baseline as it has the same decoder hyper-parameters as InstructRetro 48B (e.g., number of transformer layers, hidden sizes, etc.) and was pretrained and instruction-tuned on the same datasets. We also compare them with InstructRetro 43B, which is the derivative of InstructRetro 48B by turning off the gate of cross-attention (*i.e.*, bypassing the encoder). Note that we do instruction tuning on Retro without enabling its encoder. When we bypass the encoder during evaluation, **InstructRetro 43B solely serves as a GPT decoder** to align with the instruction tuning behaviors and simplify the inference.

For additional baselines, we also compare a wide range of state-of-the-art LLMs with comparable or larger sizes, including GPT-3 175B (Brown et al., 2020b), , GLaM 64B (Du et al., 2021), FLAN-LaMDA 137B (Wei et al., 2022a), and Llama 2 70B (Touvron et al., 2023b) with RAG[3]. Furthermore, we also compare InstructRetro with existing retrieval-augmented LLMs, including Retro 7.5B (Borgeaud et al., 2022), Retro++ 9B (Wang et al., 2023a), Atlas 11B (Izacard et al., 2022a), and Raven 11B (Huang et al., 2023).

Other details. We use greedy decoding with the max output length to be 256. We truncate the generation when we encounter the special token `|<end-of-document>|` or role-switching from "Assistant" to "User" when completing the conversation. All of the QA tasks are re-formatted in the conversational format. An example from the SQuAD 1.1 dataset in the conversational prompt format is shown in Appendix Table 10.

[3]We evaluate both Llama 2 70B text model and instruction-tuned chat model with RAG, and report the best numbers.

5.2. Zero-shot evaluation on QA tasks

We present the zero-shot evaluation results across eight short-form QA and reading comprehension datasets in Table 1. We also apply InstructRetro to four open-ended long-form QA datasets, as detailed in Table 2. These datasets are representative of real-world applications, including chatbots for IT support and customer service.

Instruction tuning post retrieval-augmented pretraining yields a better GPT decoder. From Table 1, we observe that InstructRetro 43B shows consistent accuracy improvement upon its counterpart InstructGPT$_{\text{RAG}}$ 43B across different datasets for short-form QA or reading comprehension tasks. Notably, the average relative improvement of InstructRetro across all the short-form datasets is around 7%. Given that both InstructRetro 43B and InstructGPT$_{\text{RAG}}$ 43B are pretrained and instruction tuned with identical datasets, hyper-parameters, and evaluation prompts, we attribute this consistent improvement to the training recipe of InstructRetro, which leverages continued pretraining with retrieval before instruction tuning. We hypothesize that retrieval-augmented pretraining enhances the capability of LLMs to utilize the information within the context (from both Retro encoder and decoder). The subsequent phase of instruction tuning further amplifies the effectiveness of InstructRetro in solving knowledge-intensive tasks. To have a deeper understanding, we provide an ablation study in §5.4.

From Table 1, we also show that InstructRetro 43B provides compelling performance than other state-of-the-art LLMs.

Table 1. Zero-shot evaluation on eight short-form QA and reading comprehension datasets. The average *relative* improvement of decoder-only InstructRetro 43B across the short-form QA tasks is 7% over InstructGPT$_{\text{RAG}}$. Note that, InstructRetro 43B obtains very comparable results than original InstructRetro 48B with encoder. denotes using retrieval augmentation at both training and generation, while denotes using retrieval augmentation at inference only.

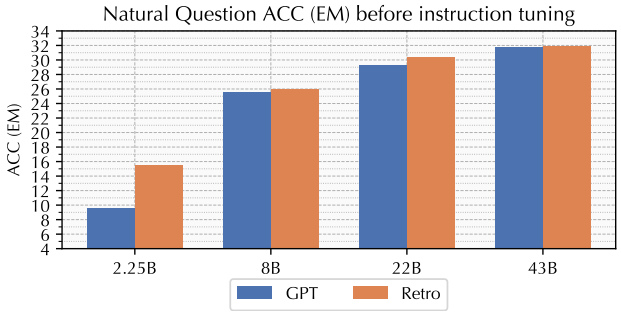
Task	NQ	TriviaQA	NewsQA	SQuAD 2.0	SQuAD 1.1	Quoref	NarrativeQA	DROP
Metric	EM	EM	F1	F1 / EM	F1 / EM	F1	F1	F1
GPT-3 175B (Brown et al.)	14.6	64.3	-	59.5 / 52.6	-	-	-	23.6
PaLM 2 -L (Chowdhery et al.)	37.5	-	-	- / -	-	-	-	-
GLaM 64B (Du et al.)	24.7	71.3	-	71.1 / 64.7	- / -	-	-	**57.3**
FLAN-LaMDA 137B (Wei et al.)	20.7	68.1	-	44.2 / -	**80.1** / -	-	-	22.7
Llama 2 RAG 70B (Touvron et al.)	37.7	65.6	53.4	71.4 / 64.1	73.4 / 66.2	69.7	52.7	57.2
Retro 7.5B (Borgeaud et al.)	8.9	36.0	-	- / -	-	-	-	-
Retro++ 9B (Wang et al.)	25.8	48.3	-	- / -	-	-	-	-
Atlas 11B (Izacard et al.)	26.7	56.9	-	- / -	- / -	-	-	-
Raven 11B (Huang et al.)	29.6	65.7	-	- / -	- / -	-	-	-
RA-DIT 65B (Lin et al.)	35.2	75.4	-	-	- / -	-	-	-
InstructGPT$_{\text{RAG}}$ 43B	37.0	78.1	52.4	70.7 / 64.3	72.4 / 65.8	71.5	53.9	51.8
InstructRetro 43B (w/o encoder, Avg: +7%)	**38.9** (+5.14%)	**78.3** (+0.26%)	**57.4** (+9.54%)	**75.6 / 69.3** (+6.93%)	77.1 / 70.4 (+6.49%)	**76.2** (+6.57%)	**60.0** (+11.32%)	54.8 (+5.79%)
InstructRetro 48B (w/ encoder, Avg: +6%)	38.6 (+4.32%)	77.8 (-0.38%)	57.0 (+8.78%)	74.8 / 67.7 (+5.80%)	76.4 / 69.0 (+5.52%)	76.1 (+6.43%)	59.8 (+10.95%)	54.6 (+5.41%)

For example, InstructRetro 43B achieves better accuracy than Llama 2 with RAG on multiple tasks, close to FLAN-LaMDA 137B, which is 3× the size of InstructRetro 43B.

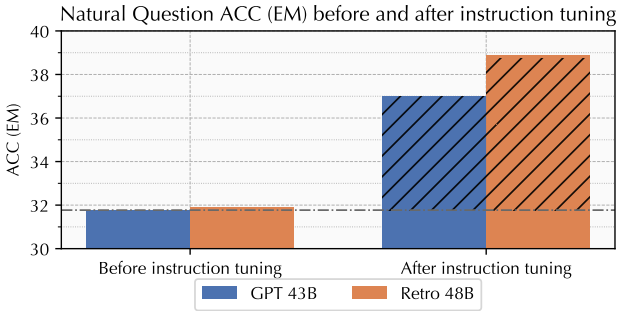
Impact of Retro encoder for downstream tasks. We also notice that InstructRetro 48B and 43B perform very comparable from Table 1. We enable the Retro encoder for retrieval-augmented pretraining, while disabling the Retro encoder due to the lack of retrieved high-quality neighbors for instruction tuning. Note that we still perform retrieval-augmented generation for downstream tasks, where the retrieved contexts are put into the decoder of both InstructRetro 48B and 43B as part of the prompts. The only difference is whether we enable the cross attention gate in Figure 4 to attend the Retro encoder in InstructRetro 48B or disable it in InstructRetro 43B. When enabling the Retro encoder, we put the top-2 neighbors in the encoder to align with the pretraining behavior.

This suggests that although Retro is proficiently trained to infer both with and without the neighbors in the encoder, it is more beneficial to align with the instruction tuning protocols and bypass the Retro encoder to solely serve as a GPT decoder during evaluation. We think it is an important and promising future research direction to explore retrieval-augmented instruction tuning with the Retro encoder activated, especially when high-quality retrieval-augmented instruction tuning data is available.

InstructRetro demonstrates larger improvement on long-form QA datasets. When comparing the results of InstructRetro on short-form QA datasets and long-form QA datasets, we observe InstructRetro 43B demonstrates large



(a) Before instruction tuning, the improvement of retrieval augmentation saturates when the size scales up.



(b) Instruction tuning further unveils the potential of retrieval augmentation even when the size scales up.

Figure 5. Zero-shot accuracy (EM) of GPT and Retro before and after instruction tuning evaluated on the Natural Question dataset.

Table 2. Zero-shot evaluation on four long-form QA datasets. We use F1 as the evaluation metric. Car #1 and #2 are short for two annotated car manual datasets. The average *relative* improvement of InstructRetro 43B across the long-form QA tasks is 10% over InstructGPT$_{\text{RAG}}$ 43B.

		doc2dial	Car #1	Car #2	IT Doc
Llama 2 RAG	70B	32.33	49.63	45.89	25.70
InstructGPT$_{\text{RAG}}$	43B	32.87	58.18	50.88	31.40
InstructRetro	43B	35.74	**63.52**	**57.49**	**34.08**
(w/o encoder, Avg: +10%)		(+8.73%)	(+9.18%)	(+12.99%)	(+8.54%)
InstructRetro	48B	**35.95**	63.16	56.82	34.07
(w/ encoder, Avg: +10%)		(+9.37%)	(+8.56%)	(+11.67%)	(+8.50%)

Table 3. Zero-shot evaluation on three summarization datasets. We use the standard ROUGE scores as the evaluation metrics. The average *relative* improvement of InstructRetro 43B across the long-form QA tasks is 16% over InstructGPT$_{\text{RAG}}$ 43B.

		GovReport	SummFD	QMSum
Llama 2 RAG	70B	16.98	10.02	14.50
InstructGPT$_{\text{RAG}}$	43B	12.59	10.43	15.06
InstructRetro	43B	**17.46**	**10.93**	**15.61**
(w/o encoder, Avg: +16%)		(+38.68%)	(+4.79%)	(+3.65%)

relative accuracy improvements, achieving 10% over the InstructGPT$_{\text{RAG}}$ 43B. As long-form QA tasks are generally more challenging than short-form QA tasks, such improvements further demonstrate the potential of retrieval-augmented pretraining. Again, the results of InstructRetro 43B and 48B results are very comparable, while InstructRetro 43B performs slightly better.

5.3. Zero-shot evaluation on summarization tasks

We also apply InstructRetro for summarization tasks, including QMSum (Zhong et al., 2021), SummScreenFD (Chen et al., 2021), and GovReport (Huang et al., 2021). Following the official metrics, we report the geometric mean of ROUGE scores (*i.e.*, ROUGE1/2/L) for these summarization tasks. The zero-shot evaluation results are shown in Table 3. From Table 3, we observe that InstructRetro consistently outperforms the InstructGPT$_{\text{RAG}}$ on these summarization tasks, especially on the GovReport dataset with 4.87 ROUGE score improvement. Moreover, InstructRetro 43B consistently outperforms Llama 2 RAG 70B across three datasets. This experiment further confirms the generalizability of InstructRetro after instruction tuning and indicates that instruction tuning post retrieval-augmented pretraining yields a better GPT decoder.

5.4. Ablation stuides

In this section, we conduct ablation studies to understand the source of improvements for InstructRetro. We show that both retrieval-augmented pretraining and instruction tuning are indispensable to unlock the potential of retrieval-augmented LLMs.

To understand how instruction tuning improves retrieval-augmented pretraining, we show the zero-shot accuracy (Exact Match score) of Retro and GPT on the Natural Question dataset before and after instruction tuning, as detailed in Figure 5. We observe that Retro achieves significantly better zero-shot accuracy than GPT when the number of parameters is relatively small (e.g., 2.25B). However, when scaling the size of parameters, the zero-shot performances of both GPT and Retro start to saturate. We hypothesize that this saturation is mainly due to the poor instruction following capability of both pretrained foundation GPT and Retro models.

To remove the instruction-following bottleneck, we apply instruction tuning to further fine-tune both Retro 48B and GPT 43B. Instruction tuning largely mitigate the instruction following bottleneck for both GPT and Retro, resulting in a significant increase of their zero-shot performance on downstream tasks, respectively. Furthermore, once this bottleneck is alleviated, the benefits of retrieval augmentation at pretraining become more pronounced, as InstructRetro excels in leveraging and integrating evidence from retrieved context. Thus, we observe significant improvement

of InstructRetro over InstructGPT$_{\text{RAG}}$ again in Figure 5b. This ablation study confirms that our training recipe - both retrieval-augmented pretraining and instruction tuning are important for achieving high performance in QA tasks.

5.5. Evaluation on MT-Bench

We further evaluate InstructRetro and InstructGPT on the MT-Bench chat benchmark (Zheng et al., 2024) to access Retro's performance on general chat tasks. One may find more details in Appendix D.

6. Conclusion

In this paper, we introduce InstructRetro 48B, the largest LLM with retrieval-augmented pretraining and instruction tuning. Specifically, we start from a pretrained GPT model, and continue to pretrain the model with retrieval, which yields the retrieval-augmented foundation model Retro 48B. After applying instruction tuning to Retro, InstructRetro 48B unveils the potential of retrieval-augmented pretraining and demonstrates significant zero-shot accuracy improvement over its GPT counterpart through our extensive experiments on a wide range of downstream tasks. Moreover, our novel findings show that only using the GPT decoder backbone, i.e., InstructRetro 43B, can achieve comparable accuracy, which sheds light on a promising direction to obtain a better GPT decoder through retrieval-augmented pretraining before instruction tuning.

Impact Statement

Our InstructRetro, similar to the line of RAG studies, offers significant advancements in addressing the practical deployment and applications of LLMs, particularly in areas of factuality, downstream task accuracy, contextual understanding, and model efficiency. Specifically, it significantly improves the generation of factual and grounded text with retrieval from high-quality databases, which helps mitigate misinformation and enhance public trust. Additionally, InstructRetro boosts the accuracy of LLMs in various downstream tasks and demonstrates better capability in contextual understanding, which is important for reasoning tasks. Remarkably, it achieves comparable performance to LLMs two to three times its size, enhancing computational efficiency and environmental sustainability. This positions InstructRetro as a practical tool in the ethical and safe deployment of LLMs.

References

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *NeurIPS*, 2020a.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020b.

Chen, M., Chu, Z., Wiseman, S., and Gimpel, K. Summscreen: A dataset for abstractive screenplay summarization. *ArXiv*, abs/2104.07091, 2021. URL https://api.semanticscholar.org/CorpusID:233240744.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models. *arXiv preprint arXiv: 2210.11416*, 2022. URL https://arxiv.org/abs/2210.11416v5.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm. *databricks*, 2023.

Dasigi, P., Liu, N. F., Marasović, A., Smith, N. A., and Gardner, M. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. *Conference on Empirical Methods in Natural Language Processing*, 2019. doi: 10.18653/v1/D19-1606.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A., Firat, O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q. V., Wu, Y., Chen, Z., and Cui, C. Glam: Efficient scaling of language models with mixture-of-experts. *International Conference on Machine Learning*, 2021. URL https://arxiv.org/abs/2112.06905v2.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *North American Chapter of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/N19-1246.

Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., and Auli, M. Eli5: Long form question answering. *Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/P19-1346.

Feng, S., Wan, H., Gunasekara, R. C., Patel, S., Joshi, S., and Lastras, L. doc2dial: A goal-oriented document-grounded dialogue dataset. *Conference on Empirical Methods in Natural Language Processing*, 2020. doi: 10.18653/v1/2020.emnlp-main.652.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Ge, T., He, K., Ke, Q., and Sun, J. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014. doi: 10.1109/TPAMI.2013.240.

Gray, R. and Neuhoff, D. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. doi: 10.1109/18.720541.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. REALM: Retrieval augmented language model pre-training. In *ICML*, 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Honovich, O., Scialom, T., Levy, O., and Schick, T. Unnatural instructions: Tuning language models with (almost) no human labor. *Annual Meeting of the Association for Computational Linguistics*, 2022. doi: 10.48550/arXiv.2212.09689.

Huang, J., Ping, W., Xu, P., Shoeybi, M., Chang, K. C.-C., and Catanzaro, B. Raven: In-context learning with retrieval augmented encoder-decoder language models. *arXiv preprint arXiv:2308.07922*, 2023.

Huang, L. R., Cao, S., Parulian, N. N., Ji, H., and Wang, L. Efficient attentions for long document summarization. *ArXiv*, abs/2104.02112, 2021. URL https://api.semanticscholar.org/CorpusID:233033613.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Atlas: Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv: 2208.03299*, 2022a.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022b.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7 (3):535–547, 2019.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Annual Meeting of the Association for Computational Linguistics*, 2017. doi: 10.18653/v1/P17-1147.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Generalization through memorization: Nearest neighbor language models. 2020.

Kim, H., Hessel, J., Jiang, L., West, P., Lu, X., Yu, Y., Zhou, P., Bras, R. L., Alikhani, M., Kim, G., Sap, M., and Choi, Y. Soda: Million-scale dialogue distillation with social commonsense contextualization. *arXiv preprint arXiv: 2212.10465*, 2022. URL https://arxiv.org/abs/2212.10465v2.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

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

Kudo, T. and Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *Conference on Empirical Methods in Natural Language Processing*, 2018. doi: 10.18653/v1/D18-2012.

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

2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., ES, S., Suri, S., Glushkov, D., Dantuluri, A., Maguire, A., Schuhmann, C., Nguyen, H., and Mattick, A. Openassistant conversations - democratizing large language model alignment. *arXiv preprint arXiv: 2304.07327*, 2023.

Laurençon, H., Saulnier, L., Wang, T., Akiki, C., Villanova del Moral, A., Le Scao, T., Von Werra, L., Mou, C., González Ponferrada, E., Nguyen, H., et al. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Advances in Neural Information Processing Systems*, 35:31809–31826, 2022.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

Lin, S.-C., Asai, A., Li, M., Oguz, B., Lin, J., Mehdad, Y., tau Yih, W., and Chen, X. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. *arXiv preprint arXiv: 2302.07452*, 2023.

Lin, X. V., Chen, X., Chen, M., Shi, W., Lomeli, M., James, R., Rodriguez, P., Kahn, J., Szilvasy, G., Lewis, M., Zettlemoyer, L., and tau Yih, W. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=22OTbutug9.

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., and Roberts, A. The flan collection: Designing data and methods for effective instruction tuning. *International Conference on Machine Learning*, 2023. doi: 10.48550/arXiv.2301.13688.

Malkov, Y. A. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.

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

OpenAI. ChatGPT. https://chat.openai.com, 2022.

OpenAI. GPT-4 technical report. *arXiv*, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022.

Petroni, F., Piktus, A., Fan, A., Lewis, P., Yazdani, M., De Cao, N., Thorne, J., Jernite, Y., Karpukhin, V., Maillard, J., Plachouras, V., Rocktäschel, T., and Riedel, S. KILT: a benchmark for knowledge intensive language tasks. In *NAACL*, 2021.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *Conference on Empirical Methods in Natural Language Processing*, 2016. doi: 10.18653/v1/D16-1264.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. *Annual Meeting of the Association for Computational Linguistics*, 2018. doi: 10.18653/v1/P18-2124.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. In *ICLR*, 2022b.

Shi, W., Min, S., Lomeli, M., Zhou, C., Li, M., Lin, V., Smith, N. A., Zettlemoyer, L., Yih, S., and Lewis, M. In-context pretraining: Language modeling beyond document boundaries. *arXiv preprint arXiv:2310.10638*, 2023a.

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., and Yih, W.-t. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023b.

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., Zhang, E., Child, R., Aminabadi, R. Y., Bernauer, J., Song, X., Shoeybi, M., He, Y., Houston, M., Tiwary, S., and Catanzaro, B. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv*, 2022.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *ARXIV*, 2023a. URL https://arxiv.org/abs/2302.13971v1.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*, 2023b. URL https://arxiv.org/abs/2307.09288v2.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. Newsqa: A machine comprehension dataset. *REP4NLP@ACL*, 2016. doi: 10.18653/v1/W17-2623.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NIPS*, 2017.

Wang, B., Ping, W., Xu, P., McAfee, L., Liu, Z., Shoeybi, M., Dong, Y., Kuchaiev, O., Li, B., Xiao, C., et al. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. In *EMNLP*, 2023a.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022a.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. *Annual Meeting of the Association for Computational Linguistics*, 2022b. doi: 10.48550/arXiv.2212.10560.

Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K. R., Wadden, D., MacMillan, K., Smith, N. A., Beltagy, I., and Hajishirzi, H. How far can camels go? exploring the state of instruction tuning on open resources. *arXiv preprint arXiv: 2306.04751*, 2023b. URL https://arxiv.org/abs/2306.04751v1.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *ICLR*, 2022a.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022c.

Yogatama, D., de Masson d'Autume, C., and Kong, L. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 2021.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*, 2021.

A. Details of Pretraining

A.1. Pretraining corpus

We prepared a pretraining dataset consisting of around 1.2 trillion tokens from English natural language data. Specifically, it consists of web-crawl data from Common Crawl, news data, conversational data, book data (e.g., Book3 and Book-Corpus2 from the Pile dataset (Gao et al., 2020)), scientific and multi-domain data (e.g., Wikipedia and the BigScience ROOTS corpus (Laurençon et al., 2022)).

A.2. Continued pretraining schedules

Based on pretrained GPT models, we further pretrain Retro with retrieval augmentation on additional 100 billion tokens, which is around 25M samples with sequence length set to 4096. We list the pretraining hyper-parameter details of Retro-fitting in Table 4. GPT-fitting uses the same training schedules as Retro-fitting.

All models use Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We employ the learning rate (LR) decay schedules with LR warmup samples of 16667 and LR decay samples of 23750000.

Table 4. Detailed pretraining setup for standard pre-trained LMs and InstructRetro.

Models Size	LR	min LR	LR Decay Styles	Batch Size	Pretraining Steps
823M	2e-5	2e-6	cosine	128	195.2k
2.25B	2e-5	2e-6	cosine	256	97.6k
8.5B	1e-5	1e-6	cosine	512	48.8K
22B	1e-5	1e-6	cosine	512	48.8K
43B	9e-6	9e-7	cosine	768	32.5k

A.3. Computational cost for continued pretraining

We present the detailed computational cost of the continued pretraining step on additional 100B tokens for both Retro and GPT across different sizes in Table 5. We can see that pretraining Retro brings around additional 35% computational overhead than pretraining GPT, which mainly comes from the Retro encoder and cross-chunk attention to incorporate and fuse the retrieved neighbor information. Moreover, we can see that scaling up the size of Retro does not bring more computational overhead and remains around 35%, shedding light on a promising way to retrieval-augmented pretraining.

A more useful perspective is looking at the overall pretraining cost. Since our Retro 48B starts from a pretrained GPT 43B on 1.1T tokens, it only need 2.58% additional GPU hours in contrast to pretraining the GPT 43B on 1.2T tokens.

$$\frac{1.1\text{T} \times 1 + 0.1\text{T} \times (1 + 31\%)}{1.2\text{T} \times 1} = 102.58\%$$

Table 5. Pretraining cost of the continued pretraining on 100B tokens for Retro and GPT across different sizes.

GPT	training on 100B token	Retro	training on 100B token	Additional Overhead (on 100B tokens)	Additional Overall Overhead (on 1.2T tokens)
823M	1408 GPU Hours	878M	1920 GPU Hours	36%	3.00%
2.25B	3226 GPU Hours	2.5B	4096 GPU Hours	27%	2.25%
8.5B	12698 GPU Hours	9.5B	17325 GPU Hours	37%	3.08%
22B	37888 GPU Hours	24B	52152 GPU Hours	37%	3.08%
43B	53329 GPU Hours	48B	69995 GPU Hours	31%	2.58%

B. Details of retrieval database

Retrieval Database. We use the whole pretraining corpus as our retrieval database, consisting of 1.2 trillion tokens as mentioned in Appendix §A.1. Our pretraining dataset with 1.2 trillion tokens yields a retrieval database consisting of 19B chunks in total with chunk size $m = 64$. To support fast similarity searches with billions of chunks, we implement the database index with Faiss index (Johnson et al., 2019). Given the BERT embeddings of an input chunk C_i, Faiss can return the approximate k nearest neighbor of C_i within a few milliseconds.

B.1. Faiss index configuration

We use the Faiss index (Johnson et al., 2019) as the implementation for the dense retriever to search for approximate nearest neighbors in the BERT embedding space. We configure the Faiss index as follows:

- **Preprocessing**: We use Optimized Product Quantization (Ge et al., 2014) to apply a rotation to the input vectors to make them more amenable to PQ coding (Gray & Neuhoff, 1998).
- **Indexer**: We use Inverted File Index (IVF) with 2^{22} centroids and accelerate it with Hierarchical Navigable Small World (HNSW) graphs (Malkov & Yashunin, 2018).
- **Encoding**: We adopt PQ encoding that compresses the dense embedding vector into 64 bits.

As a result, we can achieve *4ms* per query over the whole pretraining corpus via batch queries averaged for each chunk with less than 1TB memory usage as our max throughput. Given a single query, the latency of the response is around $0.1s$ per query. We also note that increasing the number of K in the query does not yield slower query speed. During pretraining, we follow Borgeaud et al. (2022) to pre-compute the nearest neighbors and save the data for pretraining.

B.2. Computational cost on building retrieval database

Building a Faiss index involves several steps. We detail each step with its associated computational cost as below:

- **Embedding the retrieval database into dense BERT embeddings.** Given the chunk size of $m = 64$ tokens, we embed every chunk of text corpus with BERT-large-cased. The computational cost to embed the text corpus is around 6.22M chunks per GPU hour given one A100 GPU. For our 19B chunk database, it takes around 3054 GPU hours in total.
- **Train the Faiss index.** This involves determining a smaller number of centroids to cluster the whole corpus embeddings and initializing the HNSW graph. The computational cost of training the Faiss index depends on the number of corpus embeddings and the number of centroids. Given our setup, we train the faiss index based on 600M chunks uniformly sampled from the whole retrieval database. The computational cost of this step is less than 4 hours with one DGX A100 node.
- **Add the embedded corpus to the Faiss index.** After the index has been trained, the index centroids and HNSW graph are determined, but the index itself is still empty. In this step, we add the whole dense corpus embeddings to the index data structure. The computational cost of adding the corpus to the index is around 192 CPU hours within one DGX A100 node. Moreover, it can be purely done within a CPU node to save computational cost.
- **Query the Faiss index.** As mentioned above, we can achieve *4ms* per query over the whole pretraining corpus via batch queries averaged for each chunk with less than 1TB memory usage as our max throughput. The computational cost to query over 100B tokens in our continued pretraining step is around 1736 CPU hours within a DGX A100 node. Moreover, this step can also be purely done within a CPU node to save computational cost and can run in parallel to further speed up the querying.

In summary, the overall computational cost of building Faiss index is marginal compared to the pretraining cost, especially considering the benefits of retrieval-augmentation pretraining, which further unlocks the potential of instruction tuning. Thus we believe that it is a promising direction to pretrain with retrieval augmentation.

B.3. Ablation studies on Faiss index confirations

Faiss training-time configuration. We conduct ablation studies on the quantization techniques using two index configurations on two datasets: the whole pretraining dataset and the Wikipedia Corpus. We highlight the configuration setup in Table 6 below.

Following the official guide of Faiss[4], we initialize two Faiss indexes based on the sizes of two retrieval databases: the full pretraining corpus with 19B chunks and the Wikipedia corpus with 66M chunks. We applied product quantization (Ge et al., 2014; Gray & Neuhoff, 1998) to the full pretraining corpus to reduce the dimensionality and save the index memory to support loading the full pretraining corpus, while applying uncompressed flat encoding to the Wikipedia corpus as a comparison. We benchmark the querying speed for a batch of 40K dense embeddings and evaluate the query speed for two indexes.

[4] https://github.com/facebookresearch/faiss/wiki/Guidelines-to-choose-an-index

Table 6. Ablation studies on Faiss product quantization (PQ) on two different retrieval databases.

		Retrieval Index for Full Pretraining Corpus	Retrieval Index for Wikipedia Corpus
#/ chunks		19B	66M
Configuration	Dimension Reduction	OPQ64_128	No Reduction
	Approximate Search	IVF4194304_HNSW32	IVF262144_HNSW32
	Encoding	PQ64	Flat Encoding
Query Speed	K=2	0.004 s/query	0.01 s/query
	K=20	0.004 s/query	0.01 s/query
	K=200	0.0045 s/query	0.01 s/query
	K=2000	0.004 s/query	0.01 s/query

From Table 6, we can see that applying product quantization can not only help compress the index and save memory usage but also help improve the query speed, which is critical when scaling up the retrieval database. We can also see that increasing the number of K for K nearest neighbor searchers barely impacts the query speed.

Faiss query-time configuration. For our index configuration with interveted file index structures and HNSW graph, the hyper-parameter `nprobe` and `efSearch` play important roles in the query time of Faiss, as detailed in Table 7.

Table 7. Important querying-time hyper-parameters for our Faiss index.

index type	Index class	runtime parameter	comments
IVF*, IMI2x*	IndexIVF*	nprobe	the main parameter to adjust the speed-precision tradeoff
HNSW*	IndexHNSW	efSearch	the depth of the HNSW search

To select a proper set of query-time hyper-parameters with a good tradeoff of recall and speed, we conduct ablation studies with varying `nprobe` and `efSearch`. Specifically, we use the retrieval index built on the whole pretraining corpus, query the index with randomly sampled 10K chunks from the pretraining corpus, and evaluate the recall accuracy of retrieving the query chunk itself given top-$K = 2000$. The query time and corresponding recall accuracy with different hyper-parameters are shown in Table 8. We thus select `nprobe`=4096 and `efSearch`=32 in our retrieval-augmented pretraining, considering its fast querying time without sacrificing the recall accuracy significantly.

Table 8. Trade-off between speed and recall accuracy given different query-time hyper-parameters.

	Time (per query)	Acc (K=2000)
efSearch=256, nprobe=65536	21ms	1.00
efSearch=64, nprobe=16384	6ms	0.96
efSearch=32, nprobe=4096	**4ms**	**0.93**
efSearch=16, nprobe=4096	2.5ms	0.88

C. Qualitative examples

C.1. An example From the instruction tuning data

Table 9 below is a data example from the instruction tuning data.

Table 9. A sample conversation from our instruction tuning dataset. The black part serves as the context, while the black part is masked as the answer mask, on which the LM loss is computed.

Speaker	**Conversation**
System	This is a chat between a user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions based on the context. The assistant should also indicate when the answer cannot be found in the context.
User	My bicycle tire has a puncture and while I have a pump with me, I don't have a patch repair kit. Is there a way to keep the tire from running out of air until I can get to a bicycle shop?
Assistant	Yes, you could use duct tape to seal the puncture.
User	Can you give me a tutorial on how to use the duct tape to seal the puncture?
Assistant	Sure, here's a tutorial on how to use duct tape to seal a puncture in a tire: 1. Start by cleaning the area around the puncture with a damp cloth. 2. Cut a piece of duct tape large enough to cover the puncture, plus a few millimeters of extra. 3. Remove the backing from the duct tape, and press it firmly over the puncture. 4. Rub the tape firmly with your fingers to ensure a good seal. 5. You can also use a roller or a flat item like a ruler to make sure that the tape is pressed firmly against the puncture. 6. Trim away any excess tape, and your puncture should now be sealed.

C.2. An example From the downstream QA dataset: SQuAD 1.1

We re-format all of the QA tasks in the conversational format. An example from the SQuAD 1.1 dataset with the conversational format is shown in the Table 10 below.

Table 10. A sample conversation prompt template from the SQuAD 1.1 dataset.

Speaker	**Conversation**
System	System: This is a chat between a user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions based on the context. The assistant should also indicate when the answer cannot be found in the context.
User	title: , source: The pound-force has a metric counterpart, less commonly used than the newton: the kilogram-force (kgf) (sometimes kilopond), is the force exerted by standard gravity on one kilogram of mass. The kilogram-force leads to an alternate, but rarely used unit of mass: the metric slug (sometimes mug or hyl) is that mass that accelerates at 1 ms-2 when subjected to a force of 1 kgf. The kilogram-force is not a part of the modern SI system, and is generally deprecated; however it still sees use for some purposes as expressing aircraft weight, jet thrust, bicycle spoke tension, torque wrench settings and engine output torque. Other arcane units of force include the sthène, which is equivalent to 1000 N, and the kip, which is equivalent to 1000 lbf. Based on the above article, answer a question. What is the seldom used force unit equal to one thousand newtons?
Assistant	The answer is

D. Experimental results on MT Bench

We evaluate InstructRetro and InstructGPT on the MTBench chat benchmark (Zheng et al., 2024) to understand Retro performance on general chat tasks. We use the Tulu-v2 (Wang et al., 2023b) alignment dataset as the instruction tuning dataset to enhance the chat capabilities for both pretrained GPT model and Retro model. The detailed breakdown of the MT-Bench result is shown in Table 11.

Table 11. Performance comparison of MT-Bench models

MT-Bench	InstructRetro-Tulu-v2-43B	InstructGPT-Tulu-v2-43B
Writing	**8.85**	8.15
Roleplay	7.75	**7.80**
Reasoning	**5.40**	4.75
Math	**3.15**	2.35
Coding	3.40	**4.10**
Extraction	**6.80**	6.75
STEM	**8.58**	8.53
Humanities	**9.68**	9.10
Turn 1	**6.89**	6.67
Turn 2	**6.51**	6.21
Avg	**6.70**	6.44

In Table 11, we show that on average InstructRetro outperforms InstructGPT across different turns. We also observe that Retro performs slightly lower than GPT in domains such as role play and coding. We think that the main reason is that we do not cover coding and role-playing related datasets in the pretraining dataset of the base GPT model, and thus retrieval-augmented pretraining could make little difference.

E. Potential Negative Social Impacts

In this section, we discuss a few potential negative social impacts shared by the current line of LLM research. First, similar to other very capable LLMs, InstructRetro can generate non-factual but persuasive text across a wide range of topics. This ability can be maliciously exploited to create and spread disinformation or misinformation at scale. Second, InstructRetro is trained on vast datasets collected from the internet, which may include personal or private information. In addition, the retrieval database used in InstructRetro may contain private information as well. Third, although Retro framework was found to be effective reduce toxic generations (Wang et al., 2023a), it still reflect and can amplify the biases present in the training data as other LLMs.