Improving Retrieval Augmented Open-Domain Question-Answering with Vectorized Contexts

Zhuo Chen[1], Xinyu Wang[2]*, Yong Jiang[2]*, Pengjun Xie[2], Fei Huang[2], Kewei Tu[1]*
[1]School of Information Science and Technology, ShanghaiTech University
[1]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[2]Institute for Intelligent Computing, Alibaba Group
{chenzhuo,tukw}@shanghaitech.edu.cn
{tomas.wxy,yongjiang.jy,chengchen.xpj}@alibaba-inc.com

Abstract

In the era of large language models, applying techniques such as Retrieval Augmented Generation can better address Open-Domain Question-Answering problems. Due to constraints including model sizes and computing resources, the length of context is often limited, and it becomes challenging to empower the model to cover overlong contexts while answering questions from open domains. This paper proposes a general and convenient method to cover longer contexts in Open-Domain Question-Answering tasks. It leverages a small encoder and cross-attention mechanism and effectively encodes contexts. With our method, the original language models can cover several times longer contexts while keeping the computing requirements close to the baseline. Our experiments demonstrate that after fine-tuning, there is improved performance across two held-in datasets, four held-out datasets, and also in two In Context Learning settings. Our code will be released at https://github.com/Alibaba-NLP/Vec-RA-ODQA.

1 Introduction

Transformer-based (Vaswani et al., 2017) architectures with pre-training on large corpus have become popular in recent Natural Language Processing research (Brown et al., 2020; Workshop et al., 2022; Chowdhery et al., 2023). An increasing number of Natural Language Processing (NLP) tasks need to process long contexts such as Open-Domain Question Answering (ODQA) with Retrieval Augmented Generation (RAG) (Lewis et al., 2020; Izacard and Grave, 2020; Gu et al., 2018). However, the fine-tuning and inference stages in downstream tasks are still constrained by the input length, e.g., 2048 tokens for Bloomz (Muennighoff et al., 2022) and Llama-1 (Touvron et al., 2023).

With RAG, the input can easily surpass the maximum length the model can handle and it becomes challenging for the model to perform both fine-tuning and inference on overlong contexts. Moreover, in the in-context learning (ICL) (Dong et al., 2022; Kim et al., 2022) setting, the context will be much longer together with retrieved contexts. In such cases, the demand for the model to handle longer input text significantly increases.

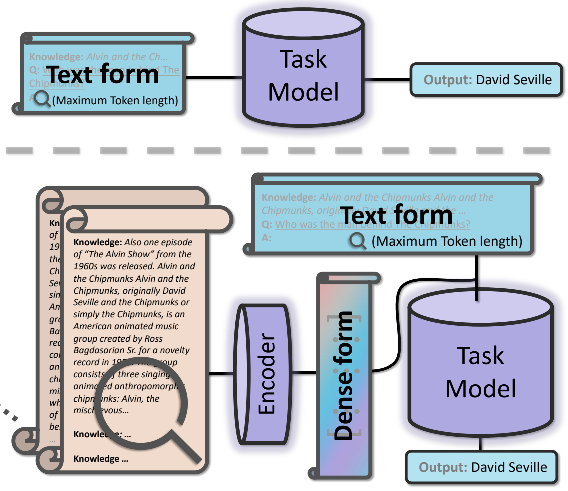

Figure 1: A comparison of our method (lower) and retrieval augmented ODQA without vectorization (upper). In the upper part, limited retrieved contexts are processed by the task model to finish the task. The lower part illustrates our method in which an encoder is incorporated to encode overlong retrieved contexts.

To enable the model to cover longer context during both fine-tuning and inference stages, this paper proposes a method that leverages a 100 million-level encoder model in downstream ODQA tasks with a 1 billion-level language model as illustrated in the lower part of Fig. 1. With our method, the length of context that the model can cover increases from 2k (in text form) to a maximum of 10k (in dense form, which is condensed by the encoder). Experiments are designed under three settings to validate the effectiveness of our method. In the experiments, we first fine-tune the model, optionally including the encoder, on two popular ODQA

*Corresponding author

Findings of the Association for Computational Linguistics ACL 2024, pages 7683–7694
August 11-16, 2024 ©2024 Association for Computational Linguistics

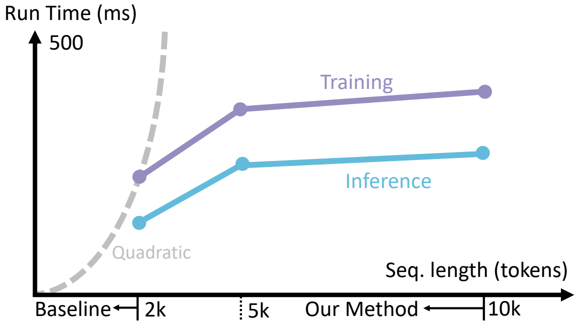

Figure 2: Speed illustration. Run time is measured on a single A100 GPU and the `batch size` is set to 1 for all curves. "2k" on the horizontal axis represents the baseline model's run time to train or infer on data of length 2k. "5k" and "10k" correspond to two variants of our method that can cover at most 5k and 10k tokens when training and inferring. Training time measures the average over five consecutive training steps. Inference time measures the average over five consecutive generation steps. Specifically, we measure the execution duration of functions `Trainer.training_step` and `model.generate` based on `huggingface`.

datasets with retrieved contexts and evaluate our method in held-in, held-out, and ICL settings. Experimental results show that our method outperforms the baseline, which is fine-tuned on data of length 2k, in all three settings.

Regarding the speed of our method, we measure the run time of each training and inference step. Compared with work that compresses the contexts with the original task model (Chevalier et al., 2023), which requires techniques to reduce the computation graph during backpropagation, we employ a 10x smaller model to perform the encoding of excessive texts, so a complete gradient descent procedure can be kept. To sum up, our contributions are as follows:

1. We propose a method that incorporates a small encoder model for excessively long context encoding by applying cross-attention mechanism with the original task model.
2. We evaluate our method in two held-in, four held-out, and two ICL settings after being fine-tuned on two ODQA datasets and obtain improved performance.
3. The computing resource requirements of our method are consistent with those of the baseline and the run time remains competitive.

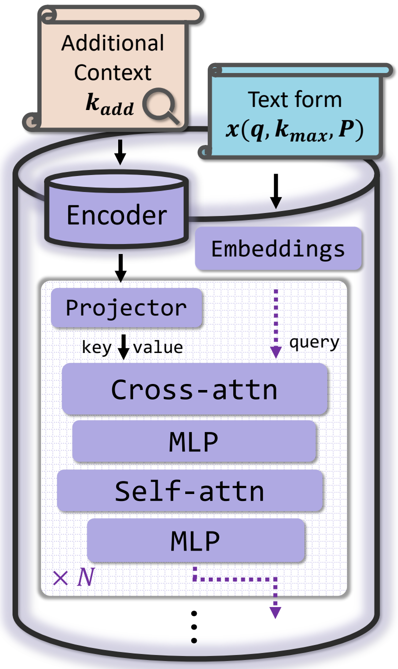

Figure 3: Method illustration of model architecture (purple blocks) and data flows (along black/purple arrows). The purple dashed arrows mean that the output of `MLP` module will be the "query" to the next layer of `Cross-attn` module. $\times N$ means that the modules with dotted backgrounds are repeated with multiple layers in the task model.

2 Method

2.1 Background

Consider an example query $\boldsymbol{q}$ with gold answer $\boldsymbol{a}$ and independent C pieces of corresponding context information $\boldsymbol{k} = \{\boldsymbol{k_1}, \boldsymbol{k_2}, ..., \boldsymbol{k_C}\}$, with each being a sequence of tokens, where $\boldsymbol{k}$ is retrieved by some retriever from a given corpus[1]

$$\boldsymbol{k} = \text{Retriever}(\boldsymbol{q}, \text{corpus})$$

Ideally, the C retrieved contexts contain the knowledge needed to answer $\boldsymbol{q}$ correctly, but there may also be noise. Given a decoder model Dec parameterized by θ, the output sequence $\boldsymbol{y}$ is usually modeled by

$$P_\theta(\boldsymbol{y}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P}) = Dec(\boldsymbol{y}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P})$$

where $\boldsymbol{k_{max}} = \{\boldsymbol{k_1}, \boldsymbol{k_2}, ..., \boldsymbol{k_m}\} \in \boldsymbol{k}, m < C$. m refers to the number of contexts that reach the model's throughput. $\boldsymbol{P}$ stands for the prompts that connect related content[2]. Given the model,

[1]Refer to Sec. 3.1 for detailed definition of corpus and retriever in our experiments.

[2]The forms of $\boldsymbol{P}$ vary with different settings, and there will be detailed definitions in Sec. 3.1.

$\boldsymbol{k_{max}}$ is usually a subset of $\boldsymbol{k}$ because the maximum length of contexts is often constrained by the model's throughput or computing resources, and

During training, we aim to maximize the term $P_\theta(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P})$, and formalize the ODQA problem as a language modeling task. Specifically, for a query $\boldsymbol{q}$, its gold answer $\boldsymbol{a}$ and contexts $\boldsymbol{k_{max}}$, they are connected linguistically with proper prompts $\boldsymbol{P}$, together denoted as an input sequence $\boldsymbol{x}(\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{k_{max}}, \boldsymbol{P}) = \{x_1, x_2, ...\}$. Then we aim to minimize the language modeling loss over the set $\mathcal{D}$ of all training examples:

$$L_\theta(\mathcal{D}) = - \sum_{\boldsymbol{x}(\boldsymbol{q},\boldsymbol{a},\boldsymbol{k_{max}},\boldsymbol{P}) \in \mathcal{D}} \sum_i log(P_\theta(x_i|x_{<i})) \quad (1)$$

2.2 Encoding and Cross-Attention

We propose a method that can utilize additional contexts $\boldsymbol{k_{add}} = \{\boldsymbol{k_{m+1}}, \boldsymbol{k_{m+2}}, ...\}$ several times longer than $\boldsymbol{k_{max}}$. First, we introduce an encoder parameterized by ϕ. Then we apply cross-attention with the original task model and introduce a projector, a cross-attention module and a Multi-Layer Perceptron (MLP) in each layer, together denoted the parameters as π. Denote $\omega = \{\phi, \pi, \theta\}$ as all the parameters in our model. On the whole, our method models the output $\boldsymbol{y}$ by an encoder-decoder model $Enc\text{-}Dec$

$$\begin{aligned} & Q_\omega(\boldsymbol{y}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P}, \boldsymbol{k_{add}}) \\ = \ & Enc\text{-}Dec(\boldsymbol{y}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P}, \boldsymbol{k_{add}}) \end{aligned}$$

During training, inputs $\boldsymbol{x}(\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{k_{max}}, \boldsymbol{P})$ are embedded by the original task model's embedding layer Emb

$$\boldsymbol{h_q} = Emb(\boldsymbol{x}(\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{k_{max}}, \boldsymbol{P}))$$

and each of the additional contexts $\boldsymbol{k_i}$ in $\boldsymbol{k_{add}}$ is encoded by the encoder Enc

$$\boldsymbol{h_{add}^{(i)}} = Enc(\boldsymbol{k_i})$$

Note that the length of encoding from the encoder is flexible practically and we compress each $\boldsymbol{k_i}$ into one vector. Following the output of the encoder, a projector $Proj$ is used to align the high-dimensional hidden spaces between the encoder and task model in each layer

$$\boldsymbol{h_{kv}} = Proj(\boldsymbol{h_{add}})$$

where $\boldsymbol{h_{add}}$ is concatenated of all $\boldsymbol{h_{add}^{(i)}}$ calculated from last step. Each layer of the task model is assigned to an independent projector as different layers may learn different representations.

In each layer, to incorporate the information stored in $\boldsymbol{k_{add}}$ we add a cross-attention module, where representations of additional contexts $\boldsymbol{h_{kv}}$ serve as "key" and "value", followed by an MLP. In the first layer, the embeddings of original input $\boldsymbol{h_q}$ act as "query", and in the rest of the layers output $\boldsymbol{h_q'}$ from the previous layer act as "query" ($\boldsymbol{h_q'}$ will be defined later).

$$\begin{aligned} \boldsymbol{h_c} &= Cross\text{-}attn(\boldsymbol{h_q}/\boldsymbol{h_q'}, \boldsymbol{h_{kv}}) \\ \boldsymbol{h_m} &= MLP(\boldsymbol{h_c}) \end{aligned}$$

$Cross\text{-}attn(\boldsymbol{h_q}, \boldsymbol{h_{kv}})$ is calculated as follows

$$\begin{aligned} Q &= W^Q \boldsymbol{h_q} \\ K, V &= W^K \boldsymbol{h_{kv}}, W^V \boldsymbol{h_{kv}} \\ o &= softmax(\frac{QK^T}{\sqrt{d_k}})V \\ \boldsymbol{h_c} &= W^O o \end{aligned}$$

where W^Q, W^K, W^V, W^O refer to weight matrices and d_k refers to the dimension of each attention head. Then the output of cross-attention and MLP is normally processed by a self-attention and another MLP module. The output acts as "query" input to the cross-attention module in the next layer.

$$\boldsymbol{h_q'} = MLP(Self\text{-}attn(\boldsymbol{h_m}))$$

At last, the output of the last layer is expanded to the vocabulary-size dimension to predict the next token (not shown in Fig. 3 for simplicity), and we aim to maximize the probability

$$Q_\omega(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P}, \boldsymbol{k_{add}})$$

Consistent with the setup mentioned before, to maximize term $Q_\omega(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{k_{max}}, \boldsymbol{P}, \boldsymbol{k_{add}})$, we turn it into minimizing the language modeling loss

$$J_\omega(\mathcal{D}) = - \sum_{\boldsymbol{x}(\boldsymbol{q},\boldsymbol{a},\boldsymbol{k_{max}},\boldsymbol{P}), \boldsymbol{k_{add}} \in \mathcal{D}} \sum_i log(Q_\omega(x_i|x_{<i}, \boldsymbol{k_{add}})) \quad (2)$$

2.3 ICL Setting

Our method can also be applied to ICL settings. Based on the aforementioned setup, we denoted ICL samples as $\boldsymbol{l_{max}} = \{\boldsymbol{l_1}, \boldsymbol{l_2}, ..., \boldsymbol{l_m}\}$, with each $\boldsymbol{l_i}$ composed of another pair of query and answer. We optimize objective 3 below on data where each $\boldsymbol{l_i}(\boldsymbol{q}^{'}, \boldsymbol{a}^{'})$ refers to only query-answer ICL samples (without context) and $\boldsymbol{q}^{'}$ $\boldsymbol{a}^{'}$ refer to another query-answer pair:

$$J_{\omega}^{'}(\mathcal{D}) = -\sum_{\boldsymbol{s}(\boldsymbol{q},\boldsymbol{a},\boldsymbol{l_{max}},\boldsymbol{P}),\boldsymbol{k_{add}} \in \mathcal{D}} \sum_{i} log(Q_{\omega}^{'}(s_i | s_{<i}, \boldsymbol{k_{add}})) \tag{3}$$

$\boldsymbol{s} = \{s_1, s_2, ...\}$ refers to the inputs composed of $(\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{l_{max}}, \boldsymbol{P})$ and $Q^{'}$ shares a similar definition to Q in objective 2. Additional contexts $\boldsymbol{k_{add}}$ are utilized in the same way as in Sec. 2.2 by performing encoding, cross-attention, etc.

2.4 Training

Theoretically, training processes stated in Sec. 2.2 all remain differentiable and thus all the parameters can be optimized via normal gradient descent w.r.t. objective 2. Note that the parameters ϕ of the encoder can be initialized from a well-pre-trained model on a large scale corpus and the pre-trained parameters possess good performance in many downstream tasks based on text encoding. However, the parameters in the projector module are randomly initialized. Thus at the start of the training, according to the chain rule, the gradients to the whole encoder will be random as well, which poses a risk of breaking the encoding utility of the encoder. This intuition proves to be true in our experiments.

Therefore, we design two strategies of training:

1. Directly freeze parameters ϕ and make parameters (π, θ) trainable during the whole training process.

2. In the first few training steps (e.g., one epoch), ϕ is kept frozen to prevent random gradients from breaking its well-pre-trained parameters. After that, ϕ is optimized w.r.t. objective 2 together with the other modules (π, θ).

3 Experiment

3.1 Experiment settings

Settings	Data format
Held-in Held-out	Answer the question: Knowledge: {context $\boldsymbol{k_1}$ } ...{context $\boldsymbol{k_m}$}. Q: Who got the first nobel prize in physics A:
ICL format w/ contexts	Answer the following questions based on the Knowledge: Knowledge: {context $\boldsymbol{k_1^{'}}$} Q: Who developed the first printing press in 1430s A: Johannes Gutenberg ...(Knowledge: ... Q: ... A: ...) Knowledge: {context $\boldsymbol{k_1^{''}}$} Q: Who got the first nobel prize in physics A:
ICL format w/o contexts (Sec. 2.3)	Answer the following questions: Q: Who developed the first printing press in 1430s A: Johannes Gutenberg (Q: ... A: ...)... Q: Who got the first nobel prize in physics A:
Additional Contexts	{context $\boldsymbol{k_{m+1}}$}; {context $\boldsymbol{k_{m+2}}$}; ...

Table 1: Examples of data format. Gray tokens refer to prompts $\boldsymbol{P}$ mentioned in Sec. 2 and the context is omitted here.

Data To evaluate our method, we first fine-tune our model on two ODQA datasets separately, TriviaQA (Joshi et al., 2017) and Natural Questions (NQ) (Kwiatkowski et al., 2019). Besides evaluating our method on the held-in data, we also evaluate four held-out data, namely CommonsenseQA (Talmor et al., 2019), SQuAD2.0 (Rajpurkar et al., 2016), Webquestions (Berant et al., 2013) and ComplexWebQuestions (Talmor and Berant, 2018). Specifically, samples in CommonsenseQA dataset are formulated as multi-choice problems, and we evaluate the performance in both multi-choice and sequence-to-sequence formats. Refer to App. A.1 for the detailed format.

Format of input $\boldsymbol{x}$ in Sec. 2.2 is formulated as "Held-in Held-out" format in Table 1, and we evaluate the model's performance on samples of ICL format with context. Format of input $\boldsymbol{s}$ in Sec. 2.3 is formulated as "ICL format w/o contexts" in Table 1.

Additional contexts $\boldsymbol{k_{m+1}}, \boldsymbol{k_{m+2}}$ are encoded by the encoder separately and independently without prompts. The forms of prompts $\boldsymbol{P}$ defined previously are shown in gray tokens in Table 1.

Retriever For contexts of the datasets TriviaQA and NQ, we utilize those collected by Karpukhin et al. (2020), which are collected with BM25 (Robertson et al., 2009) and Dense Passage Retrieval techniques. For contexts of the four held-out datasets, we follow Izacard et al. (2022) and Shi et al. (2023) and use Contriver (Izacard et al., 2021) as our retriever. Contexts $\boldsymbol{k}$ are retrieved from Wikipedia dump dated December 20, 2018, the version released by Karpukhin et al. (2020).

Baseline Recent decoder-only models like Bloomz (Muennighoff et al., 2022) and GPTs (Radford et al., 2019; Achiam et al., 2023) have shown good performance in generation-like tasks, and we use *Bloomz-1b7*[3] for the task model θ. When fine-tuning the baseline model, inputs are constructed according to the "Held-in Held-out" setting as stated in Table 1. The length of the input is extended to utilize as many contexts as possible, consistent with the maximum input length (2k) of the model while doing pre-training (Workshop et al., 2022).

Additionally, note that the context information $\boldsymbol{k}_{max}$ provided in the inputs is ranked from best to worst based on Dense Retrieval (Karpukhin et al., 2020), which means the baseline we adopt is rather stronger than randomly providing as many contexts as possible without considering the quality. The baseline can be seen as a model fine-tuned on the most relevant contexts incorporating reranking techniques (Karpukhin et al., 2020; Khalifa et al., 2023).

Initialization and Training Settings Weights of popular pre-trained encoder models like BERT (Devlin et al., 2018) should be good initialization for the encoder ϕ and thus we adopt *BERT-base-uncased*[4] for initialization of ϕ. Parameters of attention and MLP modules are also adapted from *Bloomz-1b7*. To keep the encoding process efficient, we use a simple `Linear` module as the projector that is randomly initialized and fine-tuned to align the hidden dimension of 768 (*BERT-base-uncased*) to 2048 (*Bloomz-1b7*).

In our experiment, we use BERT to independently encode additional contexts on 10 or 20 contexts, which can cover approximately 5k to 10k additional context tokens. Then the hidden states of the *[CLS]* token are concatenated and fed-forward to subsequent modules as illustrated in Fig. 3. For both the baseline and our method, we evaluate the model checkpoint with the lowest language modeling loss on the development set and report the Exact Match (EM) metric.

As discussed in Sec. 2.4, there are mainly two choices of training strategies of which parts of our proposed model are optimized. We experiment with both strategies and report the results of the "frozen encoder" setting in Sec. 3.2 and the "training encoder" setting in Sec. 4.1 respectively.

Learning Rate	2e-5
Optimizer	AdamW
Lr scheduler	cosine
Warmup ratio	0.03
FP 16	True
FP 16 eval	True
Globa batch size	8
Save steps	4000
Eval steps	4000
Max epochs	4
GPU name	NVIDIA A100-SXM 80G

Table 3: Hyperparameters

Hyperparameters We list important hyperparameters in our experiments in Table 3.

3.2 Main Results

We present our main result of the first training strategy discussed in Sec. 2.4 in Table 4. Upon fine-tuning on two datasets and evaluating on three (held-in, held-out and ICL) settings, our method achieves performance superior to that of the baseline in five out of six settings, except for one setting on one dataset.

In held-in settings (training on TriviaQA/NQ and evaluating on TriviaQA/NQ), our model consistently demonstrates superior performance relative to the baseline. Moreover, it demonstrates stable improved performance as more contexts are encoded by our method, showing the potential of our model to encode even longer contexts.

In held-out settings, our method outperforms the baseline in all the datasets after being fine-tuned on TriviaQA and outperforms three of four datasets after being fine-tuned on NQ, suggesting the general applicability of our method. From the "Com.QA

[3]https://huggingface.co/bigscience/bloomz-1b7
[4]https://huggingface.co/bert-base-uncased

Train \ Evaluate		TriviaQA		NQ		Com.QA *test*		SQuAD	Web.Q	Comp.Q	Triviaqa (ICL)		NQ (ICL)	
		dev	*test*	*dev*	*test*	*choice*	*seq2seq*	*test*	*test*	*test*	*dev*	*test*	*dev*	*test*
TriviaQA	baseline	45.740	46.203	14.868	16.288	17.199	2.785	10.191	9.524	4.490	31.764	31.857	8.999	9.058
	+ 5k	47.686	47.742	17.506	19.307	**19.328**	**3.194**	**12.684**	10.053	5.513	32.341	32.034	10.677	11.136
	+ 10k	**47.901**	**48.245**	**18.465**	**19.529**	17.363	2.539	12.667	**11.111**	**6.024**	**34.027**	**34.235**	**11.671**	**11.801**
NQ	baseline	42.809	43.976	37.159	37.978	19.410	4.095	21.199	14.815	**13.498**	35.521	35.967	19.242	21.136
	+ 5k	43.669	44.657	37.01	38.698	19.656	4.095	22.724	15.344	13.214	35.883	35.985	19.265	21.413
	+ 10k	**44.189**	**45.107**	**37.581**	**39.114**	**21.294**	**4.423**	**22.918**	**15.873**	13.413	**36.381**	**36.569**	**19.447**	**21.662**

Table 4: Main results of performance with frozen encoder on held-in, held-out and ICL settings. **Boldface** marks the best results in each setting. Com.QA refers to CommonsenseQA. Web.Q refers to WebQuestions. Comp.Q refers ComplexWebQuestions. TriviaQA (ICL) and NQ (ICL) show the results evaluated on ICL setting where the data is formed as illustrated in Table 1 ICL.

choice" setting we can see that though our model is not trained to answer multi-choice questions, it performs better in selecting choices than baseline.

In the last two columns TriviaQA (ICL) and NQ (ICL), we evaluate whether the optimized model can generalize to a similar ICL setting. Specifically, with optimized parameter ω^* after fine-tuning objective 2 we evaluate how well we can model $Q_{\omega^*}(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{l_{max}}, \boldsymbol{P}, \boldsymbol{k_{add}})$ where each $\boldsymbol{l_i}(\boldsymbol{q}', \boldsymbol{a}', \boldsymbol{k}')$ is an ICL sample composed of another query, context and answer. Surprisingly, we obtain a similar improved performance to the held-in setting. Steadily improved performance indicates that the training method we adopt is robust, maintaining both the encoder and decoder's efficacy in retrieving useful information while the evaluation data format diverges from the training data.

In summary, from the results presented in Table 4, it is observable that in comparison with the baseline, employing our method to encode a greater volume of retrieval information offers a predominantly positive enhancement to the model's performance across various settings, including held-in, held-out, and ICL.

4 Analysis

In this section, we present the results of three analytical experiments. The first one shows the result of the other training strategy discussed in Sec. 2.4. The second shows the evaluation results of optimizing objective 3. The third shows the effectiveness of our method in a more challenging setting.

4.1 Encoder Training

In our experiments, we first try optimizing the encoder ϕ with the other parameters (π, θ) from the very beginning of the training process. Results turn out to verify our anticipation: newly introduced random parameters (the projector) easily mess up with the parameters in the encoder, consequently undermining its capability to encode information and resulting in worse performance than baseline.

Here we evaluate the training strategy we proposed in Sec. 2.4 that aims to fix this problem. The encoder is optimized after several training steps, and in our experiment, we set it to one epoch. Besides, the parameters in the cross-attention module are initialized by those in the pre-trained self-attention module to minimize the amount of randomly initialized parameters.

Evaluations are done in the same settings as in Table 4. By applying this two-step training method, we succeed in obtaining better performance than the baseline in most of the settings. It can be inferred that compared with the setting of a frozen encoder (i.e., ϕ is not optimized), further introducing trainable encoder parameters did not further enhance the model's performance as anticipated. Although we can achieve better results in most settings than baseline, performance in held-in and held-out settings seems to be less stable compared to the "frozen encoder" setting. Particularly, we find that optimizing the encoder results in degraded performance in the ICL setting, especially after being fine-tuned on TriviaQA datasets. We attribute this to the fact that million-scale parameter models, after fine-tuning on certain data, cannot guarantee to generalize the encoding capability to a broader range of scenarios, e.g. the ICL setting, as defined in Table 1. We present the results of the second training strategy discussed in Sec. 2.4 in Table 5.

Train \ Evaluate		TriviaQA		NQ		Com.QA *test*		SQuAD	Web.Q	Comp.Q	Triviaqa (ICL)		NQ (ICL)	
		dev	*test*	*dev*	*test*	*choice*	*seq2seq*	*test*	*test*	*test*	*dev*	*test*	*dev*	*test*
TriviaQA	baseline	45.740	46.203	14.868	16.288	**17.199**	2.785	10.191	9.524	4.490	**31.764**	**31.857**	**8.999**	**9.058**
	+ 5k	**48.082**	**47.803**	15.896	16.898	14.333	**3.112**	10.755	**11.640**	4.945	16.646	16.645	6.178	6.676
	+ 10k	47.980	47.750	**16.079**	**17.008**	15.807	2.867	**10.823**	11.111	**5.058**	16.103	16.300	6.383	7.008
NQ	baseline	42.809	43.976	37.159	37.978	**19.410**	**4.095**	21.199	14.815	**13.498**	35.521	**35.967**	**19.242**	**21.136**
	+ 5k	**43.397**	**44.524**	**37.387**	39.03	15.889	**4.095**	22.092	14.286	12.958	33.993	34.341	17.951	19.640
	+ 10k	43.284	44.047	37.205	**39.28**	16.790	3.931	**22.143**	**16.402**	12.788	33.122	33.404	18.933	20.914

Table 5: Analysis of training encoder along with the task model when fine-tuning. Experiments are conducted under the same setting to Sec. 3.2

ICL samples w/o contexts	TriviaQA		NQ	
	dev	*test*	*dev*	*test*
baseline	20.052	20.083	19.242	19.529
+ 10 vec	19.939	20.233	**19.539**	**19.668**
+ 20 vec	**20.358**	**20.578**	19.333	19.501

Table 6: Result of fine-tuning on data with ICL samples (without context information) and evaluating on held-in setting.

$\boldsymbol{k_{max}} = \{\}$	TriviaQA		NQ	
	dev	*test*	*dev*	*test*
baseline	20.391	20.472	18.968	19.889
+ 1 vec	21.636	21.533	20.041	20.637
+ 5 vec	21.942	22.010	20.258	20.942
+ 10 vec	**21.964**	**22.072**	**22.268**	**22.632**

Table 7: Effectiveness of our method on encoding when we remove the influence on text form context information in $\boldsymbol{x}$.

4.2 ICL Setting w/o Contexts

We also experiment with optimizing objective 3 defined in Sec. 2.3 where only query-answer pairs are provided in the ICL format input. The detailed data format is shown in Table 1 "ICL format w/o contexts" and the query-answer pair is sampled as many as possible from the held-in dataset. The utility of the encoder remains the same as it encodes 10 (+ 10 vec) or 20 (+ 20 vec) pieces of context and is kept frozen during the training.

The model is fine-tuned on TriviaQA and NQ and evaluated in held-in settings. We report the result in Table 6. First, we see that our method can still enhance the model in this setting but the improvements seem to be not consistent or prominent. Second, notice that the improvement on each dataset is not as remarkable as that in the ICL setting in Table 4, where each ICL sample is provided along with one piece of context.

To summarize the findings here, our method for encoding context exhibits a more pronounced performance enhancement in ICL settings that incorporate context information. We posit that the underlying reason for this is that the cross-attention mechanism, which facilitates information interchange between inputs (embedded by the task model) and dense context information (encoded by the encoder), is particularly effective when context interacts with context, instead of context with ICL samples with only query-answer pairs.

4.3 A More Challenging Setting

In our method presented in Sec. 2.2, we adopt a projector module that is applied to align the high-dimensional hidden spaces and adopt cross-attention mechanism to incorporate the dense context information in each layer. In this section, we evaluate the effectiveness of our method in a more challenging setting.

Specifically, compared to the data format stated in the "Held-in Held-out" setting in Table 1, we remove the contexts in input $\boldsymbol{x}$ and keep only questions and answers in the training data, i.e., $\boldsymbol{x}$ in objective 2 becomes $(\boldsymbol{q}, \boldsymbol{a}, \{\}, \boldsymbol{P})$. Only several contexts are supplied as "Additional Contexts" encoded by the encoder. Note that though supplying text-form contexts can greatly enhance models in ODQA tasks, here we remove them to test the effectiveness of the encoder and cross-attention mechanism in a more challenging setting.

Results are shown in Table. 7. "+ 1/5/10 vec" means we utilize 1/5/10 pieces of contexts and encode them into 1/5/10 vectors by taking the *[CLS]* tokens' hidden states. It can be inferred that, firstly, with only one encoded vector, our method can enhance the model. Secondly, we observe consistent improvement across two datasets and three variants of our method that incorporating more contexts leads to better performance.

5 Related Work

5.1 Retrieval Augmentation

Recently, retrieval augmentation has been utilized to improve a large amount of Natural Language Processing downstream tasks such as question-answering (Chen et al., 2017; Lewis et al., 2020; Kwiatkowski et al., 2019; Fan et al., 2019), dialogue (Moghe et al., 2018), language modeling (Khandelwal et al., 2020), NER (Wang et al., 2022, 2021) and machine translation (Gu et al., 2018; Xu et al., 2022). In the aforementioned work, the utilization of retrieval information has been fundamentally capable of enhancing model performance across all dimensions.

5.2 Related Model Architectures

Referring to the base model, there has been increasing interest in using models of encoder-decoder or decoder-only architectures in solving downstream tasks with retrieval augmentation recently.

Allaouzi et al. (2019) and Zhou et al. (2023) employ models of encoder-decoder architectures to solve visual question answering task in the medical domain. In their work, the encoder model is responsible for extracting prominent features from a medical image and the decoder part generates the answer. Li et al. (2023) utilizes an encoder-decoder model with constrained decoding to solve extractive question answering task.

Decoder-only models, e.g., ChatGPT and GPT-4 (Achiam et al., 2023), are more famous for their surprisingly great performance on tasks like question answering (Ali et al., 2022) and there is abundant work that tries to improve the performance based on GPTs (Pereira et al., 2023). Kim and Min (2024) introduce a chatbot model that utilizes generative AI and the Retrieval Augmented Generation method to address the issue that achieving regulatory compliance necessitates the intricate navigation of exceptionally complex and voluminous guidelines in the pharmaceutical industry.

In our work, we also incorporate an encoder for context encoding. However, compared to the traditional encoder-decoder models, the encoder part in our method is several times smaller than the decoder part. Although our method does not alter the quadratic complexity of the attention mechanism, it instead processes the long contexts in a much lower dimension, thus being able to quintuple the capacity to cover context information without the need to utilize additional computing resources.

5.3 Utilizing Long Contexts

To handle contexts with excessive length, recently proposed techniques such as context compression are increasingly investigated in NLP research.

Chevalier et al. (2023) proposes "AutoCompressors" that uses OPT (Zhang et al., 2022) and Llama-2 (Touvron et al., 2023) to compress texts into summary vectors and show that utilizing long contexts can improve perplexity. In their method, the compression is done by the billion-level language model, and in one of their experiments, they train on sequence with 30720 tokens with 20 compression steps. However, the complete computation graph cannot be fully kept in such settings, and the optimizing process has to rely on stopping gradients, which poses potential risks to the mathematical principle behind gradient descent. Similarly in Zhang et al. (2024)'s work, the long context is first partitioned into multiple intervals, and then a sliding window is employed to sequentially process one interval at a time and the compressed token embeddings are kept for the next token prediction. It is implemented by introducing additional trainable parameters to the origin language model to finish the task of "Activation Condensing", and original parameters are frozen throughout the training process.

6 Conclusion

In this paper, we propose a method that incorporates a small encoder model for excessively long context encoding by applying cross-attention mechanism with the original task model. The method is simple and general for transformer-based language models. In our experiments, after fine-tuning on ODQA dataset, we find improved performance across two held-in, four held-out and two ICL settings, compared to a baseline that incorporates the reranking technique on training data, showing the effectiveness of our method in utilizing long contexts. We note that the intuitive explanations for the performance improvement are as follows: 1) the encoder model provides the ability to encode longer contexts; 2) the cross-attention mechanism is useful in selectively attending the correct parts of the inputs. Regarding the efficiency, the need for GPU quantity remains unchanged and the run time remains competitive to the baseline.

7 Limitations

First, we have only tested our method in 1B7 models with a 110M encoder, and yet we have not tested the effectiveness of our method on larger language models, e.g., 7B and 70B, due to limited computing resources.

Second, we observe that our method exhibits relatively modest performance under setting 4.2, with only a slight improvement compared to the baseline. We attribute the potential reasons for this to the cross-attention mechanism being unsuitable for modeling the relationship between context and ICL samples (without contexts).

Acknowledgement

This work was supported by Alibaba Group through Alibaba Innovative Research Program.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohaid Ali, Oliver Y Tang, Ian D Connolly, Jared S Fridley, John H Shin, Patricia L Zadnik Sullivan, Deus Cielo, Adetokunbo A Oyelese, Curtis E Doberstein, Albert E Telfeian, et al. 2022. Performance of chatgpt, gpt-4, and google bard on a neurosurgery oral boards preparation question bank. *Neurosurgery*, pages 10–1227.

Imane Allaouzi, Mohamed Ben Ahmed, and Badr Benamrou. 2019. An encoder-decoder model for visual question answering in the medical domain. In *CLEF (working notes)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint 2305.14788*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of* *the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Few-shot reranking for multi-hop QA via language model prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15882–15897, Toronto, Canada. Association for Computational Linguistics.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2022. Self-generated in-context learning: Leveraging autoregressive language models as a demonstration generator. *arXiv preprint arXiv:2206.08082*.

Jaewoong Kim and Moohong Min. 2024. From rag to qa-rag: Integrating generative ai for pharmaceutical regulatory compliance process. *arXiv preprint arXiv:2402.01717*.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Shaobo Li, Chengjie Sun, Bingquan Liu, Yuanchao Liu, and Zhenzhou Ji. 2023. Modeling extractive question answering using encoder-decoder models with constrained decoding and evaluation-based reinforcement learning. *Mathematics*, 11(7).

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332, Brussels, Belgium. Association for Computational Linguistics.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Jayr Pereira, Robson Fidalgo, Roberto Lotufo, and Rodrigo Nogueira. 2023. Visconde: Multi-document qa with gpt-3 and neural reranking. In *European Conference on Information Retrieval*, pages 534–543. Springer.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving Named Entity Recognition by External

Context Retrieving and Cooperative Learning. In *the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (**ACL-IJCNLP 2021**)*. Association for Computational Linguistics.

Xinyu Wang, Yongliang Shen, Jiong Cai, Tao Wang, Xiaobin Wang, Pengjun Xie, Fei Huang, Weiming Lu, Yueting Zhuang, Kewei Tu, Wei Lu, and Yong Jiang. 2022. DAMO-NLP at SemEval-2022 task 11: A knowledge-based system for multilingual named entity recognition. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1457–1468, Seattle, United States. Association for Computational Linguistics.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Jitao Xu, Josep Crego, and Jean Senellart. 2022. Boosting neural machine translation with similar translations. In *Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track)*, pages 282–292, Orlando, USA. Association for Machine Translation in the Americas.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*.

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

Yuan Zhou, Jing Mei, Yiqin Yu, and Tanveer Syeda-Mahmood. 2023. Medical visual question answering using joint self-supervised learning. *arXiv preprint arXiv:2302.13069*.

A Appendix

A.1 CommonsenseQA Format

We show how we reformat data from CommonsenseQA in Table 8. Reformated *choice* turn A/B/C/D/E into 1/2/3/4/5 to avoid causing ambiguity with "A:" in prompts $\boldsymbol{P}$. The choices are removed in *seq2seq* format and the problem becomes more challenging.

Setting	Format
Origin Format	A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? A: bank B: library C: department store D: mall E: new york Answer: A
Reformatted *choice*	Q: A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? Choose from 1-5 given below. 1: bank 2: library 3: department store 4: mall 5: new york A: Answer: 1 or bank
Reformatted *seq2seq*	Q: A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? A: Answer: bank

Table 8