IDEAL: Leveraging Infinite and Dynamic Characterizations of Large Language Models for Query-focused Summarization

Anonymous ACL submission

Abstract

Query-focused summarization (QFS) aims to produce summaries that answer particular questions of interest, enabling greater user control and personalization. With the advent of large language models (LLMs), shows their impressive capability of textual understanding through large-scale pretraining, which implies the great potential of extractive snippet generation. In this paper, we systematically investigated two indispensable characteristics that the LLMs-based QFS models should be harnessed, *Lengthy Document Summarization* and *Efficiently Fine-grained Query-LLM Alignment*, respectively. Correspondingly, we propose two modules called Query-aware HyperExpert and Query-focused Infini-attention to access the aforementioned characteristics. These innovations pave the way for broader application and accessibility in the field of QFS technology. Extensive experiments conducted on existing QFS benchmarks indicate the effectiveness and generalizability of the proposed approach.

1 Introduction

In today's world, where we are constantly bombarded with vast amounts of text, the ability to efficiently summarize information has become crucial. Textual summarization (Gambhir and Gupta, 2017), the process of condensing a lengthy document into a succinct and digestible version while preserving the most crucial information, enabling quicker understanding and better management of information. As everyone has unique needs and interests in real-life scenarios, necessitating summarizers that succinctly address the information needed for a specific query by extracting essential information from documents, *i.e.*, **Q**uery-**F**ocused **S**ummarization (QFS) (Daumé III, 2009). This task involves analyzing the content to identify key themes and then highlighting these in the summary, which draws increasing attention in the textual summarization community.

Traditionally, QFS has used extract-then-summarize methods (Zhong et al., 2021; Wang et al., 2022; Amar et al., 2023) that rely on the most relevant spans of text from a candidate document-based on the prevalence of query terms. Further onwards, the triumph of Large Language Models (LLMs) such as the GPT series (Achiam et al., 2023), LLaMA (Touvron et al., 2023) and other open-source LLMs showcased the power of large-scale pretraining in understanding, reasoning and generating intricate textual patterns, the great potential of LLMs offering new opportunities for QFS. However, there has been relatively little investigation into LLMs-based QFS methods (Yang et al., 2023a). Our primary goal in this paper is to close this gap correspondingly by proposing two indispensable characteristics that should be harnessed by LLMs while dealing with QFS: (i) **Efficiently Fine-grained Query-LLM Alignment**, as commonly known, the pre-trained LLMs are powerful when transferred to downstream tasks with instruction tuning(Ouyang et al., 2022), this also applies to the QFS task when the LLMs specialized for user's interests. However, as the parameter number grows exponentially to billions or even trillions, it becomes very inefficient to save the fully fine-tuned parameters for each downstream task. Besides, the different data distribution of diverse user's queries or instructions may introduce the negative transfer in the training stage (Wang et al., 2019). This implies the QFS model should minimize the potential interference among different user instructions, thereby accessing the fine-grained query-LLM alignment. (ii) **Lengthy Document Summarization**, general LLMs can't handle long text inputs due to the huge amount of memory required during training. Besides, the simple approach of concatenating the query to the input document is insufficient for effectively guiding the model to focus on the query while generating the summary. How to process the lengthy documents is also an important characteristic of LLMs-based QFS approaches. Summing up, these characteristics necessitate a thorough reevaluation of QFS and its corresponding solutions with LLMs.

Based on the aforementioned insights, we propose **I**nfinite and **D**ynamic larg**E** langu**A**ge mode**L**-based framework, abbreviated as IDEAL for ideal QFS, which consists of two modules: **Query-aware HyperExpert** and **Query-focused Infini-attention** achieve the two indispensable characteristics, respectively. The Query-aware HyperExpert (Figure 1) leverages the parameter-efficient fine-tuning (PEFT) (Mangrulkar et al., 2022) strategies that enable a model to perform a new task with minimal parameter updates. Innovatively, we tailor the previous PEFT approaches to QFS tasks with a HyperNetwork (Ha et al., 2016), which can dynamically generate the strongly correlated LLM's parameter shifts according to users' queries. Such dynamic characterization allows us to achieve the best of both worlds by adjusting the LLM's parameters while encouraging the model to adapt to each individual instance. By doing so, efficient and fine-grained query-LLM alignment can be achieved. Notably, we develop three types of HyperExpert, including Prompt-tuning (Lester et al., 2021), Parallel Adapter (He et al., 2022), and Low-Rank Adaptation (LoRA) (Hu et al., 2021). To process long documents with bounded memory and computation, we propose incorporating a Query-focused Infini-attention (Figure 2) module into IDEAL. Infini-attention (Munkhdalai et al., 2024) includes a long-term compressive memory and local causal attention for efficiently modeling both long- and short-range contextual dependencies. Our Query-focused Infini-attention possesses an extra query-focused compressive memory to better retain parts of the input documents that are strongly correlated with the query.

Our contributions can be summarized as follows:

- We explored query-focused PEFT methods and proposed a method, IDEAL, that tunes instance-level PEFT approaches according to query instructions, enhancing the model's fine-grained instruction-following capabilities.
- We propose to incorporate a query-focused infini-attention module to process long text under low memory resources for QFS tasks. For example, IDEAL with the backbone model LLAMA2-7B can process datasets where the average length of **input tokens is 13,000 on a single 24GB Nvidia GeForce RTX 3090**.
- We performed extensive and rigorous experiments across multiple QFS datasets. IDEAL significantly outperforms other baselines.

2 Methodology

Overview. Given a query and a document, the QFS task aims to generate a summary tailored to this query. Inspired by recent Hypernetwork-based methods (Ivison and Peters, 2022; Zhang et al., 2024), our IDEAL generate instance-level adapters according to the query instruction using an additional HyperNetwork. For long-text QFS datasets, we propose a Query-focused Infini-attention module that can be integrated into IDEAL, enabling the summarization of infinitely long texts under low-memory constraints. In our experiments, we use LLaMA as the underlying model, a popular decoder-only LLM. However, our overall approach can be applied to any generic decoder-only transformer model. In Section 2.1, we first describe the details of IDEAL, including IDEAL$_{Prompt}$, IDEAL$_{PAdapter}$, and IDEAL$_{LoRA}$. Then, Section 2.2 presents the query-focused infini-attention.

2.1 Query-aware HyperExpert Module

Given a dataset with input text pairs containing a query and a document, and outputs in the form of a summary, and a pre-trained LLaMA with an N-layer transformer, IDEAL based on three kinds of PEFT adapters to fine-tune LLaMA to generate query-focused summaries respectively. For example, IDEAL$_{LoRA}$, we place a regular (non-generated) LoRA layer in the first l layers, then we use the hidden representation $\boldsymbol{H}^l_{query}$ of query in l-th layer as the input of a Hypernetwork to generate the LoRA parameters for the last $N - l$ layers.

PEFT approaches. With the growth in model sizes, fine-tuning methods have advanced significantly, modifying only a small number of parameters or adding new ones to a frozen language model for specific tasks (Li and Liang, 2021; Lester et al., 2021; Hu et al., 2021; He et al., 2022; Zhang et al., 2023;). These methods often achieve performance comparable to full model fine-tuning. In this paper, we use three types of PEFT methods, including prompt tuning, parallel adapter, and LoRA, as baselines to investigate our approach.

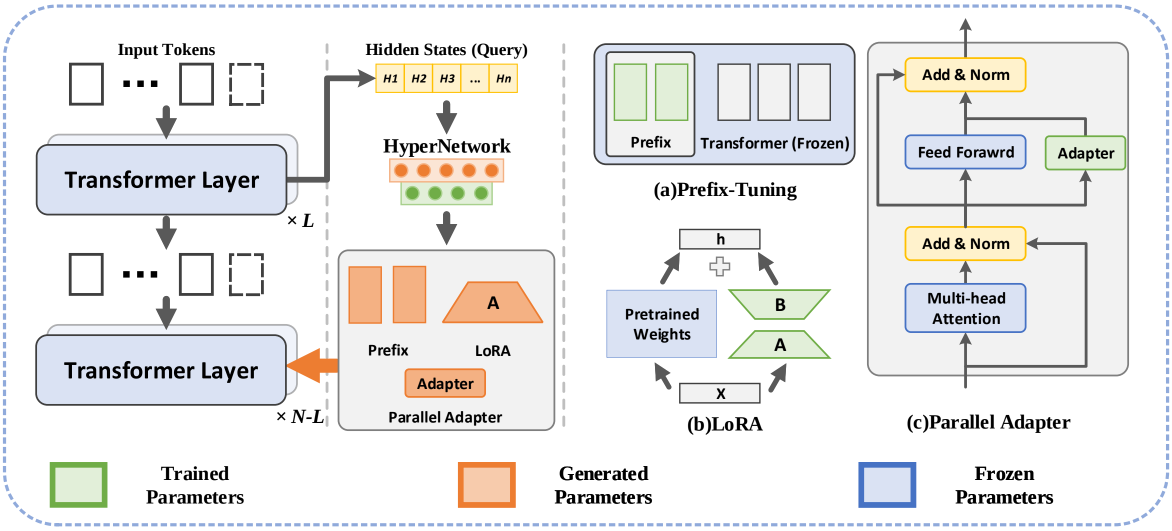

Figure 1: Overview of IDEAL. We place a regular (non-generated) PEFT Adapter layer in the first l layers, and then use the hidden states of query instruction to generate the Adapter's parameters of the last N-l layers.

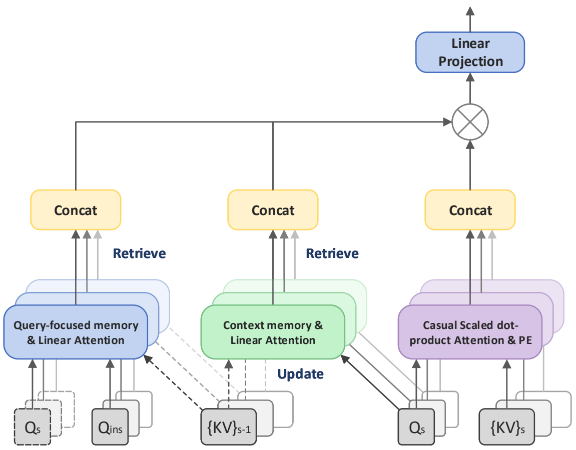

Figure 2: Query-focused Infini-attention has a long-term context memory and a query-focused memory with linear attention for processing infinitely long contexts. KV_{s-1} and KV_s are attention key and values for previous and current input segments, respectively. Q represents the attention queries for current input segment, while Q_{ins} refers to the attention queries for the input query instruction. PE signfies position embeddings.

As shown in Figure 1(a), Prompt tuning can add soft prompts to the hidden states in attention layers to guide model learning and adapt to new tasks, where only the soft prompts are updated during training. LLaMA-Adapter-v1 (Zhang et al., 2023) introduces a zero-initialized attention mechanism into prompt tuning, which adaptively incorporates the knowledge from soft prompts. We use this LLaMA-Adapter-v1 as our prompt tuning baseline.

Parallel adapters (He et al., 2022) aim to incorporate additional learnable networks in parallel with distinct sublayers within the backbone model. To reduce the number of parameters, small bottleneck networks are used as parallel adapters. In transformer-based LLMs, parallel adapters can be applied to both the feedforward and self-attention modules in each transformer block. Hu et al. (2023) conducted experiments showing that applying parallel adapters only to the feedforward module achieves the best results on math reasoning datasets. As shown in Figure 1(c), we also apply parallel adapters only to feedforward module in LLaMA's transformer block.

LoRA (Hu et al., 2021) adds trainable low-rank decomposition matrices in parallel to existing weight matrices (Figure 1(b)). For a pre-trained weight matrix $\boldsymbol{W} \in \mathbb{R}^{d\times k}$, LoRA constrains its update by adding low-rank matrix pairs, resulting in $\boldsymbol{W} + \Delta\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{BA}$, where $\boldsymbol{B} \in \mathbb{R}^{d\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d,k)$. During training, $\boldsymbol{W}$ is frozen while $\boldsymbol{B}$ and $\boldsymbol{A}$ are trainable. LoRA initializes $\boldsymbol{A}$ randomly and $\boldsymbol{B}$ to zero, ensuring that $\Delta\boldsymbol{W} = \boldsymbol{BA}$ starts from zero at the beginning of training, thereby preserving the pre-trained knowledge as much as possible.

Adapter-based HyperExpert. Previous works (Ivison and Peters, 2022; Zhao et al., 2024) indicate that hypernetworks can learn the parameter information of the main neural network under different input scenarios and efficiently adjust the target network's parameters to adapt to this information. We propose generating query-focused adapters conditioned on the query instruction using a hypernetwork.

Our hypernetwork is a bottleneck network that consists of an **encoder** to transform the mean-pooling of the query representation $\boldsymbol{H}_{query}$ into a low-dimensional representation $\boldsymbol{h}$, and a **decoder** to convert $\boldsymbol{h}$ into the parameters of the target adapters. For example, the computation of IDEAL_{LoRA} is as follows:

$$\boldsymbol{h} = dropout(ReLU(\boldsymbol{W}_0 mean(\boldsymbol{H}_{query}) + \boldsymbol{b}_0)) \tag{1}$$

$$\hat{\boldsymbol{A}}_q = \boldsymbol{W}_1\boldsymbol{h} + \boldsymbol{b}_1 \tag{2}$$

$$\hat{\boldsymbol{A}}_k = \boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2 \tag{3}$$

where $\hat{\boldsymbol{A}}_q$ and $\hat{\boldsymbol{A}}_k$ correspond to $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ in self-attention, respectively. We only generate the $\boldsymbol{A}$ matrix in the LoRA module, initializing $\boldsymbol{B}$ to zero and updating it during training as in the original LoRA. This ensures that $\Delta\boldsymbol{W} = \boldsymbol{B}\hat{\boldsymbol{A}}$ starts from zero at the beginning of training. Unlike IDEAL_{LoRA}, IDEAL_{Prompt} and $\text{IDEAL}_{PAdapter}$ generate all the parameters of the target adapters in the required layers.

In addition, each layer that needs to generate the target adapters has its own **encoder**, as shown in Equation 1, and shares a single **decoder**. This allows for generating different parameters for each layer and reduces the number of trainable parameters.

2.2 Query-focused Infini-attention Module

QFS tasks usually involve long documents. However, Transformer-based LLMs can't handle such long texts due to the quadratic complexity of the attention mechanism in terms of both memory usage and computation time. Infini-attention (Munkhdalai et al., 2024) incoporates a compressive memory and a long-term linear attention mechanism into vanilla Transformer block, scale Transformer-based LLMs to extremely long inputs with bounded memory. However, due to the information loss inherent in compressive memory modules, in QFS tasks, the model tends to lose crucial query instruction details and relevant document information after compressing query instruction and very long input documents. To minimize the information loss of query-related details in Infini-attention, we propose compressing the query-related document information into an additional memory block, termed Query-focused Infini-attention.

Similar to Infini-attention (Munkhdalai et al., 2024), the input tokens are segmented to perform standard causal dot-product attention within each segment. Before local attention for current segment is complete, we compress the cached key-value (KV) attention states into two memory blocks. One block maintains the entire context history, while another focuses on query-related information. These compressed memories are then available for subsequent segments to retrieve relevant context.

Fixed length local attention. A key-value (KV) cache is typically used in LLMs for fast and efficient inference. To maintain fine-grained local attention, for each segment, multi-head self-attention $\boldsymbol{\mathcal{A}}_{local} \in \mathbb{R}^{L\times d_{value}}$ is computed with a fixed KV length L in both the training and inference stages using the KV cache. In detail, when the last segment length is less than L, we use the KV cache to extend the length of the current KV states to L for computing the local attention and compress the remaining KV cache into the memory.

Memory update. For the s-th segment with length L, before computing the local attention, we update the full context memory $\boldsymbol{M}^{all}_{s-1} \in \mathbb{R}^{d_{key}\times d_{value}}$ and the query-focused memory $\boldsymbol{M}^{query}_{s-1} \in \mathbb{R}^{d_{key}\times d_{value}}$, and a normalization term $\boldsymbol{z}_{s-1} \in \mathbb{R}^{d_{key}}$ is then used for memory retrieval as follows:

$$\boldsymbol{M}^{all}_{s-1} \leftarrow \boldsymbol{M}^{all}_{s-2} + \sigma(\boldsymbol{K}_{cache})^T\boldsymbol{V}_{cache} \tag{4}$$

$$\boldsymbol{M}^{query}_{s-1} \leftarrow \boldsymbol{M}^{query}_{s-2} + \sigma(\boldsymbol{K}_{cache})^T\hat{\boldsymbol{V}}_{cache} \tag{5}$$

$$\boldsymbol{z}_{s-1} \leftarrow \boldsymbol{z}_{s-2} + \sum_{t=1}^{L}\sigma(\boldsymbol{K}^t_{cache}) \tag{6}$$

where σ is a nonlinear activation function. Following the work of Katharopoulos et al. (2020) and Munkhdalai et al. (2024), we employ element-wise ELU+1 as the activation function (Clevert et al., 2015). The term $\sigma(\boldsymbol{K})^T\boldsymbol{V}$ on the right side of Equation 4 and 5 is referred to as an associative binding operator (Schlag et al., 2020). The query-focused memory $\boldsymbol{M}^{query}_{s-1}$ differs from the full context memory only in the value states $\hat{\boldsymbol{V}}_{cache}$ used within the associative binding operator. We ultilize the query states $\boldsymbol{Q}_{query}$ of query instruction to scale the value states, and keep only query-related information $\hat{\boldsymbol{V}}_{cache}$ as

$$\alpha_i = sigmoid\left(\frac{mean(\boldsymbol{Q}_{query})(\boldsymbol{K}^i_{cache})^T}{\sqrt{d_{model}}}\right) \tag{7}$$

$$\hat{V}_{cache} = \boldsymbol{\alpha} \odot \boldsymbol{V}_{cache}. \tag{8}$$

Here, we use the mean pooling of $\boldsymbol{Q}_{query}$ and the key states to compute a related score for each representation.

Memory retrieval. After updating the memory, we retrieve new content $\boldsymbol{\mathcal{A}}_{all} \in \mathbb{R}^{L \times d_{value}}$ and $\boldsymbol{\mathcal{A}}_{query} \in \mathbb{R}^{L \times d_{value}}$ from the full context memory $\boldsymbol{M}^{all}_{s-1}$ and the query-focused memory $\boldsymbol{M}^{query}_{s-1}$, respectively. This retrieval is performed using the query states $\boldsymbol{Q} \in \mathbb{R}^{L \times d_{key}}$ as follows:

$$\boldsymbol{\mathcal{A}}_{all} = \frac{\sigma(\boldsymbol{Q})\boldsymbol{M}^{all}_{s-1}}{\sigma(\boldsymbol{Q})\boldsymbol{z}_{s-1}} \tag{9}$$

$$\boldsymbol{\mathcal{A}}_{query} = \frac{\sigma(\boldsymbol{Q})\boldsymbol{M}^{query}_{s-1}}{\sigma(\boldsymbol{Q})\boldsymbol{z}_{s-1}} \tag{10}$$

Long-term context injection. First, we apply a linear layer to aggregate $\boldsymbol{\mathcal{A}}_{all}$ and $\boldsymbol{\mathcal{A}}_{query}$. Then, we aggregate the retrieved content and the local attention $\boldsymbol{\mathcal{A}}_{local}$ using a learned gating scalar $\boldsymbol{\beta}$:

$$\gamma = sigmoid(\boldsymbol{W}_g \boldsymbol{\mathcal{A}}_{query}) \tag{11}$$

$$\boldsymbol{\mathcal{A}}_{ret} = \gamma \odot \boldsymbol{\mathcal{A}}_{query} + (1 - \gamma) \odot \boldsymbol{\mathcal{A}}_{all} \tag{12}$$

$$\boldsymbol{\mathcal{A}} = sigmoid(\boldsymbol{\beta}) \odot \boldsymbol{\mathcal{A}}_{ret} + (1 - sigmoid(\boldsymbol{\beta})) \odot \boldsymbol{\mathcal{A}}_{local} \tag{13}$$

where $\boldsymbol{W}_g \in \mathbb{R}^{1 \times d_{value}}$ is a trainable weight that dynamicly merges the two retieved contents. $\boldsymbol{\beta}$ contains a single scalar value per head as training parameter, enabling a learnable trade-off between the long-term and local information flows in the model.

Repeat query instruction. To incorporate query instructions into the model, we concatenate the query instruction with the document as the input of model. During local attention, the query states $\boldsymbol{Q}_{query}$ of the query instruction are utilized to compute query-focused memory within each segment. However, when generating summaries, the retrieved information from memory fails to effectively guide the model in producing summaries that adhere to the query instructions. To address this issue, we employ a straightforward approach: we replicate the query instruction at the end of the document. This ensures that the query instruction is within the window of the local attention computation when generating summaries, enabling the model to accurately generate query-relevant summaries.

3 Experiments

3.1 Datasets

We evaluate our approach on three query-focused summarization datasets: CovidET (Zhan et al., 2022), QMsum (Zhong et al., 2021), SQuALITY (Wang et al., 2022). Different from others, SQuALITY includes multiple summaries for each question. The input documents in the CovidET and QMSum (Golden) datasets have token counts of **228** and **2670**, respectively, when tokenized using the LLama2 tokenizer. In contrast, the QMSum and SQuALITY datasets feature longer input token lengths, with **8071** and **13227** tokens, respectively. The detailed statistics in Appendix A.1.

3.2 Evaluation Metrics

We evaluate the summaries using ROUGE metrics (Lin, 2004), including ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-Lsum. Additionally, we use a BART-base version of BERTScore (Zhang et al., 2020), which leverages BART to compute the similarity between the references and the model's outputs. Specifically, since SQuALITY includes multiple summaries for each question, we report multi-reference scores for all metrics following Wang et al. (2022). We calculate the metrics for each pair of a generated summary and multiple references, then choose the maximum score.

3.3 Implementation Details

We use the pre-trained LLaMA (2-7B, 3-8B) (Touvron et al., 2023) with $N = 32$ transformer layers as the backbone model. For IDEAL$_{Prompt}$, we follow LLaMA-Adapter-v1 (Zhang et al., 2023), adopting a prompt length $K = 10$ and applying prompts to the last 30 layers, with the prompts of the last 15 layers are generated . For IDEAL$_{PAdapter}$, adapters are applied to the first 16 layers and generated for the last 16 layers. For IDEAL$_{LoRA}$, only the $\boldsymbol{A}$ matrix in the LoRA module is generated for the last 16 layers. Additional details can be found in the Appendix A.2.

3.4 Comparison of Methods

We compare our approaches with several fully fine-tuned pretrained language models commonly used for summarization tasks, including Bart-base and Bart-large (Lewis et al., 2019), LED (Beltagy et al., 2020), LED-base-OASum (Yang et al., 2023b), HMNet (Zhu et al., 2020). For long document datasets, we compare our approaches against an extract-then-summarize methods (Wang et al., 2022). Unlimiformer (Bertsch et al., 2024), a retrieval-based approach that augments pretrained language models to handle unlimited-length input.

3.5 Main Results

Tables 1- 2 present the results on QFS datasets. Our approaches achieve the best results and show significant improvements over other baselines. IDEAL consistently outperform the corresponding PEFT Adapters with the same input size. For instance, on CovidET dataset, IDEAL$_{LoRA}$ surpasses the best baseline LoRA by 1.64 ROUGE-L points and 2.36 ROUGE-Lsum points with the same input size of 1.6K.

For the two long document datasets showed in Table 2, IDEAL$_{LoRA}$ with an input length of 8K achieved the best results, while IDEAL$_{LoRA}^{QF_Inf}$ also performed exceptionally well even under limited GPU memory. For example, on QMSum dataset, IDEAL$_{LoRA}^{QF_Inf}$ surpasses all baselines on ROUGE-L and and BERTScore.

The complete results, including ROUGE-1 and ROUGE-2 metrics, are presented in the Appendix A.4.

Models	LC	R-L	R-Lsum	BScore
CovidET Dataset				
Bart-base	1K	21.62	22.17	57.97
Bart-large	1K	21.66	22.24	57.85
LED-base*	4K	-	20.82	-
LED-base-OASum*	4K	-	20.45	-
ChatGPT*	-	15.35	15.36	-
Prompt	768	**23.19**	**23.79**	59.31
PAdapter	768	22.93	23.49	59.00
Lora	768	22.85	23.41	58.93
IDEAL$_{Prompt}$	768	**23.19**	23.71	**59.55**
IDEAL$_{PAdapter}$	768	23.18	**23.79**	59.18
IDEAL$_{LoRA}$	768	**23.28**	23.93	**59.40**
QMsum(Golden) Dataset				
Bart-base	1K	25.21	33.56	55.31
Bart-large	1K	25.25	33.75	55.44
ChatGPT*	-	24.23	24.19	-
Prompt	768	24.26	30.08	56.47
PAdapter	768	26.70	32.76	58.68
Lora	768	26.69	32.44	58.52
	1.6K	27.36	33.71	59.62
IDEAL$_{Prompt}$	768	24.92	30.31	56.76
IDEAL$_{PAdapter}$	768	26.87	33.94	59.35
IDEAL$_{LoRA}$	768	28.35	34.89	59.96
	1.6K	**29.00**	**36.08**	**60.63**
	3K	**29.36**	**36.65**	**60.87**

Table 1: Comparision with baselines on CovidET and QMsum(Golden). **LC** denotes the local context size of model. **R-L**, **R-Lsum**, and **BScore** denote ROUGE-L, ROUGE-Lsum, BERTSCore, respectively. * indicates that experimental results are obtained from related work. We color each row as the **best** and **second best**.

3.6 Ablation Study

Different adapter for IDEAL. As shown in Table 1, we compare the performance of IDEAL on different Adapter with same input size. On the CovidET dataset, the performance differences among the three adapters on IDEAL were minimal. However, on the QMSum(Golden) dataset, IDEAL$_{LoRA}$ outperformed IDEAL$_{PAdapter}$ by 1.48 ROUGE-L points under the same input length of 768. Overall, IDEAL$_{LoRA}$ achieves the best results on four datasets.

The effectiveness of each module in IDEAL$_{LoRA}^{QF_Inf}$. In Table 4, we evaluated the effectiveness of Query-focused Infini-attention through comparative testing. First, we implemented Infini-attention based on LoRA as Lora+Inf and observed significant improvements compared to LoRA alone under the same GPU memory constraints, with increases of 1.55 and 1.33 points in ROUGE-L and ROUGE-Lsum on QMSum dataset, respectively. These results indicate that compressing the key-value states of historical segments enables summarization of long documents within limited GPU memory. Furthermore, we enhanced IDEAL$_{LoRA}$ with Infini-attention, achieving better results than Lora+Inf in ROUGE-L. The IDEAL$_{LoRA}$ method integrated with Query-focused Infini-attention as IDEAL$_{LoRA}^{QF_Inf}$ outperformed both IDEAL$_{LoRA}$+Inf and Lora+Inf in all metrics, demonstrating that our proposed Query-focused Infini-attention effectively compresses query-related information. For the IDEAL$_{LoRA}$+Inf method, we observed a significant decline in all metrics after removing the repeated query instruction at the end of the input document, demonstrating the necessity of repeating the query instruction.

Models	LC	R-L	R-Lsum	BScore
QMSum Dataset				
Bart-base	1K	20.37	27.46	51.74
Bart-large	1K	20.02	**27.52**	51.83
LED-base*	4K	-	25.68	-
LED-base-OASum*	4K	-	26.67	-
ChatGPT*	-	17.81	18.81	-
Bart+ Unlimiformer*	1/-K	19.9	-	-
IDEAL$_{LoRA}$	8K	**22.59**	**31.30**	**57.35**
IDEAL$_{LoRA}^{QF_Inf}$	0.8/6K	**22.16**	27.05	**55.56**
SQuALITY Dataset				
Bart-base	1K	20.49	34.34	54.41
Bart-large	1K	20.97	**36.11**	54.85
LED-base*	4K	-	34.47	-
LED-base-OASum*	4K	-	35.14	-
Bart-Large*	1K	20.8	-	-
Bart-Large+ DPR*	1K	21.0	-	-
ChatGPT*	-	18.45	22.56	-
IDEAL$_{LoRA}$	8K	**24.25**	**41.72**	**59.48**
IDEAL$_{LoRA}^{QF_Inf}$	1.6/9K	**21.49**	34.86	**56.08**

Table 2: Comparision with baselines on QMSum and SQuALITY. 0.8/6K represents the local text size and the max input length, respectively.

Models	r/bs	Params(M)	R-L
Prompt	-	1.2	24.26
PAdapter	16	4.3	26.70
LoRA	8	12.3	26.69
	16	24.5	26.37
IDEAL$_{Prompt}$	-	7.2	24.92
IDEAL$_{PAdapter}$	16	15.2	26.87
	32	25.8	27.21
	64	47.0	27.66
	128	89.5	27.89
IDEAL$_{LoRA}$	8	24.5	**28.35**

Table 3: Trainable parameters comparison on **QMSum(Golden)** dataset with **768** input size. **r/bs** denote the rank in LoRA or the bottle-neck size in Parallel Adapter. **Params(M)** is the total size of trainable parameters in millions.

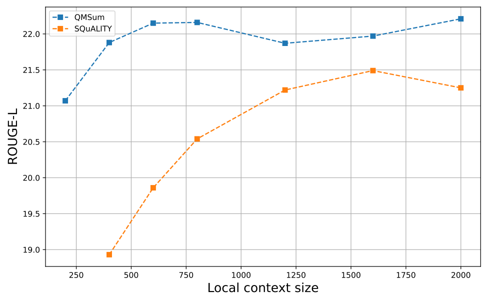

Figure 3: Performance with respect to the different local context size of IDEAL$_{LoRA}^{QF_Inf}$.

3.7 Indepth Analysis

Performance of low memory IDEAL. IDEAL$_{LoRA}$ consistently demonstrates improved performance as input length increases. However, this comes at the cost of increased GPU memory consumption. Table 4 illustrates this trade-off, showcasing IDEAL$_{LoRA}$ performance on input lengths of 1.6K, 3.8K, and 8K, requiring 24G, 40G, and 80G of memory, respectively. In contrast to IDEAL$_{LoRA}$, our proposed IDEAL$_{LoRA}^{QF_Inf}$ exhibits memory efficiency when handling long inputs. IDEAL$_{LoRA}^{QF_Inf}$ maintains a consistent memory footprint 24G regardless of input length. Notably, on the QMsum dataset, IDEAL$_{LoRA}^{QF_Inf}$ outperforms IDEAL$_{LoRA}$ with an input length of 1.6K on all metrics within a same 24GB memory constraint. Moreover, it surpasses IDEAL$_{LoRA}$ with an input length of 3.8K in 40GB memory on the ROUGE-L metric and achieves performance close to IDEAL$_{LoRA}$ with an input length of 8K in 80GB memory.

Trainable parameters comparison. In Table 3, we compare the performance of different IDEAL HyperExperts under the same parameter count. The Prompt-tuning method can adjusts parameter count only by controlling prompt length, with experiments from Hu et al. (2023) indicating optimal performance at a prompt length of 10. Despite having the fewest trainable parameters, its performance on the QMSum(Golden) dataset is the lowest. With the same parameter count, LoRA with a rank of 16 still significantly underperforms compared to IDEAL$_{LoRA}$, highlighting the effectiveness of HyperExpert. IDEAL$_{PAdapter}$ can improve performance by increasing the bottleneck size, but even with 89.5M parameters, it is still inferior to IDEAL$_{LoRA}$ with 24.5M parameters. Overall, IDEAL$_{LoRA}$ achieves the best performance and parameter efficiency.

Local context size of IDEAL$_{LoRA}^{QF_Inf}$. Figure 3 presents the performance of IDEAL$_{LoRA}^{QF_{Inf}}$ under varying local context sizes (LC). On the QMSum

Models	QMSum Dataset				SQuALITY Dataset			
	LC	R-L	R-Lsum	BScore	LC	R-L	R-Lsum	BScore
Lora	1.6K	19.58	25.25	53.76	1.6K	20.73	35.41	55.97
IDEAL_{LoRA}	1.6K	19.71	26.27	54.30	1.6K	22.16	35.73	56.50
	3.8K	21.62	28.46	56.00	3.8K	22.54	37.54	57.42
	8K	22.59	31.30	57.35	8K	24.25	41.72	59.48
LoRA+Inf	0.8/6K	21.13	26.58	55.34	1.6/9K	20.59	34.76	55.21
IDEAL_{LoRA}+Inf	0.8/6K	21.76	26.16	54.97	1.6/9K	21.68	34.81	55.72
IDEAL_{LoRA}+Inf w/o ReQ	0.8/6K	16.57	20.40	50.71	1.6/9K	17.89	30.62	50.52
$\text{IDEAL}_{LoRA}^{QF\_Inf}$	0.8/6K	22.16	27.05	55.56	1.6/9K	21.49	34.86	56.08

Table 4: Comparing $\text{IDEAL}_{LoRA}^{QF\_Inf}$ with Infini-attention based methods and IDEAL_{LoRA} with different input size. LoRA+Inf and IDEAL_{LoRA}+Inf denote the incorporation of Infini-attention into LoRA and IDEAL_{LoRA}, respectively. w/o ReQ indicates that the query instruction is not repeated at the end of the input document.

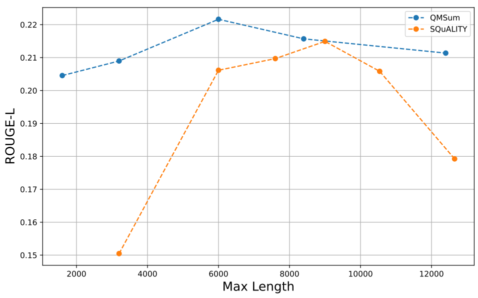

Figure 4: Performance with respect to the different max input length of $\text{IDEAL}_{LoRA}^{QF\_Inf}$.

dataset, the model exhibits stable performance when LC is beyond 400, achieving nearly the best overall performance at LC=800. Similarly, on the SQuALITY dataset, the optimal LC is observed at 1.6K. These findings indicate that $\text{IDEAL}_{LoRA}^{QF_Inf}$ differs from IDEAL_{LoRA}, the limited memory for the former is enough to handle extremely long inputs.

Max input length of $\text{IDEAL}_{LoRA}^{QF\_Inf}$. Table 4 presents the optimal input length for $\text{IDEAL}_{LoRA}^{QF\_Inf}$ on the QMsum and SQuALITY datasets. The results suggest that information relevant to the query in the QMsum dataset is primarily concentrated within the first 6000 tokens, while in the SQuALITY dataset, the relevant information is more evenly distributed throughout the document.

4 Related Works

Query-focused Summarization. Tan et al. (2020) and Yang et al. (2023b) address QFS by prepending the query or aspect to the input document and fine-tuning pre-trained models in an end-to-end manner. Zhong et al. (2021), Wang et al. (2022), and Amar et al. (2023) employ extract-then-summarize strategies that use a filter model to extract key parts of the document based on the query, then fitting the shorter text into a summarizer. Yang et al. (2023a) reveal that the performance of ChatGPT is comparable to traditional fine-tuning methods in terms of ROUGE scores on QFS tasks.

Long-context Transformers. Unlimiformer (Bertsch et al., 2024) enhances pre-trained models like BART (Lewis et al., 2019) to handle unlimited inputs without additional learned weights by employing a retrieval-based long-context method. Infini-transformer (Munkhdalai et al., 2024) integrates long-term context compressive memory into vanilla transformers, enabling Transformer-based LLMs to scale to infinitely long contexts after full continual pre-training. Unlike Infini-transformer, we explore the compressive memory method on adapter-based PEFT of LLMs and design a query-focused infini-attention for QFS tasks.

5 Conclusion

In this paper, we propose IDEAL, an efficient query-aware adaptation method on LLMs for QFS tasks, which consists of two modules: Query-aware HyperExpert and Query-focused Infini-attention. The two modules enable LLMs to achieve fine-grained query-LLM alignment efficiently and have the ability to handle lengthy documents.

Limitations

Due to the absence of longer QFS datasets currently available, we explored IDEAL only on datasets with input lengths around 10k. However, it is necessary to validate IDEAL on datasets with longer input documents, such as performing QFS tasks across entire books. Further validation and optimization of the IDEAL method on book-length inputs would be both interesting and meaningful.

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*.

Shmuel Amar, Liat Schiff, Ori Ernst, Asi Shefer, Ori Shapira, and Ido Dagan. 2023. OpenAsp: A Benchmark for Multi-document Open Aspect-based Summarization. *arXiv preprint*. ArXiv:2312.04440 [cs].

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

Hal Daumé III. 2009. Bayesian query-focused summarization. *arXiv preprint arXiv:0907.1814*.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

David Ha, Andrew M Dai, and Quoc V Le. 2016. Hypernetworks. In *International Conference on Learning Representations*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv preprint*. ArXiv:2304.01933 [cs].

Hamish Ivison and Matthew E. Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task NLP. *arXiv preprint*. ArXiv:2203.08304 [cs].

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv preprint*. ArXiv:1910.13461 [cs, stat].

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. 2020. Learning associative inference using fast weight memory. In *International Conference on Learning Representations*.

Bowen Tan, Lianhui Qin, Eric P. Xing, and Zhiting Hu. 2020. Summarizing Text on Any Aspects: A Knowledge-Informed Weakly-Supervised Approach. *arXiv preprint*. ArXiv:2010.06792 [cs].

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R. Bowman. 2022. SQuALITY: Building a Long-Document Summarization Dataset the Hard Way. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1139–1156, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. 2019. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11293–11302.

Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023a. Exploring the Limits of ChatGPT for Query or Aspect-based Text Summarization. *arXiv preprint*. ArXiv:2302.08081 [cs].

Xianjun Yang, Kaiqiang Song, Sangwoo Cho, Xiaoyang Wang, Xiaoman Pan, Linda Petzold, and Dong Yu. 2023b. OASum: Large-Scale Open Domain Aspect-based Summarization. *arXiv preprint*. ArXiv:2212.09233 [cs].

Hongli Zhan, Tiberiu Sosea, Cornelia Caragea, and Junyi Jessy Li. 2022. Why do you feel this way? summarizing triggers of emotions in social media posts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9436–9453.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention. *arXiv preprint*. ArXiv:2303.16199 [cs].

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wenqiao Zhang, Tianwei Lin, Jiang Liu, Fangxun Shu, Haoyuan Li, Lei Zhang, He Wanggui, Hao Zhou, Zheqi Lv, Hao Jiang, Juncheng Li, Siliang Tang, and Yueting Zhuang. 2024. HyperLLaVA: Dynamic Visual and Language Expert Tuning for Multimodal Large Language Models. *arXiv preprint*. ArXiv:2403.13447 [cs].

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. 2024. HyperMoE: Paying Attention to Unselected Experts in Mixture of Experts via Dynamic Transfer. *arXiv preprint*. ArXiv:2402.12656 [cs].

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online. Association for Computational Linguistics.

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 194–203.

A Appendix

A.1 Dataset statistics

A.2 Implementation Dtails

All LLaMA-based models in our experiments use Automatic Mixed Precision, with 16-bit for frozen parameters and 32-bit for trainable parameters to conserve memory. Additionally, we employ FlashAttention2 (Dao, 2024) to accelerate model training and inference for LLaMA-based models. All models in our experiments can be trained on at least a single 24GB Nvidia GeForce RTX 3090, except for the large local context size setting for long documents. The details of GPU requirements for different local context sizes are shown in Table 6. During the generation stage, we adopt top-p sampling as the default decoding method with a temperature of 0.1 and a top-p value of 0.75.

A.3 GPU Requirements

A.4 Complete Results

Type	Dataset	Domain	#Instances	#Input Tk.	#Output Tk.	#Queries|Aspects
Query	QMSum	Meeting	1808	13227(2670*)	88	1566
	SQuALITY	Story	625	8071	306	437
Aspect	CovidET	Reddit	7122	228	32	7

Table 5: Statistics of query/aspect-based summarization datasets.**#Instances** represents the total number of (document, summary) pairs in the corresponding dataset. **#Instances** and **#Input Tk.** denote the number of input and output token lengths under the Llama2 tokenizer, respectively. **#Queries|Aspects** indicates the number of unique queries or aspects appearing in the dataset. 2670* represents the number of input tokens for QMsum(Golden).

<table>
<tr><th>Models</th><th>LC</th><th>GPU</th></tr>
<tr><td>Bart-base
Bart-large</td><td>$\leq$ 1K</td><td rowspan="4">3090 24G</td></tr>
<tr><td>Prompt
PAdapter</td><td>$\leq$ 0.8K</td></tr>
<tr><td>LoRA
IDEAL$_{LoRA}$</td><td>$\leq$ 1.6K</td></tr>
<tr><td>Inf+LoRA
Inf+IDEAL$_{LoRA}$
IDEAL$_{LoRA}^{QF_Inf}$</td><td>$\leq$ 1.2K
$\leq$ 1.1K
$\leq$ 0.8K</td></tr>
<tr><td>Inf+LoRA
Inf+IDEAL$_{LoRA}$
IDEAL$_{LoRA}^{QF_Inf}$</td><td>$\leq$ 2.1K</td><td rowspan="2">A100 40G</td></tr>
<tr><td>IDEAL$_{LoRA}$</td><td>$\leq$ 3.8K</td></tr>
<tr><td>IDEAL$_{LoRA}$</td><td>$\leq$ 8K</td><td>A800 80G</td></tr>
</table>

Table 6: GPU requirements in our experiments. For all LoRA-based methods, we can extend the local context size using Flash-attention2.

Models	R-1	R-2	R-L	R-Lsum	BScore
Bart-base	27.28	7.50	21.62	22.17	57.97
Bart-large	27.54	7.72	21.66	22.24	57.85
LED-base*	26.19	6.85	-	20.82	-
LED-base-OASum*	25.61	6.58	-	20.45	-
ChatGPT*	20.81	3.99	15.35	15.36	-
Prompt	28.71	8.58	23.19	23.79	59.31
PAdapter	29.18	8.69	22.93	23.49	59.00
Lora	28.81	8.54	22.85	23.41	58.93
IDEAL$_{Prompt}$	28.55	8.56	23.19	23.71	59.55
IDEAL$_{PAdapter}$	29.40	8.92	23.18	23.79	59.18
IDEAL$_{LoRA}$	29.40	8.84	23.28	23.93	59.40

Table 7: CovidET

Models	Input Size	R-1	R-2	R-L	R-Lsum	BScore
Bart-base	1K	38.32	13.61	25.21	33.56	55.31
Bart-large	1K	38.49	14.26	25.25	33.75	55.44
ChatGPT*		36.83	12.78	24.23	24.19	-
Prompt	768	34.06	11.96	24.26	30.08	56.47
PAdapter	768	37.10	14.13	26.70	32.76	58.68
Lora	768	36.57	14.23	26.69	32.44	58.52
Lora	1.6K	38.05	14.59	27.36	33.71	59.62
IDEAL$_{Prompt}$	768	34.48	12.22	24.92	30.31	56.76
IDEAL$_{PAdapter}$	768	38.50	14.38	26.87	33.94	59.35
IDEAL$_{LoRA}$	768	39.26	15.44	28.35	34.89	59.96
IDEAL$_{LoRA}$	1.6K	40.82	16.61	29.00	36.08	60.63
IDEAL$_{LoRA}$	3K	41.61	17.07	29.36	36.65	60.87

Table 8: QMsum(Golden)

Models	Input Size	R-1	R-2	R-L	R-Lsum	BScore
Bart-base	1K	31.72	7.98	20.37	27.46	51.74
Bart-large	1K	31.76	7.76	20.02	27.52	51.83
LED-base*	4K	29.52	7.00	-	25.68	-
LED-base-OASum*	4K	30.30	7.56	-	26.67	-
ChatGPT*		28.34	8.74	17.81	18.81	-
Bart+Unlimiformer*	1K	30.9	8.0	19.9	-	-
Lora	1.6K	28.74	7.54	19.58	25.25	53.76
Inf+LoRA	0.8K/6K	30.49	7.95	21.13	26.58	55.34
IDEAL$_{LoRA}$	1.6K	29.94	8.05	19.71	26.27	54.30
IDEAL$_{LoRA}$	3.8K	32.69	9.28	21.62	28.46	56.00
IDEAL$_{LoRA}$	8K	35.50	10.62	22.59	31.30	57.35
Inf+IDEAL$_{LoRA}$	0.8K/6K	30.44	8.05	21.76	26.16	54.97
IDEAL$_{LoRA}^{QF_Inf}$	0.8K/6K	31.49	8.67	22.16	27.05	55.56

Table 9: QMsum

Models	**Input Size**	**R-1**	**R-2**	**R-L**	**R-Lsum**	**BScore**
Bart-base	1K	36.93	8.57	20.49	34.34	54.41
Bart-large	1K	38.58	9.81	20.97	36.11	54.85
LED-base*	4K	36.78	8.31	-	34.47	-
LED-base-OASum*	4K	37.6	8.81	-	35.14	-
Bart-Large*	1K	40.2	10.4	20.8	-	-
Bart-Large+DPR*	1K	41.5	11.4	21.0	-	-
ChatGPT*		37.02	8.19	18.45	22.56	-
Lora	1.6K	38.11	8.65	20.73	35.41	55.97
Inf+LoRA	1.6K/9K	37.06	8.24	20.59	34.76	55.21
IDEAL$_{LoRA}$	1.6K	38.26	9.45	22.16	35.73	56.50
IDEAL$_{LoRA}$	3.8K	40.13	10.63	22.54	37.54	57.42
IDEAL$_{LoRA}$	8K	44.59	12.87	24.25	41.72	59.48
Inf+IDEAL$_{LoRA}$	1.6K/9K	37.13	8.77	21.68	34.81	55.72
IDEAL$_{LoRA}^{QF_Inf}$	1.6K/9K	37.36	8.74	21.49	34.86	56.08

Table 10: SQuALITY