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. 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, *Efficiently Fine-grained Query-LLM Alignment* and *Lengthy Document Summarization*, 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) is 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 attracts increasing attention in the textual summarization community.

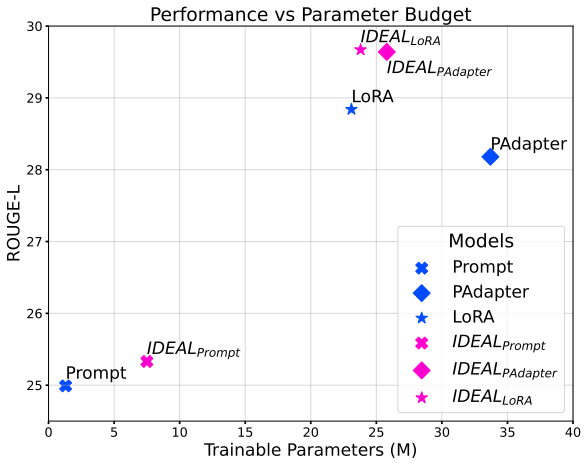


Figure 1: Our Query-aware HyperExperts outperform the corresponding PEFT methods on QFS tasks using a comparable amount of trainable parameters.

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. However, real-world QFS tasks require a comprehensive and in-depth understanding of complex and lengthy documents to generate high-quality, relevant summaries. 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 com-

monly 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, training the fully fine-tuned model for each downstream task becomes very inefficient. Moreover, the simple approach of concatenating the query to the input document proves insufficient for effectively guiding the model to focus on the query while generating the summary. Due to the small proportion of the query length in the overall input (e.g., in the QMSum (Zhong et al., 2021) dataset, the average token counts for queries and documents are 15 and 13,227, respectively), the query's control over the model tends to be relatively weak in attention-based models during summary generation. Therefore, the foundation of more effective LLM-based QFS lies in achieving fine-grained query-LLM alignment through efficient learning. (ii) **Lengthy Document Summarization**, QFS tasks usually involve long documents. However, self-attention-based LLMs have been shown to struggle with handling such long text inputs due to the quadratic complexity of the attention mechanism in terms of both memory usage and computation time. How to process lengthy documents under limited memory 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 [1] for ideal QFS, which consists of two modules: **Query-aware HyperExpert** and **Query-focused Infini-attention**, achieving the two indispensable characteristics, respectively.

The Query-aware HyperExpert (Figure 2) leverages the parameter-efficient fine-tuning (PEFT) (Mangrulkar et al., 2022) strategies that enable a pre-trained LLM to perform a new QFS 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 instance-level PEFT Adapter's parameters 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 instance. By doing so, efficient and fine-grained query-LLM alignment can be achieved. Notably, we develop three types of HyperExpert, include IDEAL$_{Prompt}$, IDEAL$_{PAdapter}$, and IDEAL$_{LoRA}$ based on Prompt-tuning (Lester et al., 2021), Parallel Adapter (He et al., 2022a), and Low-Rank Adaptation (LoRA) (Hu et al., 2021) respectively.

To enable Transformer-based LLMs to handle extremely long inputs for QFS tasks under limited memory constraints, we propose a Query-focused Infini-Attention (Figure 3) module that can be seamlessly integrated into the Query-aware HyperExpert framework. The Query-focused Infini-Attention builds upon the Infini-Attention mechanism (Munkhdalai et al., 2024), which enhances the standard Transformer architecture by introducing compressive memory and a long-term linear attention mechanism. Specifically designed for QFS tasks, the Query-focused Infini-Attention incorporates a Query-focused memory block to preserve critical query-related document details, effectively mitigating the loss of essential information during the compression of query instructions and extremely long input documents.

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

2.1 Query-aware HyperExpert Module

Given a dataset with input text pairs containing a query and a document, outputs in the form of a summary, and a pre-trained LLaMA with an N-layer transformer, IDEAL used three kinds of PEFT

[1] Code:https://anonymous.4open.science/r/IDEAL-Summary-04EA

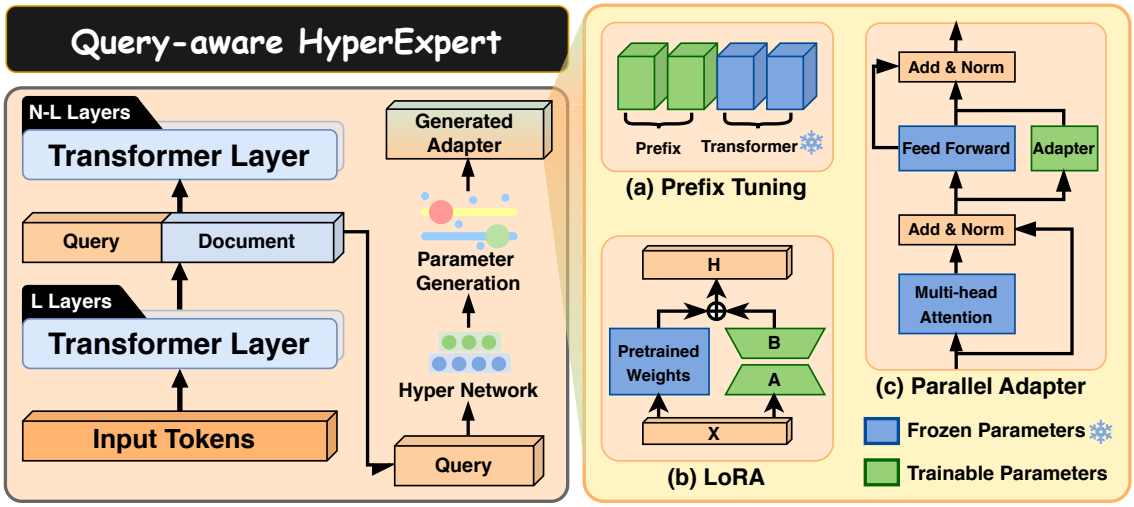


Figure 2: Overview of IDEAL. We place a regular (non-generated) PEFT Adapter layer in the first l layers, and then use the hidden states of the query instruction to generate the Adapter's parameters of the last N-l layers.

adapters to fine-tune LLaMA to generate query-focused summaries respectively. For example, IDEAL$_{LoRA}$, we place a regular (non-generated) LoRA module 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.

2.1.1 PEFT approaches

As shown in Figure 2(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., 2022a) aim to incorporate additional learnable networks in parallel with distinct sublayers in 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 2(c), we also apply parallel adapters only to feedforward modules in LLaMA's transformer block.

LoRA (Hu et al., 2021) adds trainable low-rank decomposition matrices in parallel to existing weight matrices (Figure 2(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 $\boldsymbol{W} + \Delta\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{B}\boldsymbol{A}$, 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 $\Delta\boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}$ starts from zero at the beginning of training, thereby preserving the pre-trained knowledge as much as possible.

2.1.2 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 leveraging a Hypernetwork to generate adapters conditioned on query instructions, enhancing the model's query-focused capabilities.

Our HyperExpert is a Hypernetwork that consists of an **encoder** that transforms the mean-pooling of the query representation $\boldsymbol{H}_{query}$ into a low-dimensional representation $\boldsymbol{h} \in \mathbb{R}^{b}$, and a **decoder** that converts $\boldsymbol{h}$ into the parameters of the target PEFT adapters. The **encoder** is consistent across all three types of HyperExpert and is computed as follows:

$$\boldsymbol{h} = \mathcal{D}_r(\mathrm{ReLU}(\boldsymbol{W}_0\mathrm{mean}(\boldsymbol{H}_{query}) + \boldsymbol{b}_0)) \quad (1)$$

where $\mathcal{D}_r$ denotes dropout.

The **decoder** of HyperExpert varies based on the structure of the target PEFT adapters.

Decoder of IDEAL$_{LoRA}$. The decoder uses linear layers to transform the compressed representation $\boldsymbol{h}$ into the LoRA matrix for self-attention. For instance, the computations of LoRA matrix for $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ in self-attention are as follows:

$$\hat{\boldsymbol{A}}_q = \boldsymbol{W}_1\boldsymbol{h} + \boldsymbol{b}_1, \hat{\boldsymbol{A}}_k = \boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2. \quad (2)$$

We only generate the $\boldsymbol{A}$ matrix in the LoRA module, initialize $\boldsymbol{B}$ to zero, and update it during training as the original LoRA setup. This ensures that $\Delta\boldsymbol{W} = \boldsymbol{B}\hat{\boldsymbol{A}}$ starts from zero at the beginning of training.

Decoder of IDEAL$_{Prompt}$. Prompt tuning includes an additional prompt embedding $\boldsymbol{E} \in \mathbb{R}^{K\times d}$ in each attention layer. The **decoder** is a linear layer that generates the prompt embedding $\boldsymbol{E}$ from the compressed query representation $\boldsymbol{h}$ as $\hat{\boldsymbol{E}} = \boldsymbol{W}_p\boldsymbol{h} + \boldsymbol{b}_p$, where $\boldsymbol{W}_p \in \mathbb{R}^{(K\times d)\times b}$. Here, K is the prompt embedding length, and d is the dimension of the transformer's hidden states.

Decoder of IDEAL$_{PAdapter}$. The parallel adapter is a bottleneck network composed of two linear layers. Therefore, our **decoder** uses two linear layers to generate the weights for the parallel adapter as follows:

$$\boldsymbol{L}_1 = \boldsymbol{W}_{l1}\boldsymbol{h} + \boldsymbol{b}_{l1}, \boldsymbol{L}_2 = \boldsymbol{W}_{l2}\boldsymbol{h} + \boldsymbol{b}_{l2}. \quad (3)$$

In terms of implementation details, adapter parameters can be generated in two ways: **parallel generation** and **sequential generation**. **Parallel generation** utilizes the query representation of the l-th layer to produce the parameters for the subsequent $N - l$ layers in one step. In contrast, **sequential generation** uses the query representation of the l-th layer to generate the adapter parameters for the $(l + 1)$-th layer iteratively. In HyperExpert, the number of encoder layers can correspond to the number of layers that parameters are generated or a single shared layer can be used. For the decoder layers, we adopt a shared-layer approach to reduce the parameter overhead.

2.2 Query-focused Infini-attention Module

The Query-focused Infini-Attention mechanism consists of several key steps. The first step is **Fixed-length Local Attention**, designed to maximize the utilization of the capabilities of the self-attention mechanism. The input tokens are segmented for long context documents to perform standard causal dot-product attention within each segment. In both training and inference, we cache the previous segment's key-value (KV) attention states to pad the local attention KV states to a fixed length. The next step is **Compression and Retrieval of Memory**. Before completing the local attention for the current segment, the cached KV attention states are compressed into two memory blocks: one preserving the entire historical context and another retaining query-related information. These compressed memories are subsequently used to retrieve relevant context for the following segments. The final step is **Repeated Query Instruction**, which ensures accurate query-focused summarization during inference. To achieve this, we prepend and append the query instruction to the document. The prepended query instruction facilitates the compression of historical memory, while the appended query instruction ensures that local attention adheres to the full query instruction. For a detailed description of the Query-focused Infini-attention implementation, we refer the reader to Appendix A.

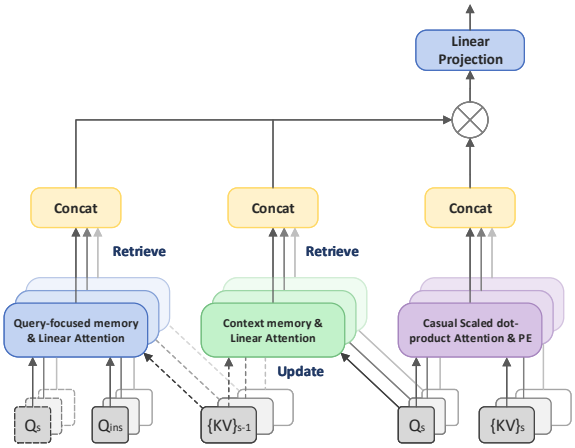


Figure 3: Query-focused Infini-attention has a long-term context memory and a query-focused memory with linear attention for processing infinitely long contexts. KV_{s-1} and KV_s are attention keys and values for previous and current input segments, respectively. Q represents the attention queries for the current input segment, while Q_{ins} refers to the attention queries for the input query instruction. PE signifies position embeddings.

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). Detailed dataset statistics are

provided in the Appendix B.1.

3.2 Evaluation Metrics

Reference-based Evaluation Metric We evaluate the summaries using ROUGE metrics (Lin, 2004), including ROUGE-1, ROUGE-2, the sentence-level ROUGE-L, and the summary-level ROUGE-Lsum. Additionally, we use a Roberta-large version of BERTScore (Zhang et al., 2020), which leverages Roberta-large 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.

LLM-based Evaluators Recent studies, such as G-Eval (Liu et al., 2023) and GPTRank (Liu et al., 2024), demonstrate that LLM-based evaluators outperform reference-based metrics with higher alignment to human judgments. Using GPTRank, we evaluate our method by prompting the LLM to first generate an **explanation** and then provide a **ranking** for a list of candidate summaries corresponding to the same source article.

3.3 Implementation Details

We use the pre-trained LLaMA (2-7B, 3.1-8B) (Touvron et al., 2023) with $N = 32$ transformer layers as the backbone model. LLaMA3.1-8B served as our primary baseline model. However, for low-memory experiments with Query-focused Infini-attention, the more memory-efficient LLaMA2-7B was employed. Additional details can be found in Appendix B.2.

3.4 Comparison of Methods

Our approach is compared against several fully fine-tuned pre-trained language models frequently employed for summarization, including BART-large (Lewis et al., 2019), LED-base-OASum (Yang et al., 2023b), and HMNet (Zhu et al., 2020) (with results reported by Zhong et al. (2021)). We further evaluate the latest iteration of LLaMA (3.1 8B) and three corresponding PEFT-based baselines: Prompt, PAdapter, and LoRA. Comparisons are also conducted with ChatGPT (Yang et al., 2023a) and, using 50 randomly sampled instances per dataset, with GPT-4O (version of 2024-08-06).

For long document datasets, we compare our methods against several Retrieval-Augmented Generation (RAG) approaches, including BART-large + DPR (Wang et al., 2022), HMNet + Locator (Zhong et al., 2021), and Qontsum (Sotudeh and Goharian, 2023). We also include the abstractive summarizer SegEnc (Vig et al., 2022), its pre-training framework Socratic Pretraining (Pagnoni et al., 2023), and Unlimiformer (Bertsch et al., 2024), a retrieval-based method for handling unlimited-length inputs.

3.5 Main Results of IDEAL

Models	LC	R-1	R-2	R-L	R-Lsum	BScore
CovidET Dataset						
Bart-large	1K	27.54	7.72	21.66	22.24	88.61
LED-base-OASum*	4K	25.61	6.58	-	20.45	-
ChatGPT*	-	20.81	3.99	15.35	15.36	-
GPT-4O	1K	16.57	2.59	12.46	12.67	87.06
Llama3.1-8B	1K	12.85	2.12	9.32	11.04	84.55
Prompt	1K	29.18	8.77	**23.64**	**24.15**	**89.20**
PAdapter	1K	29.37	**8.84**	23.20	23.86	89.06
Lora	1K	29.00	8.43	22.79	23.43	89.00
IDEAL$_{Prompt}$	1K	29.10	**9.01**	**23.65**	**24.18**	**89.26**
IDEAL$_{PAdapter}$	1K	**29.51**	8.78	23.21	23.80	89.07
IDEAL$_{LoRA}$	1K	**29.62**	**8.84**	23.40	24.06	89.12
QMsum(Golden) Dataset						
Bart-large	1K	38.49	14.26	25.25	33.75	86.38
HMNet*	-	36.06	11.36	-	31.27	-
ChatGPT*	-	36.83	12.78	24.23	24.19	-
GPT-4O	3K	33.31	9.01	19.88	28.80	85.01
Llama3.1-8B	3K	20.51	6.76	13.94	18.44	82.39
Prompt	3K	34.81	13.33	24.99	30.73	86.69
PAdapter	3K	39.41	15.77	28.18	34.98	87.54
Lora	3K	40.69	16.11	28.84	36.18	87.71
IDEAL$_{Prompt}$	3K	35.58	13.75	25.33	31.52	86.73
IDEAL$_{PAdapter}$	3K	**40.79**	**17.24**	**29.64**	**36.55**	**87.80**
IDEAL$_{LoRA}$	3K	**40.85**	**16.89**	**29.67**	**36.66**	**87.83**

Table 1: Comparison with baselines on CovidET and QMsum(Golden). **LC** denotes the local context size of the 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**.

Tables 1- 2 present the results on QFS datasets. Our approaches achieve the best results overall. IDEAL consistently outperforms the corresponding PEFT Adapters. For instance, on the QMsum(Golden) dataset, IDEAL$_{PAdapter}$ surpasses PAdapter by 1.46 (5.2%) ROUGE-L points and 1.57 (4.5%) ROUGE-Lsum points with the same input size of 3K.

For the two long document datasets shown in Table 2, IDEAL$_{LoRA}$ with an input length of 8K

Models	LC	R-1	R-2	R-L	R-Lsum	BScore
SQuALITY Dataset						
Bart-large	1K	38.58	9.81	20.97	36.11	84.81
LED-base-OASum*	4K	37.6	8.81	-	35.14	-
Bart-Large+ DPR*	1/-K	41.5	11.4	21.0	-	85.5
ChatGPT*	-	37.02	8.19	18.45	22.56	-
GPT-4O	8K	40.04	9.59	20.85	36.68	85.79
Llama3.1-8B	8K	36.84	9.34	20.14	34.17	84.39
SegEnc*	-	45.68	14.51	22.47	-	85.86
+ Socratic Pret.*	-	46.31	14.80	22.76	-	86.04
Qontsum*	-	45.76	14.27	24.14	-	86.07
Prompt	8K	36.81	10.72	23.69	33.26	85.23
PAdapter	8K	43.86	13.13	24.21	40.83	86.55
Lora	8K	44.47	13.23	24.32	**41.46**	86.63
IDEAL$_{Prompt}$	8K	37.40	11.23	23.93	34.44	85.56
IDEAL$_{PAdapter}$	8K	44.37	13.43	**24.76**	**41.47**	**86.66**
IDEAL$_{LoRA}$	8K	43.87	13.86	**25.54**	40.99	**86.86**
QMSum Dataset						
Bart-large	1K	31.76	7.76	20.02	27.52	85.22
LED-base-OASum*	4K	30.30	7.56	-	26.67	-
HMNet+Locator*	-	32.29	8.67	-	28.17	-
ChatGPT*	-	28.34	8.74	17.81	18.81	-
GPT-4O	16K	29.53	7.51	17.50	25.67	84.48
Llama3.1-8B	16K	21.34	6.26	14.39	19.03	82.69
Bart+ Unlimiformer*	1K	30.9	8.0	19.9	-	-
SegEnc*	-	37.05	13.03	-	32.62	87.44
+ Socratic Pret.*	-	38.06	13.74	-	33.51	87.63
Qontsum*	-	**38.42**	13.50	-	**34.03**	87.72
Prompt	8K	30.52	9.77	21.50	26.33	85.89
PAdapter	8K	36.03	12.61	24.64	31.74	86.96
Lora	8K	36.40	12.10	23.98	31.87	86.66
IDEAL$_{Prompt}$	8K	31.41	10.60	22.27	27.19	86.08
IDEAL$_{PAdapter}$	8K	37.27	**13.90**	**26.32**	32.73	87.19
IDEAL$_{LoRA}$	8K	**38.67**	**14.42**	**26.28**	**34.24**	87.29

Table 2: Comparison with baselines on SQuALITY and QMSum.

achieved the best ROUGE-L and BERTScore on SQuALITY, and the best ROUGE-L on QMSum.

3.6 LLM-based Evaluation

The results on reference-based metrics indicate that our method achieves a certain level of effectiveness. However, while these metrics are simple and fast, they suffer from poor correlation with human evaluators, lack interpretability, and fail to capture high-level semantic qualities of summaries. To address this, we employ GPTRank (gpt-4o-2024-08-06) to compare our method with the open-source state-of-the-art (SOTA) approach, Socratic Pret (Pagnoni et al., 2023), for evaluating the high-level semantic qualities of summaries.

As shown in the table 3, while Socratic Pret achieves comparable performance to IDEAL$_{LoRA}$ on ROUGE-L and BERTScore and even surpasses IDEAL$_{LoRA}$ on ROUGE-Lsum, the comparison using GPTRank reveals a different perspective. Across 254 test samples, IDEAL$_{LoRA}$ outperforms Socratic Pret in 214 cases, achieving an 84% win rate. This indicates that although the SOTA non-LLM method performs similarly to LLM-based methods on certain metrics, it lags significantly in terms of high-level semantic qualities. Furthermore, analysis of the GPTRank evaluations reveals that summaries generated by Socratic Pret exhibit more inconsistencies with the original article compared to those generated by IDEAL$_{LoRA}$. Additionally, IDEAL$_{LoRA}$ produces more concise and fluent summaries.

Models	Win	Lose	Tie	R-L	R-Lsum	BScore
SQuALITY Dataset						
Socratic Pret	39	214	1	23.14	**42.28**	85.86
IDEAL$_{LoRA}$	**214**	**39**	1	**25.54**	40.99	**86.86**

Table 3: GPTRank comparison between Socaric Pret. and IDEAL$_{LoRA}$.

Table 4 presents a comprehensive example comparing IDEAL$_{LoRA}$ and Socratic Pret using GPTRank, including the full prompt and GPT's evaluation response, except the full article due to space consideration. We evaluate the quality of the two summaries via prompt engineering with GPT, providing an explanation, a one-word reason for inferior summaries, and an indication of the superior summary (or a tie). To mitigate potential bias from summary ordering, the order of the two summaries was randomized in our experiments.

3.7 Ablation Study

IDEAL$_{LoRA}$ vs LoRA by different training sequence length. To evaluate the effectiveness of our approach under varying training sequence lengths, we compared IDEAL$_{LoRA}$ and LoRA on the SQuALITY dataset across training lengths ranging from 1500 to 8000. Figure 4 illustrates the results in terms of ROUGE-L, ROUGE-Lsum, and BERTScore metrics. The results demonstrate that our method consistently improves performance on the QFS task across different sequence lengths.

Performance vs Parameter Budget. As shown in Figure 1, our Query-aware HyperExperts outperform the corresponding PEFT methods on the QMSum (Golden) dataset while utilizing a comparable amount of trainable parameters. Both LoRA

Prompt:

You will be provided with an article along with a query instruction and two summaries that respond to the query instruction, numbered as follows: 1. Summary 1, 2. Summary 2. Please compare the quality of these two summaries and determine which one is better (a tie is also possible). First, provide an explanation for your decision. Then, specify a one-word reason for why the less favorable summary is inferior. Finally, give your decision in the format: 1, 2, or tie. Response format: **Explanation**: "Your explanation here". **Reason**: "One word reason". **Decision**: 1 or 2 or tie.
Query Instruction: What is Gurn's role in the story?
Article: Produced by Greg Weeks, Mary Meehan and the Online Distributed Proofreading Team at http://www.pgdp.net Raiders of the Second Moon By GENE ELLERMAN A strange destiny had erased Noork's memory,and had brought him to this tiny world to write ...
Summary 1: Gurn is the leader of the wild Vasads of the jungle. He is a renegade from the walled city of Grath, and the city rulers call him a traitor because he has spoken against the making of slaves of the captured Zurans from other valleys. Gurn is also an exile from the city of Konto, and he has told Noork that his brother, Gurn, has told him the reason for the exile. The reason is that the Zurans are no longer making slaves of other valleys, and Gurn believes that they should no longer make them slaves. He tells Noork to go to Gurn and tell him that the Misty Ones can be trapped and skinned.
Summary 2: Gurn is the leader of the Vasads, a group of people who live in the jungles of Sekk. He is a tall, golden-skinned man with a bracelet of golden discs linked together with human hair. He is an exile from the walled city of Grath, and is considered a traitor by the city rulers. He is Noork's friend, and Noork tells Sarna that he has been friends with Gurn. Gurn is also the one who gave Noork his short powerful bow.

GPT Response:

Explanation: Summary 2 provides a more accurate and concise depiction of Gurn's role in the story. It highlights his leadership position among the Vasads, his physical description, his exile status from Grath, his friendship with Noork, and his contribution to Noork by gifting him a bow. On the other hand, Summary 1 contains factual inaccuracies, confuses Gurn's relationship with other characters, and includes irrelevant information, such as exiles and slavery, that is not central to understanding Gurn's role.
Reason: "Inaccuracy". **Decision**: 2.

Table 4: An example comparing summaries generated by IDEAL$_{LoRA}$ and Socratic Pret using GPTRank on SQuALITY dataset. The red-highlighted text indicates significant errors. In this example, Summary 1 is generated by Socratic Pret, and Summary 2 is generated by IDEAL$_{LoRA}$.

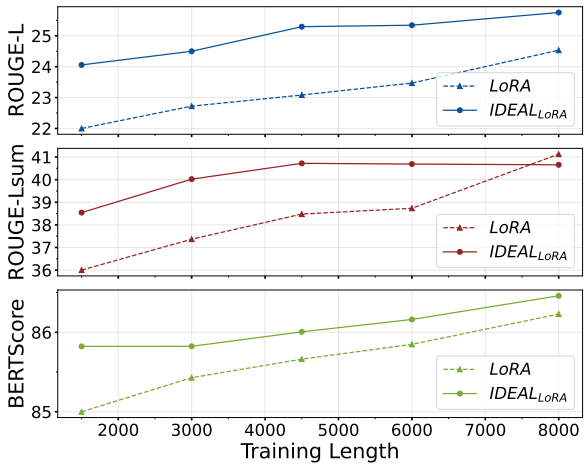


Figure 4: A comparison of LoRA and IDEAL$_{LoRA}$ under different training sequence lengths on SQuALITY dataset.

and PAdapter significantly underperform compared to IDEAL$_{LoRA}$ and IDEAL$_{PAdapter}$, respectively, even with the same or a greater number of trainable parameters, underscoring the effectiveness of HyperExperts.

3.8 Performance of Low Memory IDEAL

IDEAL$_{LoRA}$ consistently demonstrates improved performance as training input length increases. However, this comes at the cost of increased GPU memory consumption. Table 5 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$^{QF_Inf}_{LoRA}$ that integrated with Query-focused Infini-attention exhibits memory efficiency when handling long inputs. IDEAL$^{QF_Inf}_{LoRA}$ maintains a consistent memory footprint of 24G regardless of the input length. Notably, on the QMsum dataset, IDEAL$^{QF_Inf}_{LoRA}$ outperforms IDEAL$_{LoRA}$ with an input length of 1.6K on all metrics within the same 24GB memory constraint. Moreover, it surpasses IDEAL$_{LoRA}$ with an input length of 3.8K in 40GB memory on the ROUGE-L metric.

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 summarize. Vig et al. (2022) use an encoder to compute

Models	QMSum Dataset				SQuALITY Dataset			
	LC	R-L	R-Lsum	BScore	LC	R-L	R-Lsum	BScore
Lora	1.6K	19.58	25.25	84.93	1.6K	20.73	34.41	85.31
IDEAL$_{LoRA}$	1.6K	19.71	26.27	85.29	1.6K	21.16	34.73	85.52
	3.8K	21.62	28.46	85.94	3.8K	22.54	37.54	85.83
	8K	26.28	34.24	87.29	8K	25.54	40.99	86.86
LoRA+Inf	0.8/6K	21.13	26.58	86.00	1.6/9K	20.59	34.76	85.02
IDEAL$_{LoRA}$+Inf	0.8/6K	21.76	26.16	86.02	1.6/9K	21.68	34.81	85.28
w/o ReQ	0.8/6K	16.57	20.40	84.37	1.6/9K	17.89	30.62	84.13
IDEAL$^{QF_Inf}_{LoRA}$	0.8/6K	22.16	27.05	86.16	1.6/9K	21.49	34.86	85.54

Table 5: Comparing IDEAL$^{QF_Inf}_{LoRA}$ 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.

the local attention of a segmented document. The resulting encodings are then concatenated into a single embedding sequence and passed to a decoder model to generate the summary. Pagnoni et al. (2023) introduce a question-driven, unsupervised pre-training objective, specifically designed to improve controllability in summarization tasks. Sotudeh and Goharian (2023) propose a contrastive learning method aimed at improving the relevance of summaries to a given query. 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. LED (Beltagy et al., 2020) employs a more efficient self-attention pattern that allows the model to scale to long documents. 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 the Infini-transformer, we explore the compressive memory method on adapter-based PEFT of LLMs and design a query-focused Infini-attention for QFS tasks.

Hypernetwork-based Methods. Ivison and Peters (2022) investigate input-conditioned hypernetworks for multi-tasking in NLP, which generate parameter-efficient adaptations for a decoder using a hypernetwork conditioned on the output of an encoder. He et al. (2022b) incorporate hypernetworks into prompt-based, task-conditioning Transformer models in a multi-task setting, allowing the network to learn task-specific feature maps. Zhang et al. (2024) employ hypernetworks to generate adaptive parameter shifts for a visual projector and an LLM in multimodal 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. Experimental results demonstrate that our method improves performance on reference-based metrics. Furthermore, in pairwise comparisons against the SOTA fine-tuning method, Socratic Pret (Pagnoni et al., 2023), using the LLM-based evaluator GPTRank, our method achieved a win probability of 0.84, demonstrating its effectiveness.

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, and 1 others. 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. 2022a. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. 2022b. HyperPrompt: Prompt-based Task-Conditioning of Transformers. *arXiv preprint*. ArXiv:2203.00759 [cs].

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 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].

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

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522.

Yixin Liu, Kejian Shi, Katherine He, Longtian Ye, Alexander Richard Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2024. On learning to summarize with large language models as references. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8639–8656.

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, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Artidoro Pagnoni, Alex Fabbri, Wojciech Kryściński, and Chien-Sheng Wu. 2023. Socratic pretraining: Question-driven pretraining for controllable summarization. In *Proceedings of the 61st Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12737–12755.

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

Sajad Sotudeh and Nazli Goharian. 2023. Qontsum: On contrasting salient content for query-focused summarization. *ArXiv*, abs/2307.07586.

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.

Jesse Vig, Alexander Richard Fabbri, Wojciech Kryściński, Chien-Sheng Wu, and Wenhao Liu. 2022. Exploring neural models for query-focused summarization. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1455–1468.

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.

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, and 1 others. 2024. Hyperllava: Dynamic visual and language expert tuning for multimodal large language models. *arXiv preprint arXiv:2403.13447*.

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 The detailed Query-focused Infini-attention.

Memory compression. 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:

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

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

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

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 $\boldsymbol{M}_{s-1}^{query}$ differs from the full context memory only in the value states $\hat{\boldsymbol{V}}_{cache}$ used within the associative binding operator. We utilize 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

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

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

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 $\boldsymbol{M}_{s-1}^{all}$ and the query-focused memory $\boldsymbol{M}_{s-1}^{query}$, respectively. This retrieval is performed using the query states $\boldsymbol{Q} \in \mathbb{R}^{L \times d_{key}}$ as follows:

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

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

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 $\boldsymbol{\beta}$:

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

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

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

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

Repeated query instruction. To incorporate query instructions into the model, we concatenate the query instruction with the document as the model input. 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 generate query-relevant summaries accurately.

B Additional Experiments and Analyses

B.1 Dataset statistics

Table 6 shows the detailed statistics of the datasets used in our experiments. QMsum is a multi-domain dataset for meeting summarization, covering Product, Academic, and Committee meetings. QMsum(Golden) is a shorter version of QMsum where documents only contain sections relevant to the queries. SQuALITY is a public-domain dataset for story summarization. Unlike 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.

B.2 Implementation Details

All IDEAL models are trained by the AdamW optimizer with a cosine annealing schedule after the warmup starts. The warmup epochs, batch size, learning rate, and weight decay are set to 1, 32, 0.006, and 0.02, respectively. We use the validation set to find the optimal epochs for each dataset. 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.

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

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

Table 6: Statistics of query/aspect-based summarization datasets. **#Instances** represents the total number of (document, summary) pairs in the corresponding dataset. **#Input Tk** and **#Output Tk** denote the number of input and output token lengths under the LLaMA2 tokenizer, respectively. **#Queries** indicate the number of unique queries or aspects appearing in the dataset respectively. 2670* represents the number of input tokens for QMsum(Golden).

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.

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 Flash-Attention2 (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.

For the BART model baselines, we use the HuggingFace Transformers library and the AdamW optimizer. We set the batch size, learning rate, and weight decay to 32, 0.0001, and 0.1 respectively, and used the validation set to find the optimal epochs for each dataset.

B.3 Ablation Study

The layers to generate parameters. Table 7 presents the results of IDEAL$_{LoRA}$ on the QMsum (Golden) dataset when generating LoRA parameters for different numbers of layers using a hypernetwork. The results indicate that generating parameters for the last 16 layers achieves the best performance.

Layers	R-1	R-2	R-L	R-Lsum	BScore	Params(M)
8-32	40.04	16.56	29.06	35.71	87.77	24.55
16-32	**40.85**	**16.89**	**29.67**	**36.66**	**87.83**	23.76
24-32	40.73	16.36	28.96	36.22	87.76	22.98
-	40.69	16.11	28.84	36.18	87.71	23.07

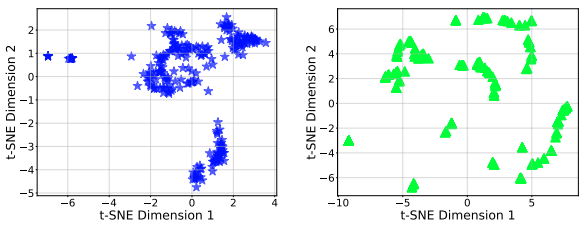
Table 7: Different number of layers that the LoRA parameters are generated of IDEAL$_{LoRA}$ on **QMsum(Golden)** dataset. **16-32** indicates that the LoRA parameters from layers 16 to 32 are generated by the Hypernetwork. **-** indicates no generated parameters.

Different configuration of HyperExpert. Table 8 shows the experimental results for four adapter parameter generation configurations: parallel versus sequential generation, and using different or shared encoders within HyperExpert. Parallel generation with different encoders yields the best performance.

Gen	Enc	R-1	R-2	R-L	R-Lsum	BScore	Params(M)
Para	Diff	**40.85**	**16.89**	**29.67**	**36.66**	**87.83**	23.76
Seq	Diff	40.49	16.56	28.95	35.99	87.70	23.76
Seq	Share	40.57	16.40	29.01	36.25	87.76	19.83
Para	Share	40.43	16.62	28.82	36.18	87.76	19.83

Table 8: Four configurations for generating adapter parameters on **QMsum (Golden)** are evaluated: **Para** denotes parallel generation, while **Seq** refers to sequential generation. **Diff** indicates that the encoder in HyperExpert corresponds to the number of transformer layers generating parameters, whereas **Share** represents the use of a single shared encoder layer.

The diversity of generated parameters. To intuitively illustrate the diversity of parameters generated by the IDEAL model, we applied the t-SNE algorithm to visualize the adapter parameters of a selected layer in two dimensions on the QMSum and SQuALITY test sets. As shown in Figure 5, the generated parameters exhibit distinct distributions across the two datasets, with clearly identifiable clusters. This demonstrates that IDEAL can dynamically generate adapter parameters conditioned on the query.



(a) QMSum. (b) SQuALITY.

Figure 5: t-SNE Visualization of Query-based Parameters' Dynamic Characterizations.

The input representation of HyperExpert. Our HyperExpert takes the representation of the query instruction as input. For comparison, we conducted experiments using the representations of the document and the combined query and document as inputs. The results in table 9 show that using only the query representation as input achieves the best performance, while using the combined query and document representation yields the second-best results.

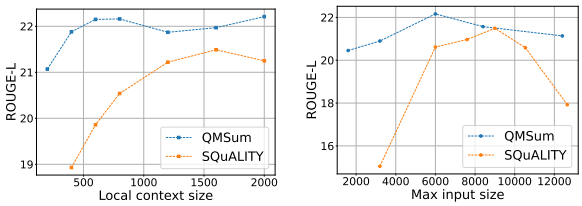
Input	R-1	R-2	R-L	R-Lsum	BScore
Query	**40.85**	**16.89**	**29.67**	**36.66**	**87.83**
Document	40.17	16.10	28.84	35.84	87.70
Query&Document	40.23	16.47	29.14	35.84	87.60

Table 9: Different input representation of IDEAL$_{LoRA}$ on **QMsum(Golden)**.

The effectiveness of each module in IDEAL$^{QF_Inf}_{LoRA}$. In Table 5, 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 the 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$^{QF_Inf}_{LoRA}$ 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.

Local context size of IDEAL$^{QF_Inf}_{LoRA}$. Figure 6a presents the performance of IDEAL$^{QF_Inf}_{LoRA}$ under varying local context sizes (LC). On the QMSum dataset, the model exhibits stable performance when LC exceeds 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 IDEAL$^{QF_Inf}_{LoRA}$ differs from IDEAL$_{LoRA}$, the limited memory for the former is enough to handle extremely long inputs.

Max input length of IDEAL$^{QF_Inf}_{LoRA}$. Figure 6b presents the optimal max input length for IDEAL$^{QF_Inf}_{LoRA}$ on the QMsum and SQuALITY datasets.



(a) Performance with respect to the different local context size of IDEAL$^{QF_Inf}_{LoRA}$.

(b) Performance with respect to the different max input length of IDEAL$^{QF_Inf}_{LoRA}$.

Figure 6: Local and max input length of IDEAL$^{QF_Inf}_{LoRA}$.

B.4 The performance under LLaMA2 and LLaMA3 backbones.

Table 10 shows the comparison of IDEAL$_{LoRA}$ under LLaMA2-7B and LLaMA3-8B, LLaMA3.1-8B backbones with the same 24GB GPU memory. Due to the higher memory consumption of IDEAL$_{LoRA}$ with the LLaMA 3 series backbone, resulting in a smaller local context size, evaluations show that it performs better with the LLaMA2-7B backbone given the same GPU memory. Therefore, in the low-memory experiments presented in Table 5, LLaMA2-7B was used for all models except IDEAL$_{LoRA}$ with an 8K context length, which used LLaMA 3.1-8B.

B.5 Training Time Comparison

Table 11 shows the comparison between our methods and baselines. IDEAL$_{LoRA}$ doesn't increase training time compared to LoRA. IDEAL$^{QF_Inf}_{LoRA}$ slightly increases training time compared to IDEAL$_{LoRA}$+Inf and LoRA+Inf.

B.6 Human Evaluation

To ensure a fair evaluation, we conducted a human evaluation of the summaries generated by BART-Large, LoRA (LLaMA2-7B), and IDEAL$_{LoRA}$ (LLaMA2-7B) on the QMSum (Golden) dataset, all using identical computational resources. Each model was tested using a 3090 (24GB) GPU. We recruited three well-educated evaluators and randomly selected 50 samples to evaluate the summary

Models	Backbone	**LC**	**R-1**	**R-2**	**R-L**	**R-Lsum**	**BScore**	**Params(M)**
IDEAL$_{LoRA}$	LLaMA2-7B	1.6K	**40.82**	**16.61**	**29.00**	**36.08**	**87.68**	24.5
IDEAL$_{LoRA}$	LLaMA3-8B	1K	39.99	15.63	27.89	35.34	87.54	23.8
IDEAL$_{LoRA}$	LLaMA3.1-8B	1K	39.90	16.06	28.55	35.57	87.48	23.8

Table 10: The comparison under LLaMA2 and LLaMA3, LLaMA3.1 backbones on **QMsum(Golden)** dataset with 24GB GPU memory.

Models	**LC**	**Time/Epoch**
Lora	1.6K	11min
IDEAL$_{LoRA}$	1.6K	11min
LoRA+Inf	0.8/6K	45min
IDEAL$_{LoRA}$+Inf	0.8/6K	46min
IDEAL$_{LoRA}^{QF_Inf}$	0.8/6K	50min

Table 11: Training time per epoch with 2 Nvidia GeForce RTX 3090 GPUs in data parallel mode on the QMSum dataset.

quality of the three models from two aspects: Correctness and Coverage (Wang et al., 2022). For each sample, the evaluators read the document and the corresponding question, then selected the best and worst summary among the three. In each case, we randomized the order of the summaries from the three models. The task instruction are detailed in Figure 7.

As shown in table 12, although the BART-Large model did not lag far behind the other two models in terms of the ROUGE-L metric, it received significantly fewer "Best" summary votes and the most "Worst" votes. This may be because the LLaMA-based methods benefit from the understanding and reasoning capabilities of LLMs. IDEAL$_{LoRA}$ received 13 more "Best" votes compared to LoRA and had the fewest "Worst" votes, far less than LoRA's 46 and BART-Large's 75. This demonstrates that our proposed Query-focused PEFT method is indeed effective for QFS tasks.

Models	LC	Best	Worst	Correctness	Coverage	Rouge-L
Bart-Large	1K	33	75	3.3	2.9	25.25
LoRA(LLaMA2-7B)	1.6K	52	46	3.6	3.4	27.36
IDEAL$_{LoRA}$(LLaMA2-7B)	1.6K	65	29	3.7	3.6	29.00

Table 12: The human evaluation results of model-generated summaries obtained using the same 24G memory GPU on the QMSum (Golden) dataset. We summed the votes of the three evaluators for the Best and Worst metrics. For Correctness and Coverage, we first calculated the average for all samples, and then computed the mean across the three evaluators.

Task Instructions

You are required to complete the evaluation of 50 documents within two days. Each document is accompanied by a question and three different summary responses. Your task is to score each of the three summaries based on **Correctness** and **Coverage** using a scale from 1 to 5. After scoring, you need to select the best and worst summary for each document based on your evaluations of these two criteria.

To ensure the quality of your assessments, please take breaks during the process.

Definitions:

- **Correctness**: Evaluates the presence of factual errors in the summary. A factual error is defined as a statement that contradicts the document, is not directly stated, or is not heavily implied or logically entailed by the document.
- **Coverage**: Measures how comprehensively the summary includes all relevant information and details from the document that are necessary to answer the question.

Example Evaluation: For Document 15, the scores might look like this:

- A: 3 (Correctness), 4 (Coverage)
- B: 5 (Correctness), 3 (Coverage)
- C: 2 (Correctness), 4 (Coverage)
- **Best**: B
- **Worst**: C

Format your results in one line per document as shown in the example:

```
Doc 15, A:3 4, B:5 3, C:2 4, Best:B, Worst:C
```

Figure 7: Task instructions of Human evaluation.