Leveraging LoRA to Compare Domain-Specific News Summary Generation: Domain Model vs. Larger General Model

Shipeng Liu
732A81- Text Mining
shili506@student.liu.se

Abstract

The burgeoning field of abstractive summarization employing large language models holds great promise, yet the associated challenges of time and resource-intensive in training for these models are substantial. To address the issue of the difficulty in training large models, various Parameter-Efficient Fine-Tuning (PEFT) technologies, including Low-rank adaptation (LoRA), have been introduced. This research encompasses two distinct tasks. Task A investigates the efficacy of fine-tuning the pre-trained model flan-t5-small using LoRA for enhancing its performance in generating news summaries. Task B explores the use of LoRA to ascertain whether larger general model outperform specialized-domain model in specific domains' task. This study proposes a research method inspired by the problem posed in a previous paper. The outcomes of Task A reveal significant improvements in news summary generation when employing LoRA for fine-tuning the pre-trained model. In Task B, it is observed that larger general model exhibits superior performance in specific domains compared to specialized-domain model. Evaluation metrics such as ROUGE and BERTScore are employed in both tasks, considering both text overlap and semantic similarity to provide a comprehensive evaluation of the models' performance.

1 Introduction

Branco et al. mentioned that in 2008, an average American received approximately 100,500 words of information daily from the media, equivalent to the brain encountering 23 words per second. The authors argue that such a vast amount of information places stress on the human brain, leading to a shortened attention span. In an effort to save human effort, time costs, and enhance work efficiency while alleviating the pressure brought about by information overload, exploring the performance of language generation models in generating news and email summaries holds great promise.

Sequence-to-Sequence (Seq2Seq) Architectures (Sutskever et al.) are often employed for summary generation. The subsequent introduction of Transformer (Vaswani et al.) significantly improved the performance of this framework. Today, large pre-trained language models trained in deep learning research, such as GPT-3 (Brown et al.) and BERT (Devlin et al.), exhibit outstanding performance in natural language processing tasks such as text classification and text generation. In Pu et al., a qualitative and quantitative comparison is made between summaries generated by Large Language Models (LLMs) and those authored by humans. The results indicate a preference among human evaluators for text summaries generated by LLMs, demonstrating higher authenticity compared to human-written summaries. The article also notes significant advancements in text summarization generated by LLMs compared to previous methods.

However, the training cost of large pre-trained models is prohibitively high. GPT-3 has 175 billion parameters, while the BERT model has 110 million parameters. To address the challenges of training, some Parameter-Efficient Fine-Tuning (PEFT) techniques have been proposed, such as LoRA (Hu et al.), Prefix Tuning (Li and Liang), P-Tuning (Liu et al., a), and Prompt Tuning (Lester et al.). PEFT techniques reduce the difficulty of training large language models by minimizing the number of parameters that need to be trained, along with computational complexity, significantly shortening the training time and computational costs. PEFT allows us to efficiently adapt to new tasks, leveraging the knowledge of pre-trained models even when computational resources are limited, enabling effective transfer learning. In this paper, we employ LoRA to fine-tune a news summary generation model on the pre-trained language model Flan-T5-small (Chung et al.).

This paper presents our work on using LoRA to adjust pre-trained models on news summary data, divided into Task A and Task B. In Task A, we utilize LoRA for fine-tuning on flan-t5-small and explore the performance improvement of the new model in news summary generation tasks. Task B represents a potential research approach based on the results proposed in the paper Suri et al.. The results in that paper indicate that after fine-tuning with LoRA, larger models perform better in professional domains compared to models specifically trained for those domains. To investigate this phenomenon, we train models A and B using LoRA, both based on Flan-T5-small. Model A has a higher lora rank, meaning it has more parameters in the LoRA layer, and we use text from all types of news to train Model A. Model B has a smaller lora rank, and we only use text from a specific domain of news to train Model B. Through the design of Task B, we offer an experimental approach to explore the observations in the mentioned paper and provide a conclusion for another question—whether training models with news from other categories can enhance the model's performance when training data for a specific category is limited.

The structure of this paper is as follows. In Section 2, we briefly introduce the technologies used in this paper. In Section 3, we present the experimental objectives and data description for Task A and Task B. In Section 4, we detail the experimental design and evaluation methods for Task A and B for reproducibility. Next, in Section 5, we summarize the experimental results and draw preliminary conclusions. Finally, we discuss the limitations of the experimental design and some shortcomings in the current evaluation methods for summary generation models.

2 Related Literature

2.1 Deep Learning in Natural Language Processing

Natural Language Processing (NLP), as a technology increasingly crucial in the era of data explosion, encompasses the understanding, processing, and generation of language. NLP tasks include but are not limited to text classification, machine translation, sentiment analysis, and summarization generation, all of which require computers to comprehend, process, and generate natural language. However, traditional NLP algorithms have gradually revealed their limitations when faced with complex semantics and context-sensitive tasks. The introduction of deep learning in the field of NLP has demonstrated superior performance in many tasks.

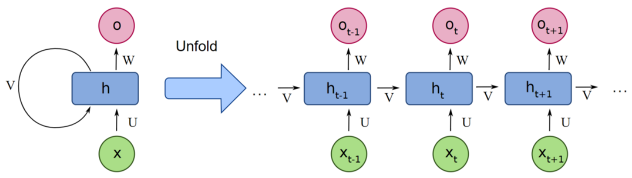

Figure 1: RNN Architecture

Natural language often exists in sequential form, and in the process of sequence modeling, a series of models such as RNN, GRU, and LSTM can be applied. Due to its recurrent structure, RNN can effectively handle text as sequential data and exhibit good performance in tasks that require sensitivity to context. The architecture of RNN is typically depicted as shown in Figure 1. From Jordan, there are

$$h_t = \sigma_h(W_h x_t + U_h o_{t-1} + b_h)$$

$$o_t = \sigma_o(W_o h_t + b_o)$$

Where the σ_h and σ_o are both activation function. The basic structure of an RNN includes the input layer, hidden layer, and output layer. At each time step, the model receives an input and the hidden state from the previous time step, generating a new hidden state through an activation function. This design enables RNN to capture information from earlier sequences and perform well in context-sensitive tasks. Additionally, there is a type of bidirectional neural network structure called Bidirectional Recurrent Neural Networks (BRNN) (Schuster and Paliwal), which connects two oppositely oriented hidden layers to the same output. Consequently, the output layer can simultaneously acquire information from both past and future states, making BRNN particularly effective in tasks such as machine translation, where context dependency needs to be considered. Figure 2 illustrates the general architecture of BRNN.

Despite the commendable performance of RNN in handling context-dependent tasks, it suffers from the issue of long-term dependencies. When dealing with sequences spanning a long time span, RNN struggles to capture information that is far from the current time step. This challenge arises due to the way hidden layer states are propagated through the multiplication of weight matrices at each time

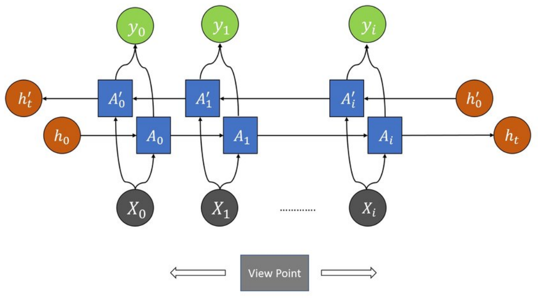

Figure 2: BRNN Architecture(Chadha et al.)

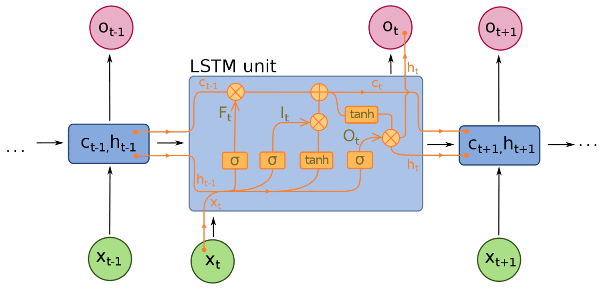

Figure 3: LSTM Unit

step during the backpropagation process. Consequently, we must multiply the gradients of each weight matrix to update each weight. If the gradient is less than 1, the multiplication causes the gradient to gradually approach zero, leading to minimal changes in parameter updates. Conversely, if the gradient is greater than 1, the multiplication results in an exponential growth of the gradient, causing overly drastic updates in parameters and rendering the model unstable.

To address the long-term dependency problem of RNN, a series of improved models have been proposed. These include Gated Recurrent Units (GRUs) (Cho et al.), and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber).

LSTM is a neural network with the ability to memorize long and short-term information. It was introduced to address the issue of long-term dependencies. LSTM builds upon the foundation of RNN by incorporating gate mechanisms to control the flow and retention of information. Figure 3 illustrates the model structure of LSTM.

In the LSTM unit,

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{C}_t = tanh(W_C x_t + U_C h_{t-1} + b_C)$$

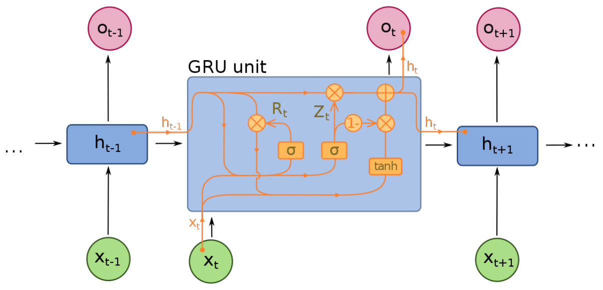

Figure 4: GRU Unit

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot tanh(C_t)$$

where the operand $\odot$ denotes element-wise product, $\sigma(\cdot)$ denotes the Sigmoid activation function. In the model, C_t is referred to as the cell state, f_t is the forget gate, indicating which features of C_{t-1} are used to compute C_t. f_t is a vector with each element within the range [0,1]. $\tilde{C}_t$ represents the update value of the cell state, obtained through a neural network layer applied to the input data x_t and the hidden state h_{t-1}. The activation function for the update value of the cell state typically utilizes the $tanh$ function. i_t is known as the input gate, also a vector with elements in the range [0,1], calculated using the Sigmoid activation function applied to x_t and h_{t-1}. i_t is responsible for controlling which features of $\tilde{C}_t$ are used to update C_t.

Finally, to compute the predicted value and the input for the next time step, LSTM needs to calculate the output of the hidden node h_t. It is obtained through the output gate o_t and the cell state C_t.

GRU (Gated Recurrent Unit) is a simplified version of LSTM. Compared to LSTM, GRU achieves comparable results and is easier to train. GRU abandons the forget gate, input gate, and output gate present in LSTM, replacing them with reset gate and update gate mechanisms. The general architecture of GRU is shown in Figure 4, where

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$\hat{h_t} = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) \odot + z_t \odot \hat{h_t}$$

where the operand $\odot$ denotes element-wise product, $\sigma(\cdot)$ denotes the Sigmoid activation function. Given the current input x_t and the hidden state h_{t-1} passed down from the previous node, which

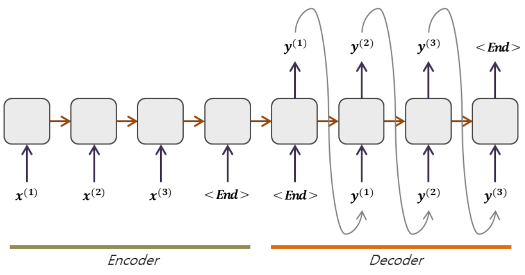

Figure 5: Seq2Seq structure

contains relevant information from previous nodes, GRU utilizes them to obtain two gate control states, r_t and z_t. Here r_t controls the reset gate, and z_t controls the update gate. The Sigmoid activation function is employed, mapping data to the [0,1] range, serving as gate control signals. Next, the data after "reset," denoted as $\hat{h_{t-1}}$, is concatenated with x_t, and then through the $tanh$ activation function, the data is mapped to the [-1,1] range to obtain $\hat{h_t}$. $\hat{h_t}$ contains information from the current input x_t. The step of adding $\hat{h_t}$ to the hidden state of the current time step is equivalent to memorizing the state at the current time. Finally, using the previously obtained update gate z_t to update the hidden state h_t. It is worth noting that when z_t is close to 1, it indicates retaining more information, and conversely, when close to 0, it signifies forgetting more information.

2.2 Seq2Seq Model

When dealing with tasks with an uncertain output length, such as machine translation, it is suitable to apply a network with the Seq2Seq architecture. The network structure is shown as Figure 5.

The Seq2Seq architecture primarily consists of two components: Encoder and Decoder. The Encoder compresses the input sequence into a vector of specified length, considering this vector as the semantics of the sequence; this process is referred to as encoding. The Decoder, on the other hand, generates a sequence based on the semantic vector; this process is known as decoding.

When using the decoder for decoding, it is necessary to use the output decoded at the previous time step as the input for the next decoding step. This can potentially lead to the gradual propagation of generated errors. To address this issue, Beam Search can be employed, as illustrated in Figure 6.

At each time step, the decoder selects the top k predicted results as the input for the next decoding step. These k results are sequentially fed into

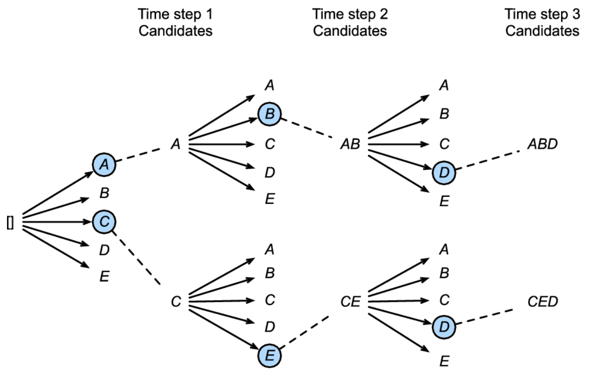

Figure 6: Beam Search

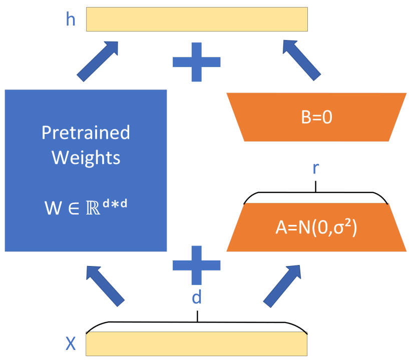

Figure 7: Reparametrization in LoRA

the decoder for decoding, resulting in k predicted outcomes. From all the decoding results, the top k predicted results are then chosen as the input for the next decoder step, ultimately selecting the output with the highest probability.

2.3 LoRA

Fine-tuning language models to adapt to specific domains and tasks is a crucial aspect of natural language processing (NLP). However, as the scale of models increases, the feasibility of fine-tuning all parameters becomes progressively lower. For instance, with GPT-3 having a parameter count of 175 billion, it implies that for every adaptation to a new domain, full fine-tuning a new model becomes a difficult task.

Aghajanyan et al. asserts that pre-trained models possess an extremely small intrinsic dimensionality, indicating the existence of a low-dimensional parameter space where fine-tuning is as effective as fine-tuning in the full parameter space. Additionally, the research reveals that, post pre-training, larger models exhibit even smaller intrinsic dimensions.

Motivated by the insights from the aforementioned paper, researchers posit the existence of an intrinsic rank during the parameter updating process. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, the parameter update $\triangle W$ can be expressed using a low-rank decomposition:

$$W_0 + \triangle W = W_0 + BA$$

where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ and $r << min(d, k)$.

During the training process, the parameters W_0 are frozen, and only the parameters in A and B are subject to training. As illustrated in 7, the forward propagation process for $h = W_0 x$ is transformed into:

$$h = W_0 x + \triangle W_x = W_0 x + BAx$$

Extensive comparative experiments have substantiated the effectiveness of LoRA. This methodology enables the training of large language models even on less efficient computing systems.

2.4 Evaluate Methods for Text Summarization

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics designed for the automatic evaluation of text summarization quality. Its purpose is to measure the similarity between generated summaries and reference summaries, focusing on recall rather than precision. In other words, it examines how many n-grams from reference summaries appear in the output.

ROUGE-N is the most basic ROUGE metric, where N represents the size of the $N - grams$. ROUGE-N includes ROUGE-1, ROUGE-2, and so on. ROUGE-N primarily measures recall on N-grams. For N-grams, the ROUGE-N score can be computed using the following formula:

$$\frac{\sum_{S \in \{RefrenceSummaries\}} \sum_{gram_N \in S} Count_{match}(gram_N)}{\sum_{S \in \{RefrenceSummaries\}} \sum_{gram_N \in S} Count(gram_N)}$$

Here, the denominator represents the number of $N - grams$ in the reference summary, and the numerator represents the number of common $N - grams$ between the reference and generated summaries.

In ROUGE-L, the "L" stands for the Longest Common Subsequence (LCS). When computing ROUGE-L, it utilizes the Longest Common Subsequence between the machine-generated summary C and the reference summary S. The calculation is formulated as follows:

$$R_{LCS} = \frac{LCS(C, S)}{len(S)}$$

$$P_{LCS} = \frac{LCS(C, S)}{len(C)}$$

$$ROUGE - L = F_{LCS} = \frac{(1 + \beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}}$$

Here, R_{LCS} denotes Recall, P_{LCS} represents Precision, F_{LCS} signifies the ROUGE-L score. As β increases, ROUGE-L places greater emphasis on recall rather than precision.

BERTScore is commonly employed for assessing the quality of generated text. It leverages a pre-trained BERT model to calculate the similarity between the generated text and the reference text. When evaluating machine-generated summaries, BERTScore compares the embeddings of the generated summary with those of the reference summary. It computes the cosine similarity between them and utilizes this information to calculate Recall, Precision, and F1 Score.

3 Task Description

This study delves into the utilization of LoRA to enhance the efficacy of pre-trained models in the domain of news summary generation tasks. The investigation is bifurcated into two distinctive subtasks: Task A aims to meticulously fine-tune a news summary generation model, aligning its output summaries with human-generated counterparts. Meanwhile, Task B focuses on the training of two summary generation models. Model A, an expansive variant catering to various news categories, is juxtaposed with Model B, a more compact version tailored exclusively for business-related news. In the model design, Model A boasts a higher LoRA rank, indicating an increased parameter count, whereas Model B employs a smaller LoRA rank with fewer parameters. The differences between Models A and B are detailed in table 1. Subsequently, we scrutinize and compare the summary generation performance of these models specifically in the context of business news.

The evaluation methodologies applied to both tasks encompass ROUGE-1, ROUGE-2, ROUGE-L (Lin), and BERTScore F1 (Zhang et al., a). The evaluation framework aims to encompass two critical dimensions: text coincidence and semantic similarity in the generated results. ROUGE-1, ROUGE-2, and ROUGE-L emphasize text overlap,

Model	**Base Model**	**LoRA rank**	**LoRA layer parameters**	**All parameter counts**
Model A	Flan-T5-Small	32	3538944	251116800
Model B	Flan-T5-Small	8	884736	248462592

Table 1: Configuration of model A and model B

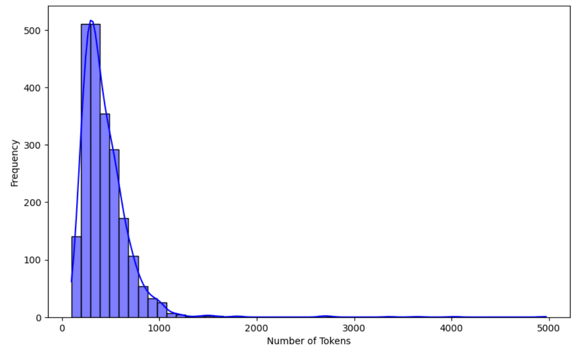

Figure 8: Distribution of Token Counts in News

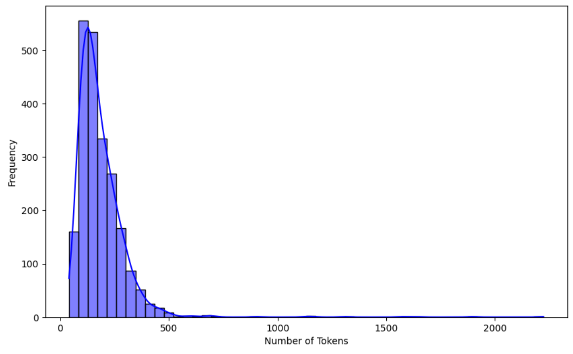

Figure 9: Distribution of Token Counts in Summaries

providing insights into textual congruence, while BERTScore places a greater emphasis on semantic similarity, offering a nuanced perspective on the models' capacity to generate summaries that capture the underlying meaning. This multifaceted evaluation approach ensures a comprehensive assessment of the models' performance.

4 Data

The data set contains 2225 news articles covering the fields of Sport, Business, Politics, Technology and Entertainment. Each article has its corresponding category and human-generated summary. All news comes from the BBC News website. The tokens distribution of news and abstracts can be found in Figure 8 and 9. In Task A, we use all data to fine-tune the model; in Task B, for model A, we use all categories of news in the training set and validation set, and only use business category news in the test set; for model B, We only use business category news as the training set, validation set, and test set.

5 Methods

5.1 Task A Methodology

The goal of this Task is to explore whether using LoRA can pre-train the model to obtain better news summary performance, and make the news summary generated by the fine-tuned model as fluent as possible and close to the summary generated by humans. The workflow of the task is shown in Figure 10.

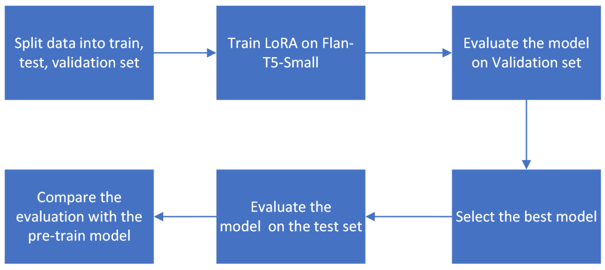
Split data into train, test, validation set
Train LoRA on Flan-T5-Small
Evaluate the model on Validation set
Select the best model
Evaluate the model on the test set
Compare the evaluation with the pre-train model

Figure 10: Task A Workflow

We use Flan-T5-Small as the backbone and initialize a LoRA layer on it. The input to the model is a complete news article, and the output is a summary of the article. During the preprocessing process, special tokens need to be used to mark the beginning, end and separation of each sentence of the text, and tokenize them into vector form. For the encoder, we use 700 tokens as the length, which can ensure that most of the news in the data set is complete when input to the model; at the same time, the length of the decoder is set as 300 tokens, which can ensure that the length of the summary generated by the model is consistent with human-generated summaries in the dataset. The data is divided into training set, validation set and test set according to the ratio of 0.9:0.05:0.05. When training the model, we use early-stopping (Yao et al.) based on Validation Negative Log Loss. The selection ranges of some hyperparameters as shown in Table 2.

There are many choices for the decoding strategy of the Seq2Seq model, such as Beam Search

Variable	Data Type	Range
lora rank	Integer	8-32
lora alpha	Integer	32
learning rate	Float	1e-3
Epochs	Integer	early stopping
batch size	Integer	auto[1]
lora dropout	Float	0.01-0.05

Table 2: Search space for LoRA training

[1] Find a batch size that will fit into memory automatically.

Variable	Value
Max New Token	200
Decoding Strategy	Beam Search
Beams Number	1
Early Stopping	False
Length Penalty	1.0

Table 3: Parameters of the docoder

(Graves), Top-k Sampling (Fan et al.), etc. In this Task, we use Beam Search and set the beam number to 1. Other parameters of the decoder are shown in Table 3.

The methods to evaluate the model are ROUGE-1, ROUGE-2, ROUGE-L and BERTScore F1. We calculate the arithmetic averages of ROUGE-1, ROUGE-2, and ROUGE-L, and then take the arithmetic average of the results along with BERTScore F1 to obtain the final score.

5.2 Task B Methodology

The goal of this Task is to use LoRA to train a model A and a model B. They are both based on Flan-T5-small, but the LoRA layer of model A has more parameters, and we use all types of news texts to train model A; The LoRA layer of model B has fewer parameters, and we only use news in specific fields to train model B. Finally, we hope to compare the performance of the two models in generating business-type news summaries. Therefore, in the data set, 10% of the news in the business category was sampled as a common test set for models A and B, and the remaining data was copied into two identical copies for training models A and B.The workflow of the task is shown in Figure 11.

When training model B, from the remaining data, we extract all news articles categorized as business, and partition the data into a training set and a validation set in a 0.95:0.05 ratio. For the training of

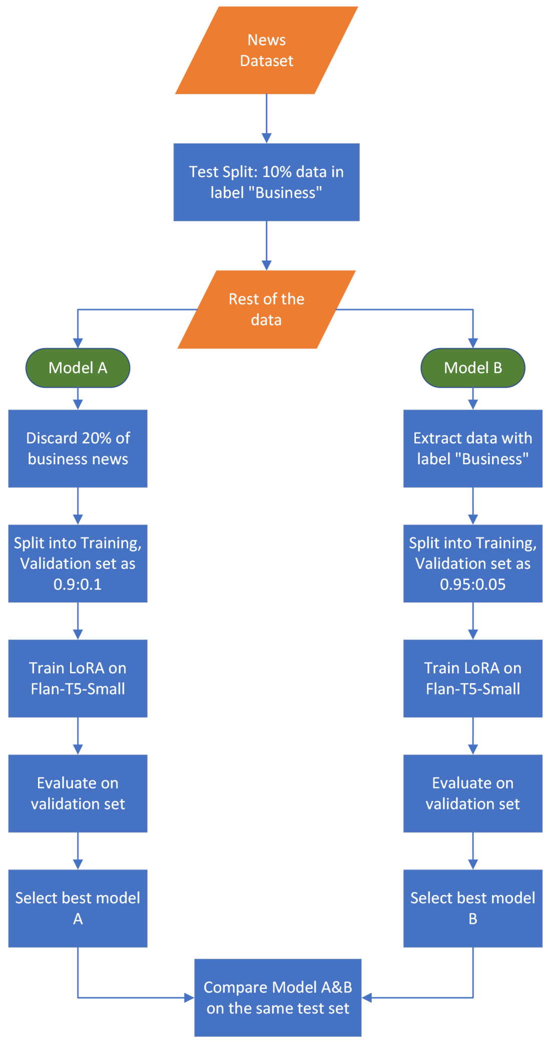
News Dataset
Test Split: 10% data in label "Business"
Rest of the data
Model A
Model B
Discard 20% of business news
Extract data with label "Business"
Split into Training, Validation set as 0.9:0.1
Split into Training, Validation set as 0.95:0.05
Train LoRA on Flan-T5-Small
Train LoRA on Flan-T5-Small
Evaluate on validation set
Evaluate on validation set
Select best model A
Select best model B
Compare Model A&B on the same test set

Figure 11: Task B Workflow

Score	Pre-train Model	LoRA Model
ROUGE-1	0.23886	0.69154
ROUGE-2	0.18065	0.60281
ROUGE-L	0.20498	0.52841
BERT F1	0.89218	0.94224
Final Score	0.55017	0.77491

Table 4: Task A evaluation results

Score	Model A	Model B
ROUGE-1	0.71634	0.70251
ROUGE-2	0.62945	0.60775
ROUGE-L	0.53240	0.51387
BERT F1	0.92463	0.90055
Final Score	0.77535	0.75430

Table 5: Task B evaluation results

model A, in order to reduce the amount of information related to business news compared to model B, we deliberately discard 20% of the data from the business category and partition the remaining data into training and validation sets in a 0.9:0.1 ratio. Throughout the training of models A and B, we employ early-stopping based on Validation Negative Log Loss to ensure optimal performance on the validation set. Finally, we evaluate the performance of both models on a dataset containing only business-category news. The parameters of the encoder and decoder, as well as the model evaluation method in Task B, remain consistent with those employed in Task A.

6 Result

6.1 Task A Result

This study trained a LoRA layer on the pre-trained model flan-t5-small and evaluated summary generation on the test set. The evaluation results are shown in Table 4.

Here we calculate the arithmetic averages of ROUGE-1, ROUGE-2, and ROUGE-L, and then take the arithmetic average of the results along with BERTScore F1 to obtain the final score. The table shows that compared with the pre-trained model, the ROUGE-1, ROUGE-2, and ROUGE-L scores of the LoRA model on the news summary generation task have been significantly improved, and BERT F1 has also been improved to a certain extent. This shows that using LoRA to fine-tune the model can improve the model's performance in news summary generation while consuming less computing resources, time and memory.

6.2 Task B Result

The evaluation of Model A (larger model) and Model B (specialized domain model) on the test set containing only business news is shown in Table 5. Model A and Model B, which performed best on the validation set, have LoRA layer parameters of 3538944 and 884736 respectively. There are a total of 2083 news items in the training set and validation set used to train model A. In contrast, there are only 459 news in the training set and validation set for model B.

It can be seen from the table that model A performs better than model B on the test set. But on the other hand, although model B only uses less news training and the number of LoRA layer parameters is much smaller than model A, the score is still high and there is not much difference with model B. Therefore, it is very efficient to use domain-specific data to train models for specific domain summarization with small LoRA rank.

Furthermore, model A outperforms model B on the test set, and the reason for this improvement may be attributed to the fact that, despite being trained with a smaller subset of business news, model A acquires information during its training on other news categories that is applicable to the generation of business news summaries, such as knowledge about the structure of summaries. Therefore, when faced with limited data for training domain-specific models, augmenting the dataset with text from a broader domain that encompasses the specific domain proves to be a viable option for enhancing the model's performance.

Simultaneously, we make a conjecture: if the amount of business news data used for training Model B is equal to the training data volume of Model A, and the LoRA rank is also equal, then the performance of Model B on the task of business news summary generation is expected to surpass that of Model A. This conjecture requires further research for validation.

7 Conclusion

In conclusion, this study presents a comprehensive exploration of LoRA's application in refining pre-training models for news summary data, delineated into Task A and Task B. In Task A, the efficacy of LoRA fine-tuning in substantially enhancing the model's performance in summary generation is convincingly demonstrated. Transitioning to Task B, the development and assessment of both the larger general model A and the specialized-domain model B on specific domain data reveal that model A outperforms model B, albeit not to a statistically significant extent. The findings from Task B highlight a insight: in scenarios where there is limited data available for training a domain-specific model, the augmentation of data with text from a broader domain encompassing the specific domain proves to be a viable strategy for optimizing model performance.

8 Discussion

In this investigation, our focus was confined to exploring the impact of model size and training data volume on the performance of language models. While acknowledging the significance of these factors, it is crucial to underscore the pivotal role of prompts in enabling Large Language Models (LLMs) to achieve zero-sample summarization proficiency, as highlighted in the Zhang et al., b. The performance of a pre-trained model's output can be substantially enhanced through the careful selection of prompts.

Furthermore, in the context of news summarization, the evaluation criteria extend beyond mere similarity to human-generated summaries. Parameters such as coherence, faithfulness, and relevance are equally critical, with a heightened emphasis on factual consistency. Unfortunately, assessing these attributes in a non-human context remains a challenge. Notably, the Microsoft team has introduced G-Eval, a method based on GPT-4, to evaluate summaries against these multifaceted standards (Liu et al., b). However, it is imperative to acknowledge the limitations imposed by the weakness of reasoning capabilities in large language models, as evident in studies demonstrating their challenge in discerning causation from correlation (Jin et al.). Consequently, the accuracy of assessments regarding factual consistency cannot be guaranteed.

Lastly, in Task B of our study, we employed 368 business news articles for training Model A and 469 for training Model B. It is noteworthy that, in practical scenarios, specialized-domain models often require a substantially larger volume of domain-specific data compared to general-purpose models. This underscores the importance of considering the unique data requirements inherent in training models tailored for specialized domains.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning.

Fernando Branco, Monic Sun, and J Miguel Villas-Boas. Too much information? information provision and search costs. 35(4):605–618. Publisher: INFORMS.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and others. Language models are few-shot learners. 33:1877–1901.

Gavneet Singh Chadha, Ambarish Panambilly, Andreas Schwung, and Steven X Ding. Bidirectional deep recurrent neural networks for process fault classification. 106:330–342. Publisher: Elsevier.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and others. Scaling instruction-finetuned language models.

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

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation.

Alex Graves. Sequence transduction with recurrent neural networks.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9(8):1735–1780. Publisher: MIT press.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models.

Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation?

Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. a. GPT understands, too. Publisher: Elsevier.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. b. G-eval: Nlg evaluation using gpt-4 with better human alignment, may 2023. 6.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (almost) dead.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. 45(11):2673–2681. Publisher: Ieee.

Kunal Suri, Prakhar Mishra, Saumajit Saha, and Atul Singh. Suryakiran at mediqa-sum 2023: Leveraging lora for clinical dialogue summarization.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. 27.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, {\textbackslash}Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. 26:289–315. Publisher: Springer.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. a. Bertscore: Evaluating text generation with bert.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. b. Benchmarking large language models for news summarization.