Advanced Techniques in NLP: Summarisation with In-Context Learning and LoRA (2024)

University of Edinburgh
Pasquale Minervini

Tutorial 6: Implementing Text Summarisation Systems (Week 12)

In our course, we explored the basics and some advanced topics in NLP, including abstractive and extractive text summarisation. In this tutorial, we delve into two cutting-edge methods: in-context learning for summarisation and parameter-efficient fine-tuning with LoRA (Low-Rank Adaptation). Both techniques represent the forefront of making NLP models more flexible and efficient.

Question 1: In-Context Learning for Summarisation

In-Context Learning allows models to perform tasks without explicit retraining or fine-tuning, using a prompt that includes examples of the task to guide the model's predictions. This question focuses on applying in-context learning to summarisation using the CNN/Daily Mail (CNN/DM) or XSum datasets.

Question 1: In this task, you will leverage LLaMA, a decoder-only transformer-based language model, for in-context learning applied to summarisation. Implement a summarisation system using in-context learning with either the CNN/DM or XSum dataset. Outline the steps you would take to format the data, design your prompts, and evaluate the system's performance. Consider the impact of the number and quality of in-context examples on the system's output.

Solution 1:

In-context learning leverages the ability of large language models to generate task-relevant outputs based on the context provided in the prompt. To implement a summarisation system using in-context learning with the CNN/Daily Mail (CNN/DM) or XSum dataset, follow these steps:

Step 1: Data Formatting First, format the articles and summaries from the chosen dataset into a prompt-friendly format. For each example, combine the article and its corresponding summary into a single text block, separating them with a clear delimiter, such as "\n\nSummary:\n".

Step 2: Designing Prompts Create a prompt template that includes a brief instruction to the model to summarise the article, followed by two to three high-quality in-context examples. The prompt should end with the article to be summarised, again followed by the delimiter. This setup teaches the model the desired input-output pattern.

Step 3: Evaluation To evaluate the system, use standard summarisation metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores. These metrics compare the generated summaries against the reference summaries in the dataset to assess the quality of the summarisation.

Impact of In-Context Examples The number and quality of in-context examples significantly impact the system's performance. Too few examples might not provide enough guidance, while too many could overwhelm the model's context window. High-quality examples that are representative of the dataset ensure the model learns the appropriate summarisation style and content.

A working example of how to use in-context learning for summarisation using open-source LLMs is available at this link — please discuss it in class if time allows!

Question 2: Parameter-Efficient Fine-Tuning with LoRA

Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning technique that adapts pre-trained models to specific tasks without modifying the original model weights. Instead, LoRA introduces trainable parameters that interact with the pre-trained weights through low-rank matrices. 5 This approach significantly reduces the number of parameters needed to fine-tune large models on tasks like summarisation.

Question 2: Using the CNN/DM or XSum dataset, implement a summarization system by applying LoRA for fine-tuning a pre-trained decoder-only transformer-based language model, such as LLaMA. Outline the steps to integrate LoRA into the LLM architecture, focusing on the specific adaptations needed for a decoder-only transformer-based model. Describe the modifications necessary to integrate LoRA into the model architecture, the training process, and how you would assess the effectiveness of this approach compared to traditional fine-tuning methods.

Solution 2: Low-Rank Adaptation (LoRA) enables efficient fine-tuning of large language models by introducing trainable parameters that interact with the pre-trained weights through low-rank matrices. To implement a summarisation system with LoRA using the CNN/DM or XSum dataset, follow these steps:

Step 1: Model Architecture Modification Modify the architecture of a pre-trained language model by adding LoRA modules to specific layers, typically attention and/or feed-forward layers. For each targeted layer, replace the weight matrix $W \in \mathbb{R}^{d \times d'}$ with $W + BA$, where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d'}$ are trainable low-rank matrices.

In the case of a decoder-only transformer-based LM, we can apply LoRA to the self-attention mechanism's key, value, and query matrices. For each matrix $\mathbf{W}$ (representing $\mathbf{W}_k$, $\mathbf{W}_v$, or $\mathbf{W}_q$), modify it as follows: $\mathbf{W}' = \mathbf{W} + \mathbf{BA}$, where $\mathbf{B}$ and $\mathbf{A}$ are the newly introduced trainable low-rank matrices.

0.1 Example

- For the key matrix ($\mathbf{W}_k$), modify it to $\mathbf{W}'_k = \mathbf{W}_k + \mathbf{B}_k\mathbf{A}_k$.
- Apply LoRA to the value ($\mathbf{W}_v$) and query ($\mathbf{W}_q$) matrices similarly, resulting in $\mathbf{W}'_v = \mathbf{W}_v + \mathbf{B}_v\mathbf{A}_v$ and $\mathbf{W}'_q = \mathbf{W}_q + \mathbf{B}_q\mathbf{A}_q$, respectively.

Step 2: Training Process During training, freeze the original model weights and only update the parameters in the low-rank matrices $\mathbf{B}$ and $\mathbf{A}$. This approach allows the model to adapt to the summarisation task without the computational overhead of training all parameters.

Step 3: Evaluation and Comparison Assess the effectiveness of the LoRA-based system using the same summarisation metrics as traditional fine-tuning methods, such as ROUGE scores. Compare the performance, training time, and parameter efficiency against a model where all its parameters were fine-tuned using gradient-based optimisation methods.

LoRA's advantage lies in its ability to fine-tune large models on specific tasks with significantly fewer trainable parameters, making it more efficient and accessible for researchers and practitioners with limited computational resources.