Fine-Tuning BERT for AI-Powered Text Summarization: A Step-by-Step Guide

BeastBoyJay Follow 4 min read · Nov 19, 2024




> *Have you ever been overwhelmed by lengthy articles and wished for concise, accurate summaries? Enter BERT, the AI powerhouse capable of understanding and summarizing text.*

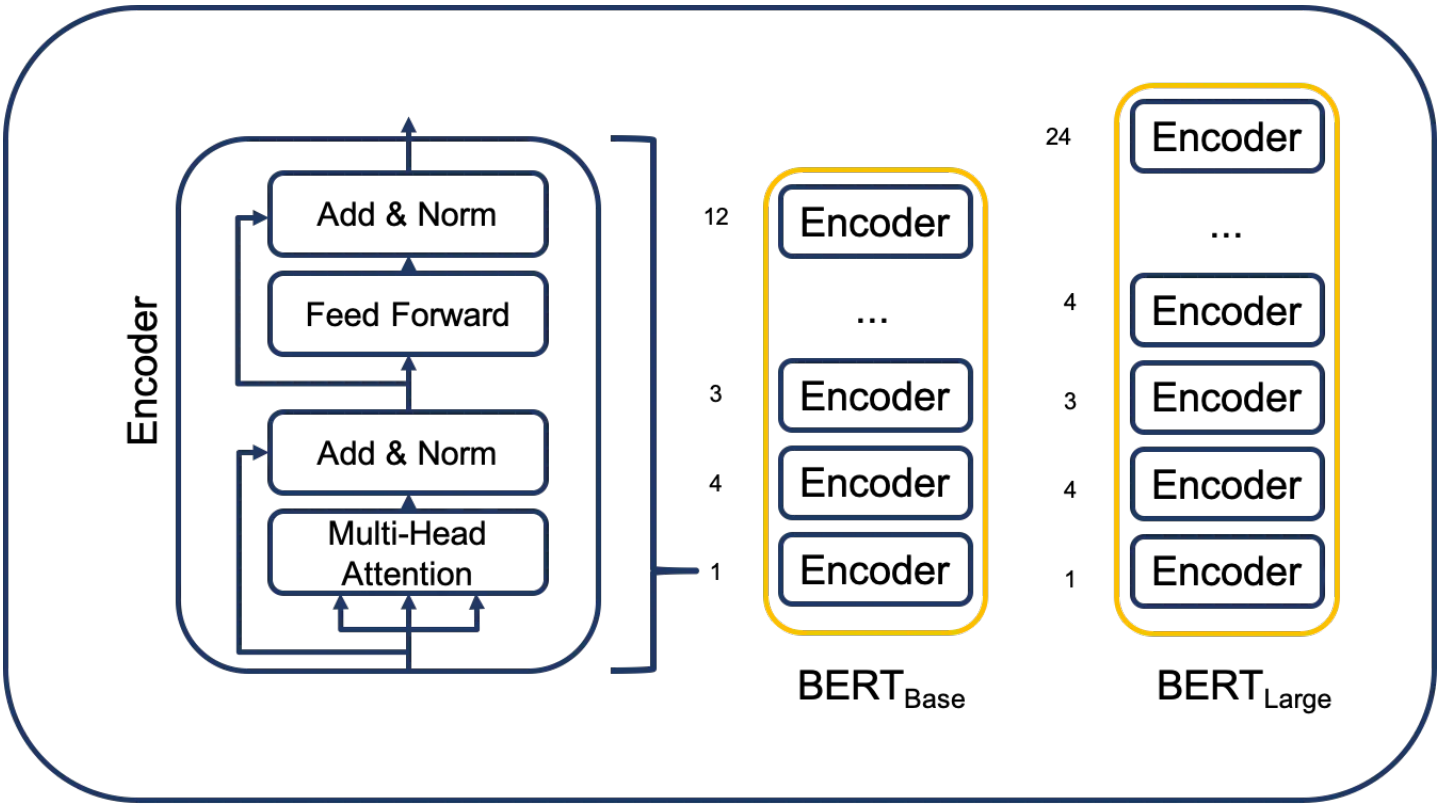
Text summarization has become an essential tool in our information-heavy world, helping us condense large amounts of text into concise, meaningful summaries. Imagine having an AI that can summarize research papers, news articles, or even novels for you. Enter **BERT,** a transformer model that has revolutionized natural language processing (NLP).

In this blog, I'll share how I fine-tuned a BERT-based Encoder-Decoder model to create an efficient text summarization tool. I'll also walk you through the code and process so you can replicate or enhance this project.

. . .

What is BERT?

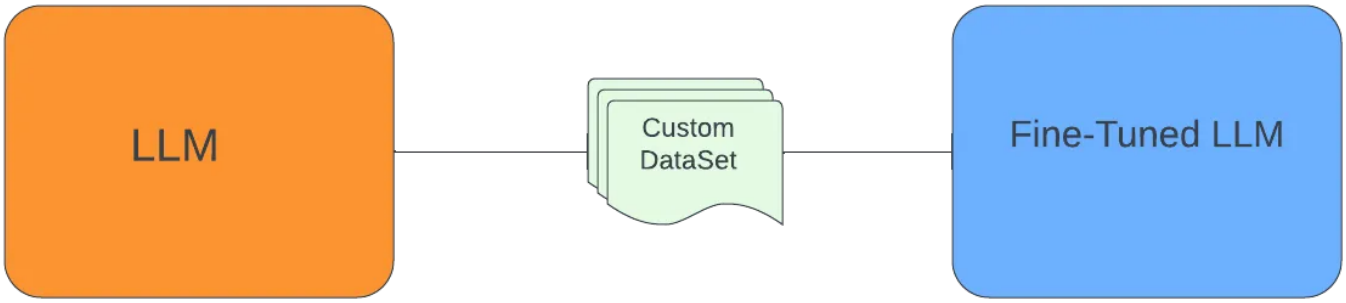
BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model designed to understand the context of words in a sentence by considering both the left and right context. Developed by Google, it has applications in a variety of NLP tasks like text classification, question answering, and, as we'll see, summarization.For more info:*BERT*



What is Fine-Tuning?

Fine-tuning involves taking a pre-trained model like BERT and adapting it to a specific task by training it further on domain-specific data. This process

leverages the general understanding of language already learned by BERT, saving time and computational resources.



. . .

Project

I developed a summarization pipeline using Hugging Face's `transformers` library. Below, I'll detail the training and inference phases.

Training Phase

Step 1: Model Setup

We use Hugging Face's **EncoderDecoderModel** to combine **BERT** as both an encoder and a decoder.Because single **BERT** has no capabilities for generating new content,In this case summary.

```
# Importing necessary libraries
from transformers import (
```

```
 BertTokenizer,
 EncoderDecoderModel,
 Trainer,
 TrainingArguments,
 DataCollatorForSeq2Seq,
)
```

Step 2: Dataset

The **CNN/DailyMail dataset** was chosen for this project. It contains news articles and summaries, making it ideal for training summarization models.

```
from datasets import load_dataset
# Load a small subset for quick experimentation
dataset = load_dataset("cnn_dailymail", "3.0.0", split="train[:1%]")# change 1
print(f"Loaded {len(dataset)} samples.")
```

Step 3: Tokenization

We preprocess the dataset by tokenizing the input articles and target summaries.A major step for the training, In this project i have used the **BERT** tokenizer which is available in the Hugging Face transformer library.

```
def preprocess_function(batch):
 inputs = tokenizer(
 batch["article"],# article as input in the model.
 max_length=512,
 truncation=True,
 padding="max_length",
 )
```

```
 labels = tokenizer(
 batch["highlights"],# highlights as a target for each input in the model.
 max_length=128,
 truncation=True,
 padding="max_length",
 )
 inputs["labels"] = labels["input_ids"]
 return inputs
tokenized_dataset = dataset.map(preprocess_function, batched=True)# Tokenizing
```

Step 4: Model Training

The **Trainer** class simplifies training with minimal setup.

```
training_args = TrainingArguments( # Traning Aruguments for the tranining
 output_dir="./results",
 per_device_train_batch_size=2,# Number of batches to train
 num_train_epochs=3,# Number of trainig epochs
 evaluation_strategy="epoch",
)
trainer = Trainer(
 model=model,
 args=training_args,
 train_dataset=tokenized_dataset,
 eval_dataset=tokenized_dataset,
 data_collator=DataCollatorForSeq2Seq(tokenizer, model=model),# this will help
)
trainer.train()
```

. . .

Inference Phase

Once the model is trained, we load it to generate summaries for new input texts.

```
from transformers import EncoderDecoderModel, BertTokenizer
class TextSummarizer:
 def __init__(self, model_path):
 self.tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
 self.model = EncoderDecoderModel.from_pretrained(model_path)

def summarize(self, text):
 inputs = self.tokenizer(
 text,
 return_tensors="pt",
 max_length=512,
 truncation=True,
 padding="max_length",
 )
 summary_ids = self.model.generate(
 inputs["input_ids"],
 attention_mask=inputs["attention_mask"],
 max_length=128,
 num_beams=4,
 )
 return self.tokenizer.decode(summary_ids[0], skip_special_tokens=True)

# Example usage
summarizer = TextSummarizer(model_path="./results/your/last/checkpoint")
text = "An example article text…"
print(summarizer.summarize(text))
```

Results

Example Input:

“Scientists have learned to supplement the sense of sight in numerous ways…”

Generated Summary:

“Scientists have developed tools to extend the sense of sight for various applications like microscopy and X-rays.”

. . .

Challenges and Insights

1. Small Dataset: Using only 1% of the dataset due to resource restrictions ,this required careful optimization of hyper-parameters.

2. Beam Search: Fine-tuning the beam size and length penalty significantly improved summary quality.

Note:

Impact of Beam Search:

Without Fine-Tuning:

Beam Size: 1 (greedy decoding), Length Penalty: 0.0

Output: "Scientists have tools."

With Fine-Tuning:

Beam Size: 4, Length Penalty: 1.5

Output: "Scientists have developed tools to extend the sense of sight for various applications like microscopy and X-rays."

• • •

Conclusion

Fine-tuning BERT for text summarization is a rewarding project that demonstrates the versatility of transformer models. With just a few lines of code, you can build an AI capable of simplifying information overload.

Try this on your own or custom dataset for **use cases tailored to your needs**. Imagine summarizing financial reports for investors, generating quick highlights for research papers, or even creating concise summaries for legal documents. The possibilities are endless!

Better yet, why stop at summarization? With a similar approach, you can explore fine-tuning BERT for **paraphrasing**, **question generation**, or even **creative writing assistance**. Take the first step and let your creativity guide your AI projects — there's a whole world of text-based applications waiting to be discovered!

Check out my full implementation on ***GitHub*** and let me know your thoughts or suggestions in the comments!

Transformers Bert Fine Tune Summary Encoder