BART Model

A transformer model combining BERT's text comprehension and GPT's text generation, ideal for summarization, translation, and question-answering tasks.

In [18]:
```
from transformers import BartTokenizer, BartForConditionalGeneration

# Initialize the BART tokenizer
tokenizer = BartTokenizer.from_pretrained('facebook/bart-large')

# Preprocessing function for tokenization
def preprocess_function(examples):
    inputs = []
    targets = []

    # Iterate over the batched data
    for i in range(len(examples['context'])):
        context = examples['context'][i]  # Access each context in the batch
        question = examples['question'][i]  # Access each question in the batch
        answer = examples['answer'][i]  # Access each answer in the batch

        # Combine the context and question as input
        inputs.append(f"Context: {context} Question: {question}")
        targets.append(answer)  # Use the answer as the target

    # Tokenize the inputs (contexts + questions) with padding and truncation
    model_inputs = tokenizer(
        inputs, max_length=1024, truncation=True, padding='max_length'  # Ensure un
    )

    # Tokenize the targets (answers) with padding and truncation
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            targets, max_length=128, truncation=True, padding='max_length'  # Ensur
        )

    # Fix the nested list issue by using 'input_ids' directly
    model_inputs['labels'] = labels['input_ids']
    return model_inputs
```

In [19]:
```
from datasets import load_dataset

# Load the dataset and ensure caching is disabled
dataset = load_dataset('json', data_files={'train': 'sb1.json'})
# Apply the preprocessing function with caching completely disabled
tokenized_datasets = dataset.map(
    preprocess_function,
    batched=True,
    load_from_cache_file=False,
    keep_in_memory=True  # Ensure the dataset is processed in memory without cache
)
```

```
Generating train split: 0 examples [00:00, ? examples/s]
Map:   0%|          | 0/47 [00:00<?, ? examples/s]
```

```
C:\Users\SColbe01\AppData\Local\miniconda3\Lib\site-packages\transformers\tokenization_utils_base.py:4117: UserWarning: `as_target_tokenizer` is deprecated and will be removed in v5 of Transformers. You can tokenize your labels by using the argument `text_target` of the regular `__call__` method (either in the same call as your input texts if you use the same keyword arguments, or in a separate call.
  warnings.warn(
```

In [20]:
```python
from transformers import BartTokenizer, BartForConditionalGeneration

# Load the model and tokenizer from local paths
tokenizer = BartTokenizer.from_pretrained('bart-large')
model = BartForConditionalGeneration.from_pretrained('bart-large')
```

```
C:\Users\SColbe01\AppData\Local\miniconda3\Lib\site-packages\transformers\tokenization_utils_base.py:1617: FutureWarning: `clean_up_tokenization_spaces` was not set. It will be set to `True` by default. This behavior will be deprecated in transformers v4.45, and will be then set to `False` by default. For more details check this issue: https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

In [21]:
```python
from transformers import Trainer, TrainingArguments

# Define the training arguments
training_args = TrainingArguments(
    output_dir='./results',              # Output directory
    eval_strategy='epoch',          # Evaluate once per epoch
    learning_rate=5e-5,                  # Learning rate
    per_device_train_batch_size=4,       # Training batch size
    per_device_eval_batch_size=4,        # Evaluation batch size
    num_train_epochs=3,                  # Number of epochs
    weight_decay=0.01,                   # Weight decay
    save_steps=500,                      # Save every 500 steps
    logging_dir='./logs',                # Logging directory
    logging_steps=100                    # Log every 100 steps
)

# Initialize the Trainer with the model, tokenizer, and the training arguments
trainer = Trainer(
    model=model,                         # Use the model defined in your code
    args=training_args,                  # Use the training arguments defined above
    train_dataset=tokenized_datasets['train'],  # Use your tokenized training datas
    eval_dataset=tokenized_datasets['train'],   # Evaluation dataset (could replace
    tokenizer=tokenizer                  # Use your tokenizer
)

# Start training the model
trainer.train()

# Save the fine-tuned model and tokenizer for future use
model.save_pretrained('./fine_tuned_bart_model')  # Save the model to the specified
tokenizer.save_pretrained('./fine_tuned_bart_model')  # Save the tokenizer

print("Training completed, and the fine-tuned BART model has been saved.")
```

[36/36 33:53, Epoch 3/3]

Epoch	Training Loss	Validation Loss
1	No log	9.661262
2	No log	6.458136
3	No log	6.066216

```
Training completed, and the fine-tuned BART model has been saved.
```

In [22]:

```
Chat with your BART model! (type 'exit' to stop)
Answer:

Answer:

Exiting chat.
```

In [5]:

```python
from transformers import BartTokenizer, BartForConditionalGeneration

# Step 1: Load the fine-tuned BART model and tokenizer
model = BartForConditionalGeneration.from_pretrained('./fine_tuned_bart_model')
tokenizer = BartTokenizer.from_pretrained('./fine_tuned_bart_model')

# Step 2: Define a chat function with debugging
def chat_with_model():
    print("Chat with your BART model! (type 'exit' to stop)")

    while True:
        # Get user input
        context = input("Context: ")
        question = input("Question: ")

        if context.lower() == 'exit' or question.lower() == 'exit':
            print("Exiting chat.")
            break

        # Prepare the input for the model
        input_text = f"Context: {context} Question: {question}"
        # print(f"Input Text: {input_text}")  # Debugging to check input format
        inputs = tokenizer(input_text, return_tensors="pt", max_length=1024, trunca

        # Check if the inputs are tokenized correctly
        # print(f"Tokenized Input: {inputs}")  # Debugging to check tokenization

        # Generate the model's response
        outputs = model.generate(inputs['input_ids'], max_length=50, num_beams=5, e

        # Decode and print the model's response
        answer = tokenizer.decode(outputs[0], skip_special_tokens=True)
        print(f"Answer: {answer}\n")

# Step 3: Start the chat
chat_with_model()
```

```
Chat with your BART model! (type 'exit' to stop)
Answer: Denver Broncos

Answer: Denver Broncos

Answer: Denver Broncos

Answer:

Answer: Denver Broncos

Answer: 2014

Exiting chat.
```

In [ ]: