Experiment No 9

NAME:DIVYESH KHUNT SAPID:60009210116 BATCH:D12

Aim: To fine-tune a BERT model on custom data for performing a question-answering task.

Introduction:

Question Answering (QA) systems aim to provide precise answers to questions posed in natural language. Fine-tuning BERT (Bidirectional Encoder Representations from Transformers) on custom datasets allows for developing QA models tailored to specific domains. By training on relevant question-answer pairs, the model learns to locate and provide precise answers based on context.

BERT is especially effective for this task due to its ability to understand contextual embedding's, making it well-suited for handling the complexities of language. Fine-tuning involves training the model on a labeled dataset where each entry includes a passage, a question, and the corresponding answer span within the passage.

Fine-Tuning BERT for Question Answering

1. **Model Selection**: Use the pre-trained BERT model as the base, available through Hugging Face's Transformers library. Models such as bert-large-uncased are commonly used for QA tasks.
2. **Data Preparation**: Format the custom data into question-passage-answer triplets. The passage contains the context, and the answer is a span within this context.
3. **Tokenization**: Tokenize the data using the BERT tokenizer, which splits text into tokens compatible with BERT's vocabulary. Additionally, align each token with its position in the original text to aid in identifying the answer span.
4. **Fine-Tuning Process**: Train the BERT model on the formatted dataset, optimizing it to locate the start and end tokens of the answer span within the context passage.

Lab Experiment

Step 1: Install required libraries.

Step 2: Load a pre-trained BERT model for QA and tokenizer.

Step 3: Prepare the custom data in question-passage-answer format, ensuring each answer is marked with its start and end positions in the passage.

Step 4: Tokenize the data and format it for BERT.

Step 5: Fine-tune the model.

Step 6: Test the model with new questions and passages.

```
!pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/di
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-p
```

```
[3] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns

    import torch
    from transformers import BertForQuestionAnswering
    from transformers import BertTokenizer
```

```
coqa = pd.read_json('http://downloads.cs.stanford.edu/nlp/data/coqa/coqa-train-v1.0.json')
coqa.head()
```

	version	data
0	1	{'source': 'wikipedia', 'id': '3zotghdk5ibi9ce...
1	1	{'source': 'cnn', 'id': '3wj1oxy92agboo5nlq4r7...
2	1	{'source': 'gutenberg', 'id': '3bdcf01ogxu7zdn...
3	1	{'source': 'cnn', 'id': '3ewijtffvo7wwchw6rtya...
4	1	{'source': 'gutenberg', 'id': '3urfvvm165iantk...

```
[8]  del coqa["version"]

     cols = ["text","question","answer"]


     comp_list = []
     for index, row in coqa.iterrows():
         for i in range(len(row["data"]["questions"])):
             temp_list = []

             temp_list.append(row["data"]["story"])
             temp_list.append(row["data"]["questions"][i]["input_text"])
             temp_list.append(row["data"]["answers"][i]["input_text"])
             comp_list.append(temp_list)

     new_df = pd.DataFrame(comp_list, columns=cols)
     new_df.to_csv("CoQA_data.csv", index=False)
```

```
data = pd.read_csv("CoQA_data.csv")
data.head()
```

	text	question	answer
0	The Vatican Apostolic Library (), more commonl...	When was the Vat formally opened?	It was formally established in 1475
1	The Vatican Apostolic Library (), more commonl...	what is the library for?	research
2	The Vatican Apostolic Library (), more commonl...	for what subjects?	history, and law

```
[11] print("qna", len(data))

qna 108647
```

```
from transformers import BertForQuestionAnswering, BertTokenizer

model_name = "bert-large-uncased-whole-word-masking-finetuned-squad"  # BERT fine-tuned on SQuAD
model = BertForQuestionAnswering.from_pretrained(model_name)
tokenizer = BertTokenizer.from_pretrained(model_name)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret i
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100% 443/443 [00:00<00:00, 9.16kB/s]
model.safetensors: 100% 1.34G/1.34G [00:14<00:00, 202MB/s]
Some weights of the model checkpoint at bert-large-uncased-whole-word-masking-finetuned-squad were not used when initializing BertForQuesti
- This IS expected if you are initializing BertForQuestionAnswering from the checkpoint of a model trained on another task or with another
- This IS NOT expected if you are initializing BertForQuestionAnswering from the checkpoint of a model that you expect to be exactly identi
tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 3.16kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 8.32MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 15.9MB/s]
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `clean_up_tokenization_spaces` was not
  warnings.warn(
```

```
[15] model = BertForQuestionAnswering.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
     tokenizer = BertTokenizer.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
```

```
Some weights of the model checkpoint at bert-large-uncased-whole-word-masking-finetuned-squad were not used when initializing BertFo
- This IS expected if you are initializing BertForQuestionAnswering from the checkpoint of a model trained on another task or with a
- This IS NOT expected if you are initializing BertForQuestionAnswering from the checkpoint of a model that you expect to be exactly
```

```
random_num = np.random.randint(0,len(data))
question = data["question"][random_num]
text = data["text"][random_num]
answer = data["answer"][random_num]
print(text)
print(question)
print(answer)
```

```
Shane Thomas is a 10-year-old pianist from England. He's being called the next Mozart because of his amazing abilities.

He has only been having piano lessons for four months, and practices four hours a week, but he has already played difficult classica

When Shane was three years old, he said that he could play the piano, but nobody took him seriously. At school, he could listen to t

His tutor, Richard Goffin-Lecar, says he is like Amadeus Mozart, who lived during the 18thcentury in Salzburg, Austria, and was one

His father, a single parent with two other children, says that although he has little money, he wants to send Shane to a good music
When did his lessons start?
four months ago
```

```
tokens = tokenizer.convert_ids_to_tokens(input_ids)
for token, id in zip(tokens, input_ids):
    print('{:8}{:8,}'.format(token,id))
```

```
money       2,769
,           1,010
he          2,002
wants       4,122
to          2,000
send        4,604
shane       8,683
to          2,000
a           1,037
good        2,204
music       2,189
school      2,082
.           1,012
"           1,000
i           1,045
'           1,005
m           1,049
a           1,037
single      2,309
father      2,269
,           1,010
but         2,021
```

```
[29] output = model(torch.tensor([input_ids]),  token_type_ids=torch.tensor([segment_ids]))

answer_start = torch.argmax(output.start_logits)
answer_end = torch.argmax(output.end_logits)
if answer_end >= answer_start:
    answer = " ".join(tokens[answer_start:answer_end+1])
else:
    print("I am unable to find the answer to this question. Can you please ask another question?")

print("\nQuestion:\n{}".format(question.capitalize()))
print("\nAnswer:\n{}.".format(answer.capitalize()))
```

```
Question:
When did his lessons start?

Answer:
Four months.
```

```
def question_answer(question, text):
    #tokenize question and text as a pair
    input_ids = tokenizer.encode(question, text)
    #string version of tokenized ids
    tokens = tokenizer.convert_ids_to_tokens(input_ids)
    #first occurence of [SEP] token
    sep_idx = input_ids.index(tokenizer.sep_token_id)
    num_seg_a = sep_idx+1
    num_seg_b = len(input_ids) - num_seg_a
    #list of 0s and 1s for segment embeddings
    segment_ids = [0]*num_seg_a + [1]*num_seg_b
    assert len(segment_ids) == len(input_ids)
    #model output using input_ids and segment_ids
    output = model(torch.tensor([input_ids]), token_type_ids=torch.tensor([segment_ids]))
    #reconstructing the answer
    answer_start = torch.argmax(output.start_logits)
    answer_end = torch.argmax(output.end_logits)
    if answer_end >= answer_start:
        answer = tokens[answer_start]
        for i in range(answer_start+1, answer_end+1):
            if tokens[i][0:2] == "##":
                answer += tokens[i][2:]
            else:
                answer += " " + tokens[i]
    if answer.startswith("[CLS]"):
        answer = "Unable to find the answer to your question."

    print("\nPredicted answer:\n{}".format(answer.capitalize()))
```

```
[35] text = """New York (CNN) -- More than 80 Michael Jackson collectibles -- including the late pop star's famous rhinestone-studded glove f
     question = "Where was the Auction held?"
     question_answer(question, text)
     #original answer from the dataset
     print("Original answer:\n", data.loc[data["question"] == question]["answer"].values[0])

Predicted answer:
Hard rock cafe in new york ' s times square
Original answer:
 Hard Rock Cafe
```

```
text = input("Please enter your text: \n")
question = input("\nPlease enter your question: \n")
while True:
    question_answer(question, text)

    flag = True
    flag_N = False

    while flag:
        response = input("\nDo you want to ask another question based on this text (Y/N)? ")
        if response[0] == "Y":
            question = input("\nPlease enter your question: \n")
            flag = False
        elif response[0] == "N":
            print("\nBye!")
            flag = False
            flag_N = True

    if flag_N == True:
      break
```

```
Please enter your text:
The Vatican Apostolic Library, often referred to simply as the Vatican Library, is the library of the Holy Se

Please enter your question:
When was the Vatican Apostolic Library founded?

Predicted answer:
15th century

Do you want to ask another question based on this text (Y/N)?
```