Bhuvi Ghosh
60009210191

Experiment No 9

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.

Using Simple Transformers library:

```
train_contexts = [
    "Nokia C12 Android 12 (Go Edition) Smartphone, All-Day Battery, 4GB RAM (2GB RAM + 2GB Virtual RAM) + 64GB Capacity | Light Mint",
    "Nokia G21 Android Smartphone, Dual SIM, 3-Day Battery Life, 6GB RAM + 128GB Storage, 50MP Triple AI Camera | Nordic Blue",
    "realme narzo 50i Prime (Dark Blue 4GB RAM+64GB Storage) Octa-core Processor | 5000 mAh Battery",
    "realme narzo N53 (Feather Gold, 4GB+64GB) 33W Segment Fastest Charging | Slimmest Phone in Segment | 90 Hz Smooth Display",
    "realme narzo N55 (Prime Blue, 4GB+64GB) 33W Segment Fastest Charging | Super High-res 64MP Primary AI Camera",
    "Redmi 9A Sport (Carbon Black, 2GB RAM, 32GB Storage) | 2GHz Octa-core Helio G25 Processor | 5000 mAh Battery",
    "Redmi 11 Prime 5G (Thunder Black, 4GB RAM, 64GB Storage) | Prime Design | MTK Dimensity 700 | 50 MP Dual Cam | 5000mAh | 7 Band 5G",
    "Redmi 12C (Royal Blue, 4GB RAM, 64GB Storage) | High Performance Mediatek Helio G85 | Big 17cm(6.71) HD+ Display with 5000mAh(typ) Battery",
    "Redmi A1 (Light Green, 2GB RAM 32GB ROM) | Segment Best AI Dual Cam | 5000mAh Battery | Leather Texture Design | Android 12",
    "Samsung Galaxy M04 Light Green, 4GB RAM, 64GB Storage | Upto 8GB RAM with RAM Plus | MediaTek Helio P35 Octa-core Processor | 5000 mAh Battery | 13MP Dual Camera",
    "Samsung Galaxy M13 (Midnight Blue, 4GB, 64GB Storage) | 6000mAh Battery | Upto 8GB RAM with RAM Plus",
    "Tecno Camon 20 (Serenity Blue, 8GB RAM,256GB Storage)|16GB Expandable RAM | 64MP RGBW Rear Camera|6.67 FHD+ Big AMOLED with in-Display Fingerprint Sensor",
]
```

```
[3] len(train_contexts)
```

```
12
```

```
train_questions_answers = [
    {
        "context_index": 0,
        "question": "What is the operating system of the Nokia C12 smartphone?",
        "answer": "Android 12 (Go Edition)"
    },
    {
        "context_index": 0,
        "question": "How much RAM does the Nokia C12 have?",
        "answer": "4GB"
    },
    {
        "context_index": 0,
        "question": "Does the Nokia C12 have virtual RAM?",
        "answer": "(2GB RAM + 2GB Virtual RAM)"
    },
    {
        "context_index": 0,
        "question": "What is the total capacity of the Nokia C12?",
        "answer": "64GB"
    },
    {
        "context_index": 0,
        "question": "What is the color option available for the Nokia C12?",
        "answer": "Light Mint"
    },
```

Data format conversion

- Now, we will transform the training data into the format required by SimpleTransformers, which will be using to train the BERT model for our qa task
- After trasnsformation with loops on each context, their questions and anwers, we will dump the final formated train data to a json file for reuse.

```
[5] train_data = []
    train_contexts_data = []

    for i, context in enumerate(train_contexts):
        qas = []
        for qa in train_questions_answers:
            if qa["context_index"] == i:
                answer_start = context.find(qa["answer"])
                if answer_start != -1:
                    qas.append({
                        "id": str(len(qas) + 1).zfill(5),
                        "is_impossible": False,
                        "question": qa["question"],
                        "answers": [
                            {
                                "text": qa["answer"],
                                "answer_start": answer_start,
                            }
                        ],
                    })
        train_contexts_data.append({
            "context": context,
            "qas": qas,
        })

    train_data.extend(train_contexts_data)
```

```
[6] train_data[0]
```

```
{'context': 'Nokia C12 Android 12 (Go Edition) Smartphone, All-Day Battery, 4GB RAM (2GB RAM + 2GB Virtual RAM) + 64GB Capacity | Light Min
 'qas': [{'id': '00001',
   'is_impossible': False,
   'question': 'What is the operating system of the Nokia C12 smartphone?',
   'answers': [{'text': 'Android 12 (Go Edition)', 'answer_start': 10}]},
  {'id': '00002',
   'is_impossible': False,
   'question': 'How much RAM does the Nokia C12 have?',
   'answers': [{'text': '4GB', 'answer_start': 63}]},
  {'id': '00003',
   'is_impossible': False,
   'question': 'Does the Nokia C12 have virtual RAM?',
   'answers': [{'text': '(2GB RAM + 2GB Virtual RAM)', 'answer_start': 71}]},
  {'id': '00004',
   'is_impossible': False,
   'question': 'What is the total capacity of the Nokia C12?',
   'answers': [{'text': '64GB', 'answer_start': 101}]},
  {'id': '00005',
   'is_impossible': False,
   'question': 'What is the color option available for the Nokia C12?',
   'answers': [{'text': 'Light Mint', 'answer_start': 117}]}]}
```

- The above example gives us the training data in suitable format for index id = 0
- Now we will save the training data into a json file named 'amazon_data_train'

```
import json

with open('amazon_data_train.json', 'w', encoding='utf-8') as f:
    json.dump(train_data, f, ensure_ascii=False, indent=4)
```

Now, we will use a different set of contexts, with their ground truth question answers.

```
test_contexts = [
    "Redmi Note 11 (Space Black, 4GB RAM, 64GB Storage)|90Hz FHD+ AMOLED Display | Qualcomm® Snapdragon™ 680-6nm | 33W Charger Included",
    "Redmi Note 10S (Deep Sea Blue, 6GB RAM, 64GB Storage) - Super Amoled Display | 64 MP Quad Camera | 6 Month Free Screen Replacement (Prime
    "Lava Blaze 5G (Glass Green, 6GB RAM, UFS 2.2 128GB Storage) | 5G Ready | 50MP AI Triple Camera | Upto 11GB Expandable RAM | Charger Inclu
    "Oppo A78 5G (Glowing Black, 8GB RAM, 128 Storage) | 5000 mAh Battery with 33W SUPERVOOC Charger| 50MP AI Camera | 90Hz Refresh Rate | wit
]

test_questions_answers = [
    {
        "context_index": 0,
        "question": "What is the model name of the Redmi smartphone?",
        "answer": "Redmi Note 11"
    },
    {
        "context_index": 0,
        "question": "What is the color option available for the Redmi Note 11?",
        "answer": "Space Black"
    },
    {
        "context_index": 0,
        "question": "How much RAM does the Redmi Note 11 have?",
        "answer": "4GB"
    },
    {
        "context_index": 0,
        "question": "What is the storage capacity of the Redmi Note 11?",
        "answer": "64GB"
    },
    {
        "context_index": 0,
        "question": "What is the display feature of the Redmi Note 11?",
        "answer": "90Hz FHD+ AMOLED Display"
    },
```

```
[9] test_data = []
    test_contexts_data = []

    for i, context in enumerate(test_contexts):
        qas = []
        for qa in test_questions_answers:
            if qa["context_index"] == i:
                answer_start = context.find(qa["answer"])
                if answer_start != -1:
                    qas.append({
                        "id": str(len(qas) + 1).zfill(5),
                        "is_impossible": False,
                        "question": qa["question"],
                        "answers": [
                            {
                                "text": qa["answer"],
                                "answer_start": answer_start,
                            }
                        ],
                    })
        test_contexts_data.append({
            "context": context,
            "qas": qas,
        })

    test_data.extend(test_contexts_data)
```

```
[10] with open('amazon_data_test.json', 'w', encoding='utf-8') as f:
         json.dump(test_data, f, ensure_ascii=False, indent=4)
```

- Now we have two datasets in proper format as json files= amazon_data_train.json and amazon_data_test.json.
- We will use them to fine tune our BERT model

Department of Computer Science and Engineering (Data Science)
Lab Manual

Sub: Advanced Computational Linguistics **Year/Sem: BTech/VII**

```
import json

with open(r"amazon_data_train.json", "r") as read_file:
    train = json.load(read_file)

with open(r"amazon_data_test.json", "r") as read_file:
    test = json.load(read_file)
```

```
[13] train[0]
```

```
{'context': 'Nokia C12 Android 12 (Go Edition) Smartphone, All-Day Battery, 4GB RAM (2GB RAM + 2GB Virtual RAM) + 64GB Capacity |
 'qas': [{'id': '00001',
   'is_impossible': False,
   'question': 'What is the operating system of the Nokia C12 smartphone?',
   'answers': [{'text': 'Android 12 (Go Edition)', 'answer_start': 10}]},
  {'id': '00002',
   'is_impossible': False,
   'question': 'How much RAM does the Nokia C12 have?',
   'answers': [{'text': '4GB', 'answer_start': 63}]},
  {'id': '00003',
   'is_impossible': False,
   'question': 'Does the Nokia C12 have virtual RAM?',
   'answers': [{'text': '(2GB RAM + 2GB Virtual RAM)', 'answer_start': 71}]},
  {'id': '00004',
   'is_impossible': False,
   'question': 'What is the total capacity of the Nokia C12?',
   'answers': [{'text': '64GB', 'answer_start': 101}]},
  {'id': '00005',
   'is_impossible': False,
   'question': 'What is the color option available for the Nokia C12?',
   'answers': [{'text': 'Light Mint', 'answer_start': 117}]}]}
```

```
[14] test[0]
```

```
{'context': 'Redmi Note 11 (Space Black, 4GB RAM, 64GB Storage)|90Hz FHD+ AMOLED Display | Qualcomm® Snapdragon™ 680-6nm | 33W Cha
 'qas': [{'id': '00001',
   'is_impossible': False,
   'question': 'What is the model name of the Redmi smartphone?',
   'answers': [{'text': 'Redmi Note 11', 'answer_start': 0}]},
```

```
[15] import logging
     from simpletransformers.question_answering import QuestionAnsweringModel, QuestionAnsweringArgs
```

We will use 340M parameter bert-large-uncased.

I tried 10 epochs in the beginning, but had to settle with 25 epochs

```
[16] #train_args are the parameters the QuestionAnswerringModel will use
     train_args = {
         'overwrite_output_dir': True,
         "evaluate_during_training": True,
         "max_seq_length": 128,
         "num_train_epochs": 25, #25, after experimentations
         "evaluate_during_training_steps": 500,
         "save_model_every_epoch": False,
         "save_eval_checkpoints": False,
         "n_best_size":16, #batch_size is another important argument
         "train_batch_size": 16,
         "eval_batch_size": 16
     }
```

```
[17] model = QuestionAnsweringModel("bert",
                                    "bert-large-cased",
                                    args = train_args,
                                    use_cuda=True) # I will use GPU for faster performance
```

```
/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
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% 762/762 [00:00<00:00, 14.9kB/s]
model.safetensors: 100% 1.34G/1.34G [00:36<00:00, 36.2MB/s]
```

```
[18] model.train_model(train, eval_data=test)

convert squad examples to features: 100%|| 82/82 [00:00<00:00, 601.61it/s]
add example index and unique id: 100%|| 82/82 [00:00<00:00, 428844.05it/s]
Epoch 25 of 25: 100% 25/25 [03:20<00:00, 2.65s/it]
/usr/local/lib/python3.10/dist-packages/simpletransformers/question_answering/question_answering_model.py:697: FutureWarning: `tc
  scaler = amp.GradScaler()
Epochs 1/25. Running Loss:  3.8687: 100% 6/6 [00:03<00:00, 2.58it/s]
/usr/local/lib/python3.10/dist-packages/simpletransformers/question_answering/question_answering_model.py:720: FutureWarning: `tc
  with amp.autocast():

convert squad examples to features: 100%|| 31/31 [00:00<00:00, 427.60it/s]

add example index and unique id: 100%|| 31/31 [00:00<00:00, 199117.04it/s]
Running Evaluation: 100% 2/2 [00:00<00:00, 11.87it/s]
/usr/local/lib/python3.10/dist-packages/simpletransformers/question_answering/question_answering_model.py:1184: FutureWarning: `t
  with amp.autocast():
Epochs 2/25. Running Loss:  3.2856: 100% 6/6 [00:01<00:00, 3.44it/s]

convert squad examples to features: 100%|| 31/31 [00:00<00:00, 506.11it/s]

add example index and unique id: 100%|| 31/31 [00:00<00:00, 244404.93it/s]
Running Evaluation: 100% 2/2 [00:00<00:00, 11.76it/s]

Epochs 3/25. Running Loss:  2.3560: 100% 6/6 [00:01<00:00, 3.31it/s]

convert squad examples to features: 100%|| 31/31 [00:00<00:00, 522.52it/s]

add example index and unique id: 100%|| 31/31 [00:00<00:00, 205408.25it/s]
Running Evaluation: 100% 2/2 [00:00<00:00, 11.73it/s]

Epochs 4/25. Running Loss:  2.1818: 100% 6/6 [00:02<00:00, 3.23it/s]

convert squad examples to features: 100%|| 31/31 [00:00<00:00, 516.67it/s]
```

```
[19] # Evaluate the model
     result, texts = model.eval_model(test)

convert squad examples to features: 100%|| 31/31 [00:00<00:00, 134.82it/s]
add example index and unique id: 100%|| 31/31 [00:00<00:00, 242129.28it/s]
Running Evaluation: 100% 2/2 [00:00<00:00, 6.27it/s]
```

```
[20] print(result)

{'correct': 5, 'similar': 4, 'incorrect': 0, 'eval_loss': -7.48828125}
```

MODEL INFERENCE

LET'S TEST OUR BEST MODEL WITH THE QUESTION: What is the model name of the Samsung smartphone?

```
[22] # Load model from training checkpoint
     from simpletransformers.question_answering import QuestionAnsweringModel, QuestionAnsweringArgs

     model = QuestionAnsweringModel("bert", "/content/outputs/best_model")


     # Make predictions with the model
     to_predict = [
         {
             "context": "Samsung Galaxy M14 5G (Smoky Teal, 6GB, 128GB Storage) | 50MP Triple Cam | 6000 mAh Battery | 5nm Octa-Core P
             "qas": [
                 {
                     "question": "What is the model name of the Samsung smartphone?",
                     "id": "0",
                 }
             ],
         }
     ]
```

```
answers, probabilities = model.predict(to_predict, n_best_size=2)
print(answers)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 7724.32it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 8830.11it/s]
Running Prediction: 100% 1/1 [00:00<00:00, 15.21it/s]
[{'id': '0', 'answer': ['Samsung Galaxy M14', 'Samsung Galaxy M14 5G']}]
/usr/local/lib/python3.10/dist-packages/simpletransformers/question_answering/question_answering_model.py:1348: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated
  with amp.autocast():
```