PROJECT PHASE-2

-JAYAPRAKASH.M

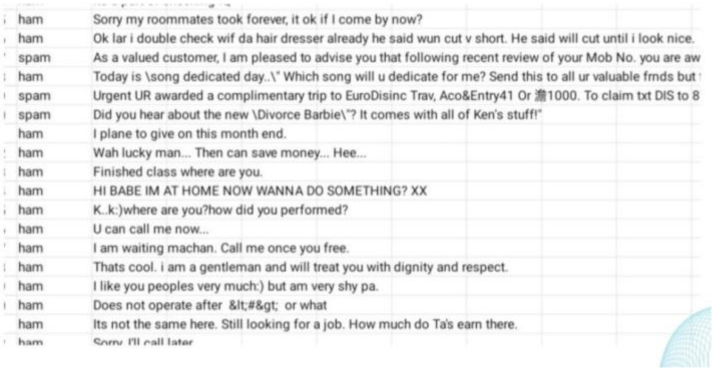
BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER

BERT- Building a spam classifier using BERT involves several steps. BERT (Bidirectional Encoder Representations from Transformers) is a powerful pre-trained language model that can be fine-tuned for various NLP tasks, including spam detection. Here's a high-level overview of the process:

1. **Data Collection and Preprocessing**
2. **Fine-tuning BERT**
3. **Feature Extraction**
4. **Training**
5. **Fine-tuning Parameters**
6. **Evaluation**
7. **Testing**
8. **Deployment**
9. **Monitoring and Maintenance**

1. **Data Collection and Preprocessing**:

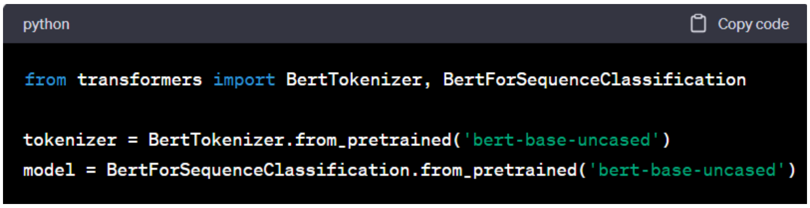
- Gather a dataset containing labeled examples of spam and non-spam (ham) messages.
- Preprocess the data, including tasks like lowercasing, removing special characters, and tokenizing the text.



ham	Sorry my roommates took forever, it ok if I come by now?
ham	Ok lar i double check wif da hair dresser already he said wun cut v short. He said will cut until i look nice.
spam	As a valued customer, I am pleased to advise you that following recent review of your Mob No. you are aw
ham	Today is \song dedicated day..\" Which song will u dedicate for me? Send this to all ur valuable frnds but
spam	Urgent UR awarded a complimentary trip to EuroDisinc Trav, Aco&Entry41 Or 澹1000. To claim txt DIS to 8
spam	Did you hear about the new \Divorce Barbie\"? It comes with all of Ken's stuff!"
ham	I plane to give on this month end.
ham	Wah lucky man... Then can save money... Hee...
ham	Finished class where are you.
ham	HI BABE IM AT HOME NOW WANNA DO SOMETHING? XX
ham	K..k:)where are you?how did you performed?
ham	U can call me now...
ham	I am waiting machan. Call me once you free.
ham	Thats cool. i am a gentleman and will treat you with dignity and respect.
ham	I like you peoples very much:) but am very shy pa.
ham	Does not operate after &lt;#&gt; or what
ham	Its not the same here. Still looking for a job. How much do Ta's earn there.
ham	Sorry, I'll call later

2. **Fine-tuning BERT**:

- Fine-tuning involves training BERT on your specific spam classification task.
- You'll need to add a classification layer on top of the pre-trained BERT model. This layer will have two output nodes (spam or non-spam).
- Initialize the classification layer with random weight.



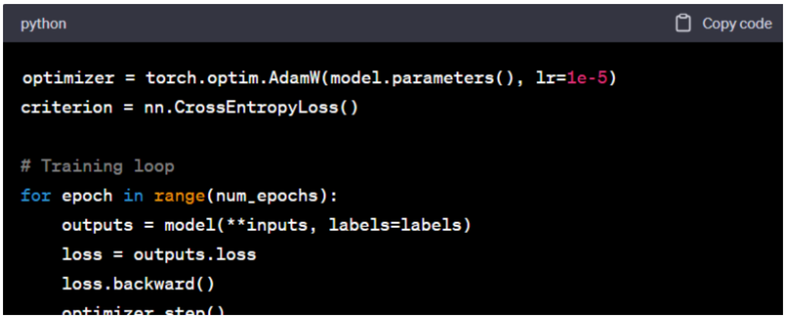
```python
from transformers import BertTokenizer, BertForSequenceClassification

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
```

3. **Feature Extraction**:

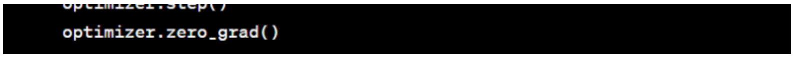
- Use the pre-trained BERT model to convert the text into high-dimensional embeddings.
- These embeddings capture semantic information about the text.

4. **Training**:

- Split your dataset into training and validation sets.
- Train the model using the training set. The loss is computed using a suitable loss function (e.g., binary cross-entropy).
- Use the validation set to monitor the performance and prevent overfitting.



```python
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-5)
criterion = nn.CrossEntropyLoss()

# Training loop
for epoch in range(num_epochs):
    outputs = model(**inputs, labels=labels)
    loss = outputs.loss
    loss.backward()
    optimizer.step()
```



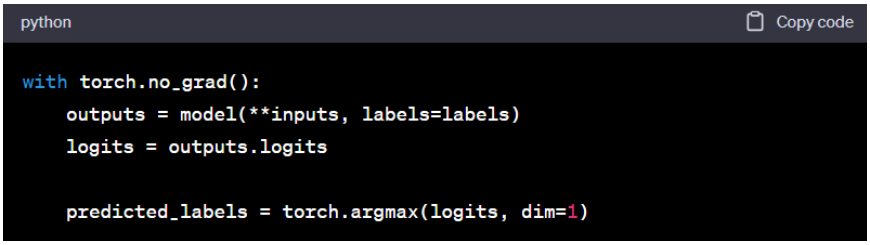
```
    optimizer.step()
    optimizer.zero_grad()
```

5. **Fine-tuning Parameters**:

- Experiment with different hyperparameters like learning rate, batch size, and number of epochs to optimize performance.

6. **Evaluation**:

- Evaluate the model using metrics like accuracy, precision, recall, F1-score, etc.
- Additionally, you can use techniques like cross-validation to get a more robust estimate of performance.



```python
with torch.no_grad():
    outputs = model(**inputs, labels=labels)
    logits = outputs.logits

    predicted_labels = torch.argmax(logits, dim=1)
```

7. **Testing**:

- Use a separate test set to get an unbiased estimate of the model's performance.

8. **Deployment**:

- Once satisfied with the model's performance, deploy it in a production environment. This could be on a server, cloud platform, or even on edge devices depending on your specific use case.

9. **Monitoring and Maintenance**:

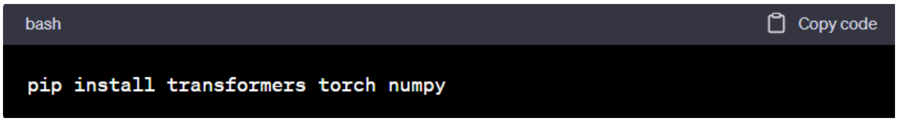
- Regularly monitor the model's performance in the real-world setting. If the data distribution changes, retraining may be necessary.

STEPS To build an AI-powered spam classifier using BERT

1. **Setting Up the Environment**
2. **Load and Preprocess Data**
3. **Fine-tuning BERT**
4. **Tokenization and Formatting**
5. **Train-Test Split**

1. **Setting Up the Environment**:

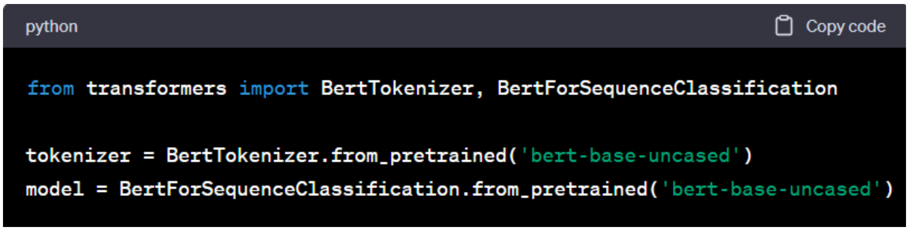
- Install the necessary libraries: transformers, torch, numpy, and any other dependencies you might need.



```bash
pip install transformers torch numpy
```

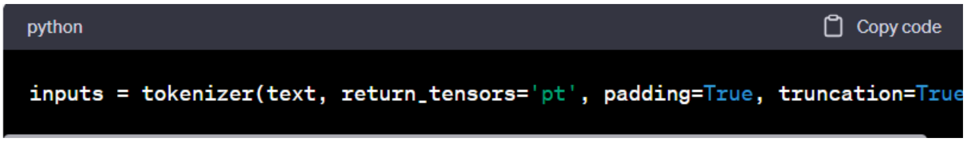
2. **Load and Preprocess Data**:
    - Gather a labeled dataset of spam and non-spam messages.
    - Preprocess the data: remove special characters, convert to lowercase, handle numbers, and perform other necessary text cleaning steps.

3. **Fine-tuning BERT**:
    - Load the pre-trained BERT model and tokenizer.



```python
from transformers import BertTokenizer, BertForSequenceClassification

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
```

4. **Tokenization and Formatting**:
    - Tokenize your text data using the BERT tokenizer. BERT requires specific formatting of input data, including tokenization and adding special tokens for the start and end of the sequence.



```python
inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True
```

5. **Train-Test Split**:
    - Split your dataset into training.