

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:

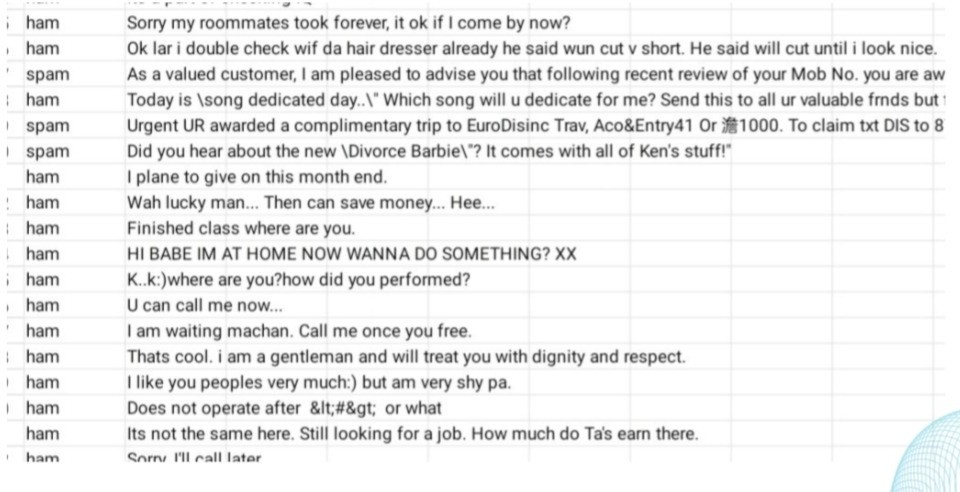
# Data Collection and Preprocessing

1. **Fine-tuning BERT**
2. **Feature Extraction**
3. **Training**
4. **Fine-tuning Parameters**
5. **Evaluation**
6. **Testing**
7. **Deployment**
8. **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.

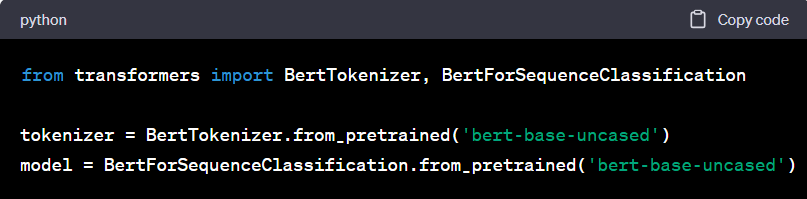


# Fine-tuning BERT:

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.



# Feature Extraction:

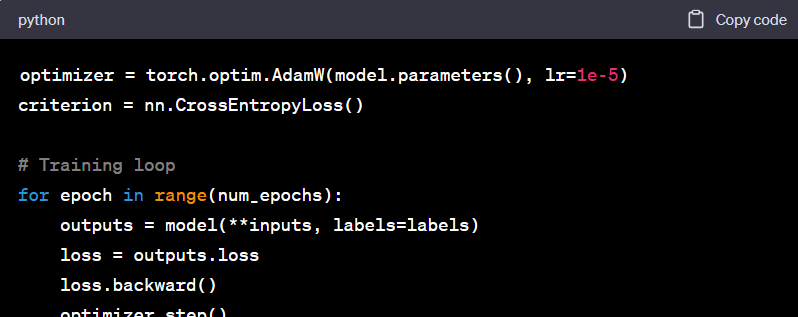
Use the pre-trained BERT model to convert the text into high-dimensional embeddings. These embeddings capture semantic information about the text.

# 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.



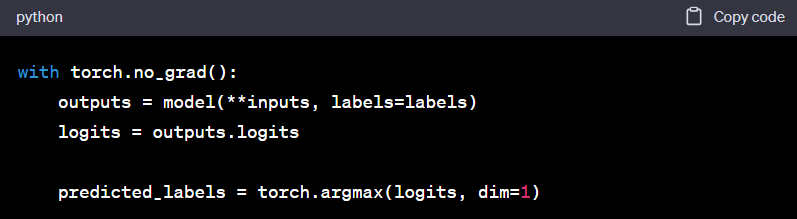


# Fine-tuning Parameters:

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

# 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.



# Testing:

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

# 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.

# Monitoring and Maintenance:

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



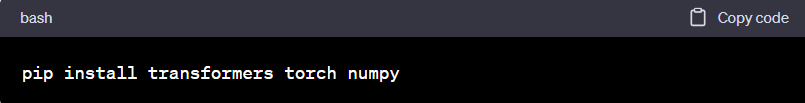
**STEPS To build an AI-powered spam classifier using BERT**

# Setting Up the Environment

* 1. **Load and Preprocess Data**
  2. **Fine-tuning BERT**
  3. **Tokenization and Formatting**
  4. **Train-Test Split**

1. **Setting Up the Environment**:

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



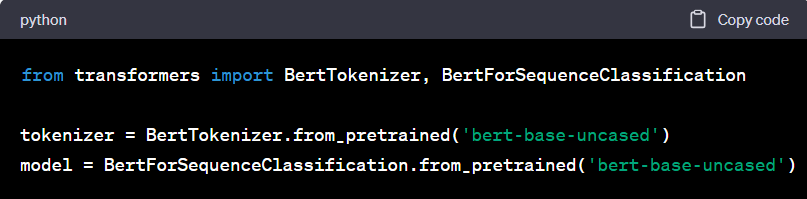
# Load and Preprocess Data:

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.

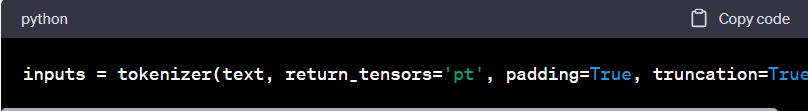
# Fine-tuning BERT:

Load the pre-trained BERT model and tokenizer.



# 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.



# Train-Test Split:

Split your dataset into training.

