Building A Smarter AI Powered Spam Classifier

Phase 4: Development Part II

Student Name: Samyukthan Ananth

College Name: Madras Institute of Technology

Project Title: Implementing An Intelligent Spam Classifier Powered By BERT.

Problem Statement: The goal of this project is to develop an AI-powered spam classifier that can accurately differentiate between spam and non-spam (ham) messages. The problem statement is to build a natural language processing (NLP) model capable of classifying text messages as either spam or ham using the BERT (Bidirectional Encoder Representations from Transformers) model, a state-of-the-art transformer-based deep learning architecture.

Solution: The solution involves using a labelled dataset containing both spam and ham messages for training and evaluation. This is the "spam.csv" file imported from the **Kaggle Spam Database**.

The steps to address this problem include:

1. **Data Preprocessing:** The initial step involves data preprocessing, including the loading of the dataset and transforming the labels into numeric format for machine learning. The text data is also cleaned and prepared for further processing.

2. **Model Selection:** Instead of traditional text classification models, we aim to use a BERT-based model, which has demonstrated exceptional performance in NLP tasks. We select a pre-trained BERT model and tokenizer for our text classification task.

3. **Fine-tuning BERT:** The selected BERT model is fine-tuned on the training data to adapt it to the specific spam classification task. Fine-tuning involves training the model on the labelled dataset, adjusting model parameters to optimize its performance.

4. **Prediction:** After fine-tuning, the BERT model is capable of making predictions on new text data. The model scores each text message, and based on these scores, a threshold is used to classify the message as either spam or ham. The higher the score, the more likely the message is classified as ham.

5. **User Interaction:** The final application allows users to input text messages for classification. Users receive real-time predictions, helping them identify spam messages more effectively.

To achieve this, we require the following tools:

1. Python 3.X with pip
2. pandas
3. sklearn (scikit-learn)
4. transformers - We import the BertTokenizer and BertForSequenceClassification objects.
5. Torch

To implement this project:

1. Import the "spam.csv" file previously installed from the Kaggle Spam Database.
2. Format the csv database by:
    a. Renaming "v1" – "Analysis" and "v2" – "Text".
    b. Removing the three "Unnamed" columns.
    c. Rename "spam" – 0 and "ham" – 1 (binary int value).

    Note: These are not necessary but are performed to make the code easy to understand and more efficient.

3. Define a function **predict()** that takes a string as input. This function:
    a. Loads a pre-trained BERT model (bert-base-uncased) and its tokenizer.
    b. Initializes variables x and y for text data and labels.
    c. Splits the data into training and testing sets.

d. Tokenizes the text data using BERT's tokenizer and truncates it to a maximum length of 512 tokens.
e. Converts the labels to PyTorch tensors.
f. Initializes a BERT model for sequence classification.
g. Fine-tunes the BERT model for spam classification using training data and optimizes it.
h. The code includes a training loop for a specified number of epochs (3 in this case).
i. After training, it predicts the label (spam or ham) for the input text using the fine-tuned model.

4. Take string input from user and run it through the predict() function which returns ham or spam.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from transformers import BertTokenizer, BertForSequenceClassification
import torch
from torch.utils.data import TensorDataset, DataLoader

data = pd.read_csv("spam.csv", encoding = "latin-1")

data.rename(columns = {"v1" : "Analysis", "v2" : "Text"}, inplace = True)

ctd = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]
data.drop(columns = ctd, inplace = True)

data["Analysis"] = data["Analysis"].map({"spam" : 0, "ham" : 1})

def predict(content: str):
    model_name = "bert-base-uncased"
    tokenizer = BertTokenizer.from_pretrained(model_name)
    model = BertForSequenceClassification.from_pretrained(model_name)

    x = data["Text"]
    y = data["Analysis"]
    xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2,
random_state = 3)

    xtrain_tokens = tokenizer(xtrain.tolist(), padding = True, truncation =
True, return_tensors = "pt", max_length = 512)
```

```
    xtest_tokens = tokenizer(xtest.tolist(), padding = True, truncation =
True, return_tensors = "pt", max_length = 512)

    ytrain = torch.tensor(ytrain.tolist())
    ytest = torch.tensor(ytest.tolist())

    batch_size = 32

    train_dataset = TensorDataset(xtrain_tokens.input_ids,
xtrain_tokens.attention_mask, ytrain)
    train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)

    optimizer = torch.optim.AdamW(model.parameters(), lr = 1e-5)

    for epoch in range(3):
        model.train()
        for input_ids, attention_mask, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(input_ids, attention_mask = attention_mask, labels
= labels)
            loss = outputs.loss
            loss.backward()
            optimizer.step()

    model.eval()
    with torch.no_grad():
        input_ids = tokenizer(content, padding = True, truncation = True,
return_tensors = "pt", max_length = 512).input_ids
        attention_mask = tokenizer(content, padding = True, truncation = True,
return_tensors = "pt", max_length = 512).attention_mask
        prediction = model(input_ids, attention_mask = attention_mask).logits

    return ("Spam" if prediction[0][0] > prediction[0][1] else "Ham")

while True:
    n = input("Enter The Text To Analyze ['Exit' To End]: ")
    if n.lower() == "exit":
        break
    else:
        result = predict(n)
        print(result)
```

Output:

```
C:\Users\shyaa\Desktop\learning c++>python AI_Phase4.py
Enter The Text To Analyze ['Exit' To End]: Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to re
ceive entry tickets now!!
Spam Mail
Enter The Text To Analyze ['Exit' To End]: How are you doing today?
Ham Mail
Enter The Text To Analyze ['Exit' To End]: Congrats! 1 year special cinema pass for 2 is yours. call 09061209465 now! C Suprman V, Ma
trix3, StarWars3, etc all 4 FREE! bx420-ip4-5we. 150pm. Dont miss out!
Spam Mail
Enter The Text To Analyze ['Exit' To End]: Congrats on your new job!
Ham Mail
Enter The Text To Analyze ['Exit' To End]: 07732584351 - Rodger Burns - MSG = We tried to call you re your reply to our sms for a fre
e nokia mobile + free camcorder. Please call now 08000930705 for delivery tomorrow!!!
Spam Mail
Enter The Text To Analyze ['Exit' To End]: Its pretty funny lol
Ham Mail
Enter The Text To Analyze ['Exit' To End]: Congratulations! You've won a free iPhone. Claim your prize now!
Spam Mail
Enter The Text To Analyze ['Exit' To End]: Ok I think its enough
Ham Mail
```

Conclusion:

This code reads a CSV file containing text data and labels, fine-tunes a pre-trained BERT model for spam/ham classification, and allows users to input text for real-time classification using the fine-tuned model.