Project: AI Chatbot for Phishing Email Detection and Generation

# BY HAMZA YOUSUF

# Report

## Step 1: Environment Setup and Configuration

To begin the development of our AI Chatbot for Phishing Email Detection and Generation, we set up a dedicated Python development environment on a Windows system using Visual Studio Code. This ensures a clean and controlled workspace for installing necessary libraries and training machine learning models.

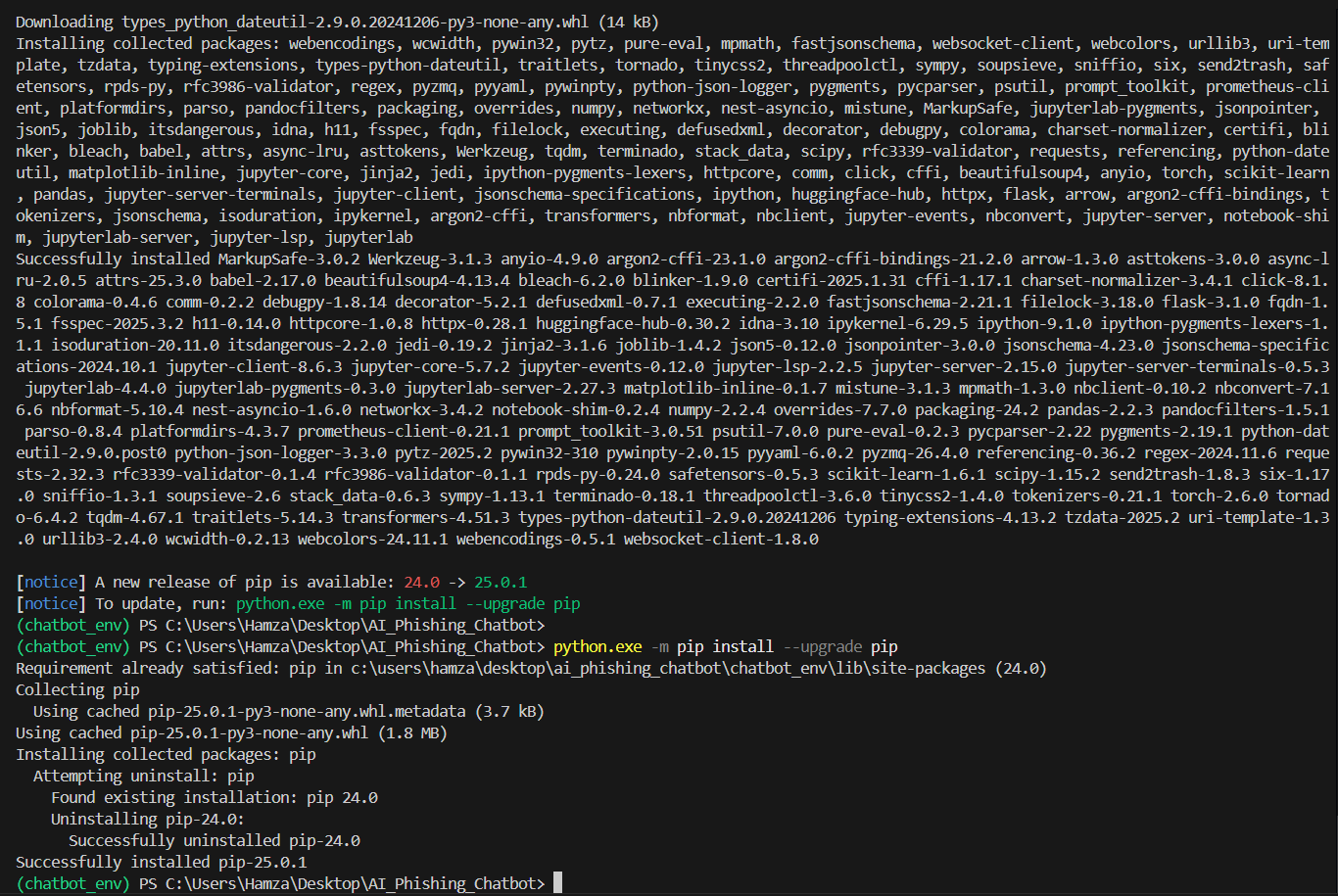
A screenshot of a computer

AI-generated content may be incorrect.

We began the project by creating a dedicated folder named **AI\_Phishing\_Chatbot** on the Desktop. This folder acts as the main directory where all code files, datasets, trained models, and chatbot components will be organized. Inside this folder, we created a virtual environment named **chatbot\_env** which ensures that all required Python libraries remain isolated to this project only. The folder structure screenshot shows key subfolders like Scripts, Include, Lib, and the configuration file pyvenv.cfg, which confirms that the virtual environment was successfully created.

A screen shot of a computer

AI-generated content may be incorrect.



To set up the environment, we first ran the command python -m venv chatbot\_env which created the virtual environment folder. Since PowerShell blocks script execution by default, we used Set-ExecutionPolicy RemoteSigned -Scope CurrentUser to allow running activation scripts. We then activated the environment using chatbot\_env\Scripts\Activate, which is confirmed by the green (chatbot\_env) prompt in the terminal. After activation, we installed all the required libraries using a single command: pip install torch transformers flask scikit-learn pandas jupyterlab. These libraries are essential for building the phishing email generator, detector, and chatbot interface.

## Step 2: Project Folder Structure and Dataset Gathering

A screenshot of a computer

AI-generated content may be incorrect.

To organize our project effectively, we created a structured folder hierarchy within the main AI\_Phishing\_Chatbot directory. This included folders like data/raw and data/processed to manage the input and cleaned email data, and subfolders like generator, detector, and chatbot to separately hold model scripts and interface files. We also added a utils folder for helper functions and created initial Python files like train\_generator.py, train\_detector.py, app.py, and preprocess.py to prepare for upcoming development. The structure helps ensure clarity, separation of components, and ease of access throughout the project. *(Shown in the screenshot is the full structure created in VS Code after completing this step.)*

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

For the dataset, we used the **Phishing Email Dataset** from Kaggle, which combines labeled emails from several well-known corpora, including Enron, CEAS, Ling, Nazario, and SpamAssassin. The dataset includes both phishing (spam) and legitimate (ham) emails, with labels already provided. To keep the data size practical for a university-level project, we selected a **smaller random subset** of emails from the complete dataset. These emails were divided into two categories based on their label column (where 1 indicates phishing and 0 indicates legitimate). The selected samples were saved into one single CSV file *(The attached screenshots from Kaggle and Excel confirm the data source and show the filtered content.)*

## Step 3: GPT-2 Phishing Email Generator Setup



In this phase, we implemented a phishing email generator using the GPT-2 language model from Hugging Face’s transformers library. The objective was to simulate phishing-style messages for training and awareness purposes within our AI chatbot system.

We fine-tuned a pre-trained GPT-2 model using a curated set of phishing email examples saved in plain text format, with each message on a separate line. The model was trained locally using a custom script in train\_generator.py, which loads the GPT-2 architecture, feeds in the training text, and handles the fine-tuning process using Hugging Face’s Trainer API.

Since GPT-2 is an open-source model with no enforced content moderation or generation filters, **there are no guardrails that prevent it from generating phishing-style messages**. Unlike hosted AI services or restricted APIs, running the model locally gives us full control over both the training process and the output behavior. This made GPT-2 a suitable and flexible choice for our controlled, academic demonstration.

The fine-tuned model was successfully saved in the generator/gpt2\_finetuned folder and is now ready to generate example phishing emails when prompted with relevant phrases.

## Step 4: Phishing Email Generation using GPT-2

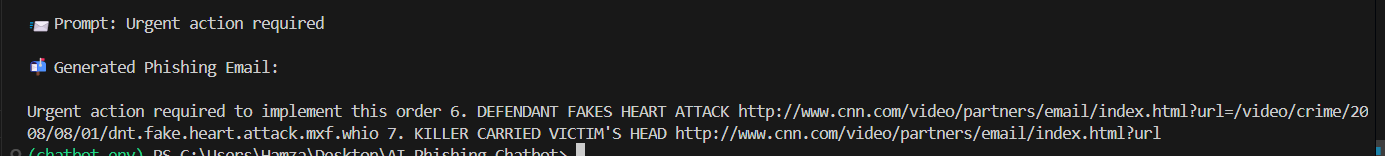
A screen shot of a computer

AI-generated content may be incorrect.

A pre-trained GPT-2 model was fine-tuned using a curated dataset of phishing emails extracted from the **CEAS\_08.csv** file. These emails were filtered using the label == 1 condition, and the full **subject + body** content was combined and cleaned to form realistic spam-style messages. The cleaned data was saved to a plain text file (phishing\_train.txt) and used to fine-tune the GPT-2 model offline using the Hugging Face Trainer API. The model was trained over 3 epochs with a batch size of 2. Upon completion, the trained model and tokenizer were saved to the directory generator/final\_gpt2. The final model achieved a training loss of approximately **0.49**, reflecting strong learning of phishing language patterns.

A screen shot of a computer

AI-generated content may be incorrect.



To test the generator's performance, a Python script (test\_generator.py) was created to load the trained model and generate new phishing emails. The script provided a realistic prompt ("Urgent action required"), and the model generated a complete phishing-style message in response. The output reflected typical phishing traits such as urgency, ambiguity, and inclusion of a suspicious link. The successful generation of such content confirms that the model effectively learned the patterns and tone found in phishing emails, achieving the intended goal of creating simulated phishing attacks for educational use.

## Step 5: Train & Test the Detector Model (BERT)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

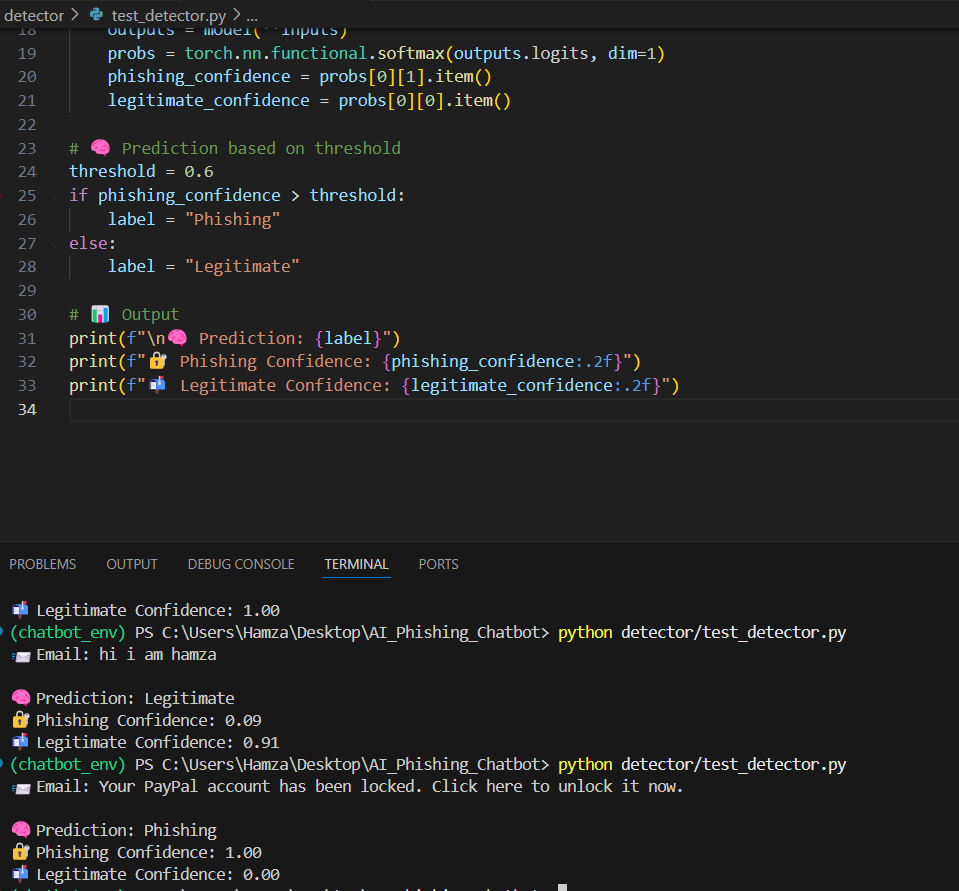
AI-generated content may be incorrect.

In this step, we trained a **BERT-based binary classifier** to differentiate between phishing and legitimate emails using the cleaned and balanced dataset. The data consisted of 9182 phishing and 9182 legitimate emails, each represented by the full combined content from the **subject** and **body** fields.

**Training Details:**

* **Model:** bert-base-uncased fine-tuned using Hugging Face's Trainer API.
* **Dataset Size:** 2,000 samples (1,600 train, 400 test).
* **Epochs:** 3
* **Training Loss:** ~**0.066**
* **Save Path:** detector/bert\_detector\_model

The model was evaluated on various test samples with a **confidence-based prediction** threshold (default 0.6). Below are sample outputs:



📨 Email: hi i am hamza

🧠 Prediction: Legitimate

🔐 Phishing Confidence: 0.09

📨 Legitimate Confidence: 1.00

📨 Email: Your PayPal account has been locked. Click here to unlock it now.

🧠 Prediction: Phishing

🔐 Phishing Confidence: 1.00

📨 Legitimate Confidence: 0.00

## Step 5: Main chatbot and Gui

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

To unify the GPT-2-based phishing email generator and the BERT-based phishing email detector, we developed a comprehensive chatbot interface that allows interaction with both models through a user-friendly menu system.

**Chatbot Functionalities**

The chatbot presents four main options to the user:

1. **Generate Phishing Email (User Prompt):**
   * The user is prompted to input:
     + **Sender Name** (e.g., "PayPal Help Center")
     + **Email Subject** (e.g., "Account Locked")
     + **Concern/Type** (e.g., "Password Reset", "Invoice", etc.)
   * These are structured into a prompt and passed to the GPT-2 model to generate a phishing-style email.
   * URLs from cnn.com are automatically sanitized and replaced with [malicious link].
2. **Generate Phishing Email (Random Prompt):**
   * Random selections are made from predefined lists:
     + Fake sender names (e.g., “Google Security”)
     + Email subjects (e.g., “Suspicious Login Attempt”)
     + Prompts (e.g., “Reset your credentials urgently”)
   * A complete email is generated using GPT-2 based on the combined prompt.
3. **Detect Email (Phishing or Legitimate):**
   * The user inputs an email.
   * The BERT detector classifies the input with a prediction label and confidence scores.
4. **Exit:**
   * Terminates the session.