

Data Science Report

Navyatha Chunduri
IIT Hyderabad

September 17, 2025

1 Overview

The AI Email Agent is a Python-based application designed to automate email processing. It connects to a user's Gmail account, fetches unread emails, and uses a two-stage AI pipeline to classify them for importance and summarize the critical ones. The goal is to reduce manual email triage and provide the user with immediate, concise summaries of what matters most.

2 Fine-Tuning Setup

2.1 Data

2.1.1 Data Collection

The initial dataset was collected programmatically from a personal Gmail account using the official Google Gmail API. A Python script was developed to authenticate with the service via OAuth 2.0 and fetch the content of the 500 most recent emails. The script was designed to handle both plain text and HTML formats to ensure the full body content was extracted. The raw data for each email was saved into a JSONL file. The collection script is shown below.

```
1 import base64
2 from bs4 import BeautifulSoup
3 from tqdm import tqdm
4 import json
5
6 # Function to list message IDs
7 def list_message_ids(service, max_results=500, user_id="me"):
8     """Return a list of Gmail message IDs"""
9     try:
10         response = service.users().messages().list(userId=user_id, maxResults=
max_results).execute()
11         return response.get("messages", [])
12     except Exception as e:
13         print("Error listing messages:", e)
14         return []
15
16 # Function to fetch full message
17 def get_message(service, msg_id, user_id="me"):
18     try:
19         msg = service.users().messages().get(userId=user_id, id=msg_id, format=
"full").execute()
20         return msg
21     except Exception as e:
22         print("Error fetching message:", e)
```

```

23         return None
24
25 # Function to parse message
26 def parse_message(msg):
27     """Extract subject, sender, date, body"""
28     headers = {h['name']: h['value'] for h in msg['payload']['headers']}
29     subject = headers.get('Subject', '')
30     sender = headers.get('From', '')
31     date = headers.get('Date', '')
32
33     body = ""
34     # (Body parsing logic as provided by user) ...
35     return {"subject": subject, "from": sender, "date": date, "body": body}

```

Listing 1: Python script for Gmail data collection.

2.1.2 Data Preprocessing and Labeling

After collection, the raw emails were processed to clean the text and assign a binary label ('True' for important, 'False' for not important). A rule-based approach was used for initial labeling, where emails were scanned for keywords related to assignments, exams, interviews, and registrations. HTML content in the email bodies was parsed and converted to plain text using the 'BeautifulSoup' library. This script produced a clean, labeled dataset ready for model training.

2.1.3 Data Splitting

The final, preprocessed dataset of 500 emails was split into three distinct sets to ensure a robust training and evaluation process:

- **Training Set:** 80% (400 emails) - Used to fine-tune the model.
- **Validation Set:** 10% (50 emails) - Used to monitor the model's performance during training and prevent overfitting.
- **Testing Set:** 10% (50 emails) - Held back completely and used only for the final evaluation of the trained model's performance.

2.2 Method

The fine-tuning process was conducted using a combination of a well-established pre-trained model and a robust set of open-source libraries.

2.2.1 Model Choice

The model chosen for the classification task was **bert-base-uncased**. This model was selected for several key reasons:

- **Dataset Constraints:** The training dataset was limited to 500 emails. This small, specialized dataset is insufficient for fine-tuning a Large Language Model (LLM) effectively, as LLMs require much larger datasets to learn and avoid overfitting.
- **BERT's Suitability for Smaller Datasets:** Models like BERT are pre-trained on vast amounts of general text and can be effectively fine-tuned on smaller, domain-specific datasets. For a dataset in this range, BERT-variants are an excellent choice as they can learn the specific patterns, keywords, and context that define an "important" email for the user.

- **Efficiency:** As a lightweight, local model, it makes classification extremely fast and cost-effective, avoiding the need for an API call for every incoming email.

2.2.2 Tools and Libraries

The project was implemented in Python, leveraging a suite of powerful libraries for machine learning and data handling. The core components of the technical stack are listed below.

```
1 # Core libraries for data manipulation and deep learning
2 import pandas as pd
3 import torch
4 import numpy as np
5
6 # Hugging Face ecosystem for transformers and datasets
7 from transformers import (
8     AutoTokenizer,
9     AutoModelForSequenceClassification,
10    Trainer,
11    TrainingArguments
12 )
13 from datasets import Dataset
14
15 # Scikit-learn for data splitting and evaluation
16 from sklearn.model_selection import train_test_split
17 import evaluate
```

Listing 2: Key Python libraries used for model training.

2.3 Results

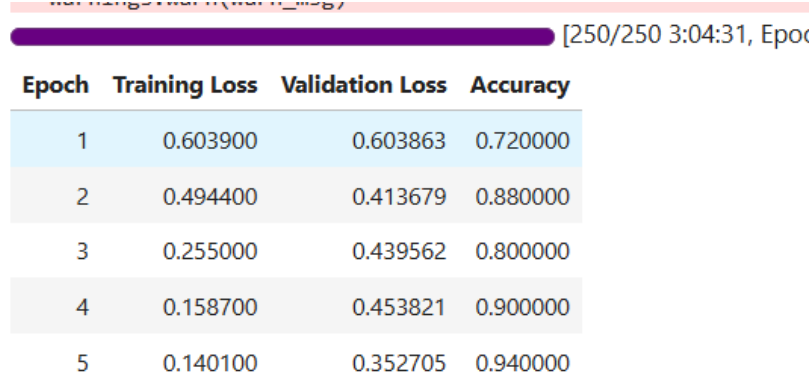
The fine-tuning process concluded successfully. The training job completed without errors, and the resulting model artifacts were saved to a local directory named **Classifier Model**. This folder contains all the necessary components—including the model weights, tokenizer configuration, and model configuration files—required to load and run the trained classifier for inference in the final AI agent application..

3 Evaluation Methodology and Outcomes

3.1 Classifier Evaluation (Quantitative)

The performance of the fine-tuned classifier was evaluated quantitatively using the held-back testing set (50 emails). This test set was not used during training, ensuring an unbiased assessment of the model’s ability to generalize to new data.

The primary metric was accuracy. The model’s validation accuracy progressed steadily during training, reaching 94% by the final epoch as shown in Figure 1.



Epoch	Training Loss	Validation Loss	Accuracy
1	0.603900	0.603863	0.720000
2	0.494400	0.413679	0.880000
3	0.255000	0.439562	0.800000
4	0.158700	0.453821	0.900000
5	0.140100	0.352705	0.940000

Figure 1: Model validation accuracy and loss over 5 training epochs.

After training, predictions were generated for the test set. The model achieved a final test accuracy of **90%** (0.9), demonstrating a strong ability to correctly identify important emails.

3.2 Summarizer Evaluation (Qualitative)

The summarization module, which uses the Gemini API with a few-shot prompting technique, was evaluated qualitatively. As this is a generative task, human judgment is the most effective measure of performance.

3.2.1 Methodology

The methodology involved feeding the summarizer a variety of emails that the classifier had correctly identified as important. The output was then assessed based on the following criteria:

- **Clarity:** Is the summary easy to understand?
- **Conciseness:** Does it capture the main point without unnecessary detail?
- **Relevance:** Does it accurately reflect the core message and required action of the original email?

3.2.2 Outcomes

The outcomes were consistently positive, with the agent producing relevant and actionable summaries that successfully captured the essence of the original emails. Below are two representative examples of the summarizer’s performance.

Example 1: Internship

- **Original Email Body:** "Reminder to register The registration deadline for the Senior Product Analyst Intern profile of Media.net is 9/2/2025, 4:00:00 PM Please log in to the Student Portal complete the registration at the earliest if you are eligible. Student Portal If you are unable to click the above button, copy paste the below URL into your address bar <https://ocs.iith.ac.in/portal> Regards Office of Career Services IIT Hyderabad Email: support.ocs@iith.ac.in Phone: 040 2301 6098 Disclaimer:- This footer text is to convey that this email is sent by one of the users of IITH. So, do not mark it as SPAM."
- **Generated Summary:** Comapany: Media.net, Role: Senior Product Analyst Intern, Deadline: 4:00 PM, 02/09/2025"

Example 2: Assignment

- **Original Email Body:** Notification settings (CS3530 AUG2025) Computer Networks
New assignment Assignment-1 Please find the assignment instructions in the Google doc
below: [https://docs.google.com/document/d/1FvSKhF5sNTDVUkds97vVoZVFRjDj4a8pCkO2jpabLU/t.0DueSep15SeedetailsPostedon9:53AM,Sep2\(GMT+05:30\)byKShivKumarGoogleLLC1600Amph](https://docs.google.com/document/d/1FvSKhF5sNTDVUkds97vVoZVFRjDj4a8pCkO2jpabLU/t.0DueSep15SeedetailsPostedon9:53AM,Sep2(GMT+05:30)byKShivKumarGoogleLLC1600Amph)
- **Generated Summary:** Course: CS3530 (Computer Networks), Task: Assignment-1,
Due Date: 15/09/2025.