Problem Statement: Gen AI-Based Email Classification
Team: TechByteCoders

---

1. Introduction

Manually triaging service requests from emails is time-consuming and inefficiency, human error in routing the emails, especially with high volumes. This document presents an automated email classification and data extraction system using the **Gemma 2B LLM** and **FastAPI** to streamline the process, improve accuracy, and reduce turnaround time.

---

2. Problem Definition

Manual email triage involves:

- Reading and interpreting email content and attachments.
- Identifying intent and classifying request types.
- Extracting key attributes for service request..
- Assigning requests to appropriate teams based on skills.

Challenges:

- **Time-Consuming** – Requires dedicated effort from human triage teams.
- **Error-Prone** – Susceptible to misclassification and inconsistencies.
- **Scalability Issues** – Increased workload leads to slower response times.

---

3. Solution Overview

Our solution leverages the **Gemma 2B LLM** to automate email classification and data extraction. The **application** performs the following tasks:

- FastAPI to expose the API.
- Ingests emails from a specified folder.
- Python packages for extraction.
- Uses **Gemma 2B LLM** (via **Ollama**) for classification
- Returns classification results in **JSON format**.

---

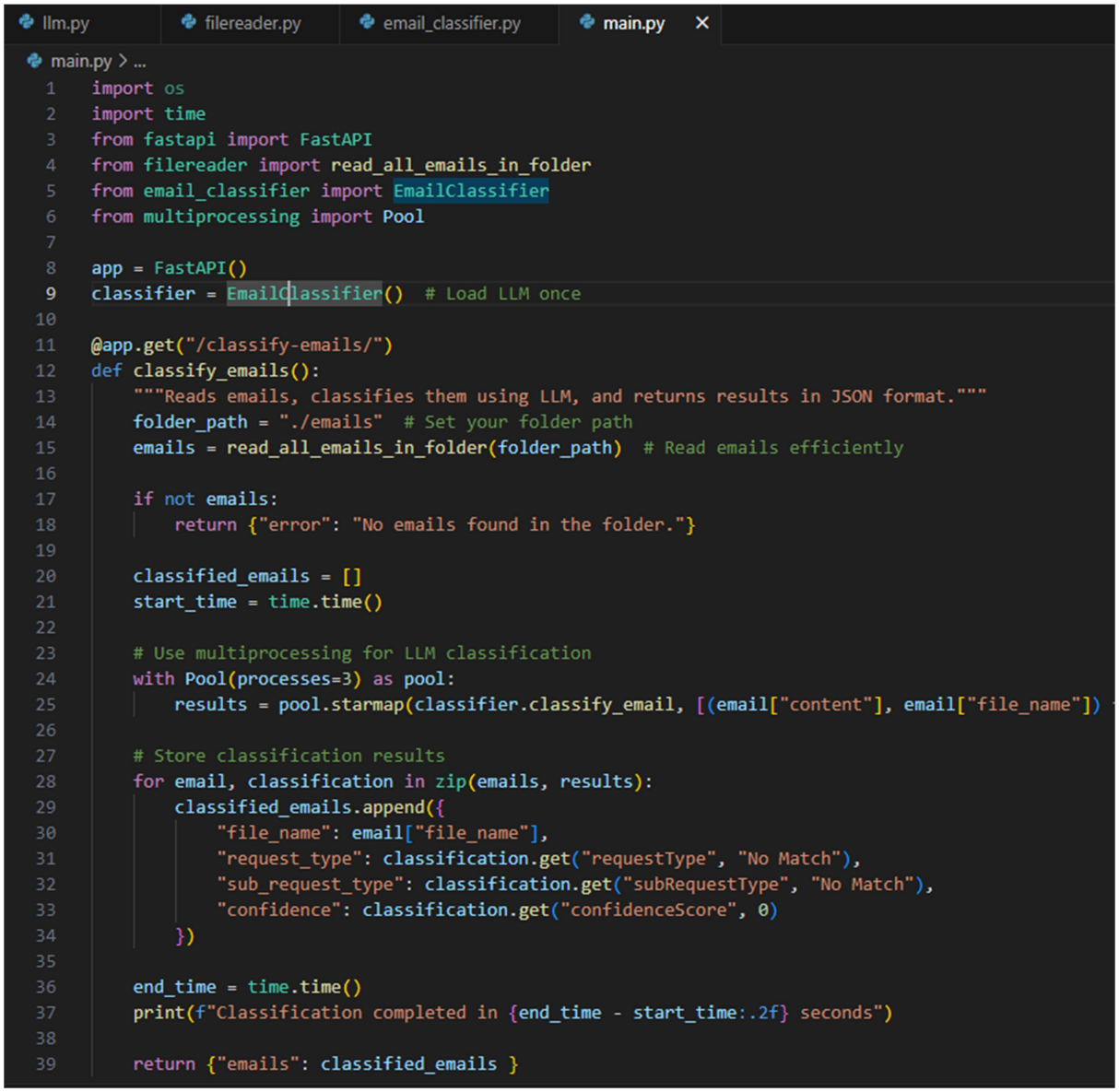
4. Implementation Details

4.1. Code Structure

The solution comprises the following Python files:

- **main.py** – FastAPI application exposing the `/classify-emails/` endpoint.
- **file_reader.py** – Reads emails from the specified folder.
- **email_classifier.py** – Contains the `EmailClassifier` class for LLM-based classification.

4.2. Code Snippet (main.py)



```python
import os
import time
from fastapi import FastAPI
from filereader import read_all_emails_in_folder
from email_classifier import EmailClassifier
from multiprocessing import Pool

app = FastAPI()
classifier = EmailClassifier()  # Load LLM once

@app.get("/classify-emails/")
def classify_emails():
    """Reads emails, classifies them using LLM, and returns results in JSON format."""
    folder_path = "./emails"  # Set your folder path
    emails = read_all_emails_in_folder(folder_path)  # Read emails efficiently

    if not emails:
        return {"error": "No emails found in the folder."}

    classified_emails = []
    start_time = time.time()

    # Use multiprocessing for LLM classification
    with Pool(processes=3) as pool:
        results = pool.starmap(classifier.classify_email, [(email["content"], email["file_name"])

    # Store classification results
    for email, classification in zip(emails, results):
        classified_emails.append({
            "file_name": email["file_name"],
            "request_type": classification.get("requestType", "No Match"),
            "sub_request_type": classification.get("subRequestType", "No Match"),
            "confidence": classification.get("confidenceScore", 0)
        })

    end_time = time.time()
    print(f"Classification completed in {end_time - start_time:.2f} seconds")

    return {"emails": classified_emails }
```

4.3. Execution and Results

The FastAPI application is run using **Uvicorn**:

```
INFO:     Application startup complete.
WARNING:  StatReload detected changes in 'main.py'. Reloading...
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [6624]
INFO:     Started server process [8132]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
Reading completed in 2.01 seconds
```

```
INFO: Uvicorn running on http://127.0.0.1:5000 (Press CTRL+C to quit)
INFO: Application startup complete.
Classification completed in 762.08 seconds
```

Performance Insights:

- **Email reading:** 2.01 seconds.
- **LLM classification:** 762.08 seconds (affected by system performance and memory optimization).

Response Json Report:

response_1743008223602.json

Request URL

```
http://127.0.0.1:8000/classify-emails/
```

Server response

Code	Details
200	Response body


```
{
  "emails": [
    {
      "file_name": "adding attachment test.eml",
      "classification": [
        {
          "requestType": "Adjustment",
          "subRequestTypes": [
            {
              "subRequestType": "Correction",
              "keywords": [
                "correction",
                "adjustment",
                "rectify",
                "modify",
                "revise"
              ]
            }
          ],
          "confidenceScore": 85,
          "filename": "adding attachment test.eml"
        }
      ]
    },
    {
      "file_name": "closure.txt",
      "classification": [
```


Download

```
      {
        "requestType": "Closing Notice",
        "subRequestTypes": [
          {
            "subRequestType": "Reallocation Fees",
            "keywords": [
              "reallocation fee"
            ],
            "confidenceScore": 90
          },
          {
            "subRequestType": "Loan Closure",
            "keywords": [
              "loan closure"
            ],
            "confidenceScore": 85
          }
        ],
        "filename": "closure.txt"
      }
    ]
  },
  {
    "file_name": "multiple request email chain.docx",
    "classification": [
      {
        "requestType": "Adjustment",
```


Response body


```
  {
    "file_name": "multiple request email chain.docx",
    "classification": [
      {
        "requestType": "Adjustment",
        "subRequestTypes": [
          {
            "subRequestType": "Correction",
            "keywords": [
              "correction",
              "adjustment",
              "rectify",
              "modify",
              "revise"
            ]
          }
        ],
        "filename": "multiple request email chain.docx"
      }
    ]
  },
  {
    "file_name": "Request money out.msg",
    "classification": [
      {
        "requestType": "Error",
        "subRequestType": "No Match",
```


Download

Example for Duplicate


```
{
  "emails": [
    {
      "file_name": "adding attachment test.eml",
      "classification": [
        {
          "requestType": "Loan Closure",
          "subRequestTypes": [
            {
              "subRequestType": "Reallocation Fees",
              "keywords": [
                "reallocation fee"
              ]
            },
            {
              "subRequestType": "Loan Closure",
              "keywords": [
                "loan closure"
              ]
            }
          ],
          "confidenceScore": 95,
          "filename": "adding attachment test.eml"
        }
      ],
      "duplicate": false
```

Response body


```
    "classification": [
      {
        "requestType": "Money Movement In-Bound",
        "subRequestType": "Deposit",
        "confidenceScore": 95,
        "filename": "Request to understand on money transfer - duplicate.eml"
      }
    ]
  },
  {
    "file_name": "Request to understand on money transfer.eml",
    "classification": [
      {
        "requestType": "Money Movement In-Bound",
        "subRequestTypes": [
          {
            "subRequestType": "Deposit",
            "keywords": [
              "fund deposit"
            ]
          }
        ],
        "confidenceScore": 85,
        "filename": "Request to understand on money transfer.eml"
      }
    ]
  }
```


---

5. Interactive Demonstrations

- **API Endpoint Testing:** FastAPI automatically generates Swagger UI (`http://127.0.0.1:5000/docs`).
- **User Interface:** A Streamlit or Gradio interface can be integrated for real-time classification.

Request Types & Subcategories

Request Type	Sub-Request Type
Closing Notice	Loan Closure
Adjustment	Correction
Money Movement Out-Bound	Deposit
Money Movement In-Bound	Deposit

File Formats Covered

File Format	Examples
.txt	closure.txt
.docx	multiple request email chain.docx
.eml	adding attachment test.eml, Request to understand on money transfer - duplicate.eml
.msg	Request money out.msg

---

6. Results and Evaluation

6.1. Accuracy of LLM Classification

Confidence Range	Reliability	Example Cases
100% Confidence	Absolute Certainty	-
95% Confidence	High Reliability	Request to understand on money transfer - duplicate.eml
90% Confidence	Moderate Reliability	adding attachment test.eml
85%	Lower Reliability	closure.txt
20%	Very Low Reliability/human intervention required	Request money out.msg

6.2. Time Savings Compared to Manual Triage

Process	Manual Triage	Automated (LLM-Based)
Email Review	3-5 min/email	~2.01 sec total
Classification	5-15 min/email	762.08 sec batch
Total Per Email	~8-20 min	~14.77 sec

---

7. Future Enhancements

1. **Integration with Service Request Platforms**
   - ServiceNow, JIRA, Salesforce for automated ticket creation.
2. **Skill-Based Routing**
   - Extract key attributes for intelligent request assignment.
3. **Improved LLM Prompt Engineering**
   - Refine LLM prompts for better accuracy and multi-request handling.

---

8. Conclusion

This solution demonstrates the potential of **LLMs in automating email triage**, improving efficiency and accuracy. Future enhancements will further streamline operations, optimize classification, and reduce manual effort.
