**Architecture Document: Email Classification System**

1. **Overview** The purpose of this document is to define the architecture of an Email Classification System that reads emails and classifies them into request and sub-request types. The system is implemented using Python in the PyCharm development environment, leveraging the Llama-3.1-8B-Instruct model for natural language processing (NLP).

**The model is also trained to pull the required fields from the email which can later be used to create the Service Request in another system like Service Now.**

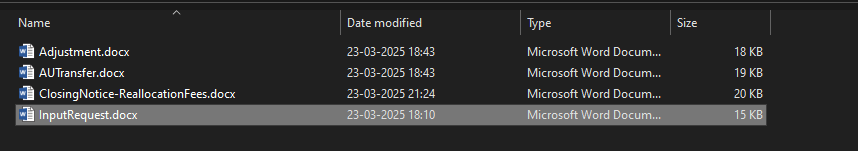
**2. Use Case**

* Read incoming emails from a specified email server. For the current implement we are putting the emails in a folder to train the model.
* Process the email content and classify it into predefined request and sub-request types.
* Retrieve the values as a json object for further processing.

**3. System Components**

**3.1 Email Reader Module: Training the model & Input data**

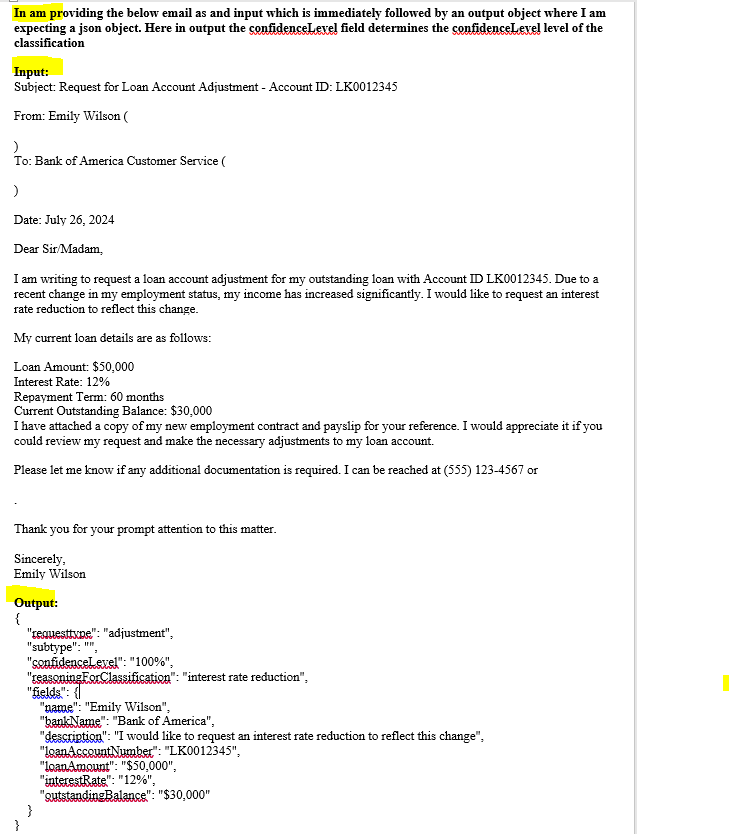
* Fetch the same emails and create mapping outs for them and store per request type basis in the local drive.
* Later the system can be integrated with an Email Server like like IMAP, SMTP, and POP3 for fetching the actual email.
* Create a document with the email which we want to classify and place in the InputRequest.docx.



Hear the first 3 files are the emails which will be used for training the model.

**3.2 LLM Classification Module**

* Uses Llama-3.1-8B-Instruct for natural language processing.
* Pre-processes email text (removal of signatures, greetings, and unnecessary content).
* Classifies emails into a request category and determines its sub-request type based on trained models or predefined rules.
* Prompt is set in the training data to understand the input and the expected output.



**3.3 Data Storage Module**

* As of now we are not using any storage mechanism but later we can use any **vector database** to store the embeddings (vectors) generated while training or fine-tuning the LLM (e.g., PostgreSQL, Redis etc ).
* Maintains metadata for quick clasification.

**3.4 API & Integration Module – Yet to be created**

* Provides API endpoints to retrieve classified emails.
* Enables integration with third-party systems for automated workflows.

**4. Workflow**

1. Fetch predefined emails from the local folder.
2. Extract relevant content and metadata.
3. Pre-process and clean the email text.
4. Use Llama-3.1-8B-Instruct and train it based on the mapped input and outputs set in predefined emails.
5. Then system picks the input email from InputRequest.docx.
6. Post that the solution returns the json output with the classification.

**5. Technology Stack**

* **Programming Language:** Python
* **IDE:** PyCharm
* **Machine Learning Model:** Llama-3.1-8B-Instruct
* **Email Handling:** IMAP/POP3 (via imaplib or poplib)
* **Database:** PostgreSQL/MongoDB/SQLite
* **API Framework:** FastAPI/Flask/Django

Why I choose Llama-3.1-8B-Instruct

Llama-3.1-8B-Instruct is a **large language model (LLM)** from Meta, optimized for instruction-following tasks. It is available on Hugging Face and can be used for various NLP applications such as text generation, summarization, and question-answering.

**Key Features**

* **8 Billion Parameters** – Balanced for performance and efficiency.
* **Instruction-Tuned** – Trained on diverse prompts to improve reasoning and task execution.
* **Supports RAG (Retrieval-Augmented Generation)** – Can be integrated with vector databases for contextual retrieval.
* **Optimized for Inference** – Works efficiently on GPUs (NVIDIA A100, H100) and TPUs.

**6. Security Considerations**

* Secure storage of email credentials.
* Data encryption for sensitive information.
* Rate limiting and authentication for API access.

**7. Conclusion** This architecture ensures an efficient and scalable solution for classifying emails based on request types, leveraging AI-powered classification with Llama-3.1-8B-Instruct. Further improvements can include continuous model fine-tuning and additional automation workflows.