ANNEXES AND TECHNICAL DETAILS

1 JARVIS Architecture

The JARVIS framework was implemented using **LangGraph** and **LangChain**, two powerful libraries for building multi-agent systems and reasoning pipelines . The underlying models are distributed using **Ollama** for execution on device and **GroqCloud** to access high-performance inference.

The architecture consists of three main components:

- **Orchestrator (OR)**:It manages task distribution and coordination among modules. It is powered by **LLaMa 3.2 3B**, distributed locally through Ollama, which ensures fast response time and efficient handling of requests.
- **Subjective Hemisphere (SH)**: Responsible for generating custom synthetic data and fitting the model to individual users. This module also uses **LLaMa 3.2 3B** on Ollama, benefiting from its light but effective contextual adaptation.
- **Objective Hemisphere (OH)**: Dedicated to the consolidation of knowledge acquired through structured processing and tuning. Given its computational complexity, this component is powered by **LLaMa 3.3 70B** via the **GroqCloud API**, which enables large-scale reasoning and deep knowledge integration.

2 Orchestrator Module

The Orchestrator component represents the decision-making core of the JARVIS architecture, acting as the main router for requests between the objective and subjective agents. Its implementation is based on a state graph constructed with `LangGraph`, which features:

- **Graph Construction**: Nodes represent the different agents (Supervisor, Subjective Agent, Objective Agent) and arcs define the possible transitions.
- **Decision Logic**: A sophisticated prompt is used within the supervisor node to classify incoming user requests as either *subjective* or *objective*.
- **Routing Mechanism**: Based on the classification, requests are routed to the appropriate agent for processing.
- **Self-Judging, Self-Reasoning, Self-Reflexion**:The Orchestrator node uses approaches of Self-Judging , Self-Reflexion ,Self-Reasoning to combine, evaluate and refine the responses obtained from the two hemispheres.

This design allows for efficient handling of tasks by dynamically allocating requests to the agent best suited for the job.

3 Subjective Module

The Subjective Hemisphere is responsible for the customization and personalization aspects of the system. Its main functions include:

- **Dreaming Component**: Generates synthetic data by creating multiple variants of recent user texts, effectively expanding the dataset for model fine-tuning.
- **Fine-Tuning Component**: Employs techniques such as LoRA (Low-Rank Adaptation) and Direct Preference Optimization (DPO) to update the model, enabling it to mimic the user's tone and style.

This module ensures that responses are not only factually correct but also reflect the personalized style of the user, providing a natural and engaging interaction.

4 Objective Module

The Objective Hemisphere focuses on interpreting, processing, and responding to dynamic user requests with precision and consistency. Key features include:

- **Integration with External Tools**: Utilizes the capabilities of LangChain and LangGraph to interact with external services (e.g., web search, email management) for task execution.
- **Conversational Memory**: Implements persistent memory via MongoDB, which maintains the history of interactions to support context-aware responses.
- **Dynamic Tool Selection**: Automatically selects the most appropriate tools based on the nature of the user's request, ensuring effective and relevant responses.
- **Sanity check**:The Objective Agent is also involved in the step of sanity check where the final answer to provide the final user must be validated and checked before released.

5 Experimental Evaluation

5.1 Experimental Setup

The JARVIS framework was subjected to a quantitative evaluation focusing on the customisation capacity of the subjective hemisphere and the effectiveness of the synthetic data generation process.

5.1.1 Dataset

The experiment used a subset of the *The Enron Email Dataset* [**?**], configured as follows:

- 5 distinct user profiles
- 50 original emails per profile
- 50 synthetic emails generated per profile
- Split: 80 training (80 emails), 20 test (20 emails)

5.2 Fine-tuning process

Fine-tuning was performed in two distinct phases:

1. First phase: use of 50 original emails
2. Second phase: integration of 30 synthetic emails generated through the subjective hemisphere dreaming routine.

The methodology implemented the *Direct Preference Optimisation* (DPO) to optimise the learning of the model.

5.3 Evaluation Metrics

The evaluation was conducted using *BERTScore* as the main metric to measure the semantic similarity between:

- Emails generated by the model
- Original test set emails (ground truth)

5.4 Results

The main results of the experiment are:

Utente	Precision	Recall	F1-score
1	0.74	0.84	0.80
2	0.77	0.85	0.81
3	0.78	0.85	0.81
4	0.77	0.82	0.80
5	0.77	0.85	0.80

Table 1: Metrics for Each User

5.5 Analysis of Results

Integration of the synthetic data demonstrated:

- Improving the representation of user preferences .
- Maintaining semantic consistency
- Effective generalization to new data

5.6 Technical Conclusions

The experiment validated the effectiveness of the JARVIS architecture, in particular:

- The robustness of the two-step fine-tuning process.
- The utility of synthetic data generation for dataset expansion
- The appropriateness of BERTScore as an evaluation metric for personalization

The overall performance of the system indicates a robust customization capability while maintaining the semantic accuracy of the generated responses.