Minds with a Mission: The Role of Personality in a Multi-Agent System

1 Implementation Details

We implemented our approach by adopting the Meta Llama 3.1-70B Instruct large language model (LLM). The inference process is guided by a temperature setting of 1.2, which introduces a moderate degree of randomness in the model's responses, allowing for more diverse and creative outputs. Additionally, nucleus sampling is set at 0.9, ensuring that the model considers only the most relevant and probable tokens while generating text, striking a balance between diversity and coherence. Finally, in this first set of experiments, we do not focus on obtaining the highest probability answer from the LLM. Instead, we configure the system to generate a single alternative response for each input, ensuring a deterministic output. We ensure reproducibility by setting a fixed seed value for random number generation directly within our implementation files.

Additionally, we leverage the default *chat-template* provided by Llama 3.1 to structure interactions between the system and the user. This template explicitly defines the roles of both the system and the user, ensuring that the model correctly interprets input prompts and maintains consistency in its responses. By adhering to this standardized template, we enhance the model's ability to generate contextually appropriate and structured outputs, improving overall response coherence and alignment with the intended person assigned to the agent.

1.1 Computational Cost Estimation

We conduct our experiments using the CINECA HPC infrastructure. This section provides an estimation of the required GPU hours.

For each of the five datasets, we run 13 description frameworks. Each experiment is executed on 3 H100 GPUs. Typically, processing 500 experimental samples requires approximately 24 hours. Below, we report the

number of samples tested for each dataset and the corresponding estimated computational cost.

- **Strategy QA**: More than 1600 samples, but we consider 1500 for computational cost estimation. Since we process 500 samples per day, the experiments require 3 days. The computational cost is:

$$\textit{GPU hours cost} = 3 \times 13 \times 24 \times 3 = 2808$$

- **Common Sense QA**: Approximately 500 samples. As we process 500 samples per day, the experiments require 1 day. The computational cost is:

$$\textit{GPU hours cost} = 3 \times 13 \times 24 \times 1 = 936$$

- **Last Letter Concat**: 500 samples. Since we process 500 samples per day, the experiments require 1 day. The computational cost is:

$$\textit{GPU hours cost} = 3 \times 13 \times 24 \times 1 = 936$$

- **Social IQa**: More than 1700 samples, but we consider 1500 for computational cost estimation. Since we process 500 samples per day, the experiments require 3 days. The computational cost is:

$$\textit{GPU hours cost} = 3 \times 13 \times 24 \times 3 = 2808$$

- **Social Support**: 897 samples, rounded to 1000 for computational cost estimation. Since we process 500 samples per day, the experiments require 2 days. The computational cost is:

$$\textit{GPU hours cost} = 3 \times 13 \times 24 \times 2 = 1872$$

Thus, the total estimated computational cost is:

$$\textit{Total GPU hours cost} = 2808 + 936 + 936 + 2808 + 1872 = 9360$$

1.2 System Usage

Each parameter in the JSON file plays a specific role in guiding the system's behavior. Below we report a detailed explanation of each parameter that must be included in the JSON configuration file.

- **name**: The name of the experiment.
- **task**: the task instruction to provide to the experts.
- **context**: The broader domain of the task, indicating that it belongs to, e.g., commonsense question answering.
- **description_framework**: The theoretical description framework guiding the generation of the agent description profiles.
- **model_name**: The specific language model used, in this case, Meta Llama 3.1-70B Instruct.
- **output_dir**: The directory where the generated results are stored.
- **input**: The input file containing queries, with each query on a separate line.
- **temperature**: Controls response randomness. We adopt a value of 1.2 to introduce moderate variability, allowing for diverse outputs.
- **nucleus**: Defines the nucleus sampling threshold. We adopt a value of 0.9 ensures that the model considers only the most relevant tokens during generation.
- **alternatives**: The number of alternative outputs generated per query. Here, it is set to 1 to ensure a deterministic response.
- **resume**: A flag indicating whether to resume execution from a previous run.
- **cache_dir**: Specifies the directory for caching model weights and computations. This parameter avoids downloading the same model multiple times.
- **max_experts_number**: The maximum number of expert models used during inference. In our experiments, we fix this parameter to 3.
- **token**: The authentication token required to access the model on the HuggingFace repository.

This configuration setup allows users to customize the system's behavior while ensuring reproducibility across different experimental settings.

To execute our system, we utilize a SLURM job script to manage resource allocation and scheduling efficiently. Below, we provide an example SLURM configuration file:

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=16
#SBATCH --gpus=3
#SBATCH --mem=100GB
#SBATCH --time="23:59:59"
#SBATCH --partition="cluster-partition"
#SBATCH --qos=normal
#SBATCH --job-name="job-name"
#SBATCH --output="std out"
#SBATCH --error="std err"

echo "Job starting"

source /myvenv/bin/activate

python ../PoE_Small/config_file.py
--config-file configuration.json
```

After specifying the SLURM directives, the script activates a virtual environment (‘source /myvenv/bin/activate‘) and executes the system using a Python script with a predefined JSON configuration file. This setup ensures efficient resource utilization and facilitates job execution on a high-performance computing cluster (e.g., CINECA pre-exascale HPC).