**Introduction**

While large language models have garnered significant attention, smaller language models (SLMs) with 1-4 billion parameters have shown impressive capabilities for specific tasks. These compact models offer a balance between performance and efficiency, making them suitable for applications where resource constraints matter. For instance, SLMs could potentially enhance text editors with real-time writing suggestions, running smoothly on standard laptops without requiring powerful cloud servers.

In this task, we'll explore the potential of these SLMs in another practical application: sentiment analysis of movie reviews. We avoid model training in test tasks as it might be time-consuming. This task is designed to assess your skills in creative problem-solving, insightful analysis and prompt engineering. Additionally, we'll evaluate your Python coding skills through the implementation of inference and results visualization.

**Task Objectives**

1. Implement local inference for SLMs using Python
2. Design and refine prompts to achieve maximum accuracy in sentiment analysis
3. Compare the performance and efficiency of two SLMs of different sizes
4. Analyze results and draw meaningful insights
5. Document your research process and findings effectively

**Models and Dataset**

We'll use two quantized small language models from the Qwen2.5 family for this task:

- bartowski/Qwen2.5-1.5B-Instruct-GGUF: A lightweight 1.5B-parameter instruction-tuned model with enhanced capabilities in coding, mathematics, and handling structured data.
- bartowski/Qwen2.5-0.5B-Instruct-GGUF: A model with 500M parameters, making it three times smaller than its 1.5B counterpart, yet still highly capable for many tasks.

We've chosen these two models to explore how different model sizes affect performance in sentiment analysis. The 1.5B model is larger and potentially more capable, while the 0.5B model is more compact. This comparison will help us understand the trade-offs between model size, speed, and accuracy.

For the dataset, we'll use ajaykarthick/imdb-movie-reviews. This dataset contains 40,000 entries, which is too large for a complete local run. We expect you to choose a reasonable subset of the data for your analysis and explain your selection criteria.

**Task Description**

1. Prepare the dataset for use, explaining your selection criteria for the subset you choose.
2. Implement model inference in a Python script
    - It should support GPU and CPU inference. You can use the llama-cpp-python library for efficient CPU inference of quantized GGUF models.
    - Choose optimal inference parameters (e.g., temperature, top_p, top_k) for each model

    - Do not use CLI tools like Ollama for this task
3. Experiment with various prompt engineering techniques with the goal of maximizing accuracy in sentiment analysis.
    - Different prompt structures you tried
    - Whether prompt engineering techniques can help the 500M model approach or match the 1.5B model's performance
    - Observations and insights from failed attempts
    - Any challenges encountered and how you solved them
4. Propose and implement relevant evaluation metrics. Justify your choice of metrics.
5. Analyze the results you've gathered. Create visualizations to compare how the models performed.
6. Write up a comprehensive summary of your findings, including:
    - Your thought process in arriving at your final prompt(s)
    - Comparative analysis of model performance
    - Justification for your choice of inference parameters
    - Interesting patterns or insights discovered during experimentation
    - Potential real-world applications and limitations of your approach
    - Any performance bottlenecks encountered and your strategies to address them
7. Publish your code, research outcomes, and a README.md with instructions for running your solution locally to a GitHub repository. Please ensure the repository is publicly accessible **but avoid any references to JetBrains**. Share the repository link with us.

**What We're Looking For**

- Quality, clarity, and reproducibility of your Python code
- Thoughtful approach to choosing and justifying evaluation metrics and inference parameters
- Clear documentation of your research process, decision-making, and problem-solving

**Time Expectation and Submission**

This task is designed to be completed in approximately 4-8 hours. While there's no strict time limit, we encourage you to manage your time efficiently and focus on the key aspects of the task.

Remember, we need to be able to run your solution on our local machines, so make sure to include any setup steps or list any dependencies we might need in your `README.md`.

We're more interested in seeing your approach to the problem and the insights you uncover than in achieving perfect results. As a point of interest, we've seen solutions achieve up to 90% accuracy, but don't let that number constrain your thinking. Your unique approach might uncover different insights or trade-offs that we haven't considered.

Feel free to explore and be creative with your solution. If you have any questions during the process, please don't hesitate to reach out.

Good luck!