Name: Jayraj M. Pamnani
NetID: jmp10051

Model Blending of Open-Source Large Language Models using MergeKit

1. Problem Statement

Training state-of-the-art large language models (LLMs) from scratch demands substantial computational resources, large datasets, and significant funding, often exceeding tens of millions of dollars. For students, independent researchers, and small organizations, this level of resource commitment is unattainable.

However, recent advancements in *model blending* techniques demonstrate that combining the weights of existing fine-tuned open-source models can yield powerful hybrid models rivaling proprietary LLMs on public benchmarks. This project aims to explore, implement, and evaluate model blending as a practical and low-cost alternative to full-scale training or fine-tuning of LLMs.

The core problem this project addresses is:

How can multiple fine-tuned LLMs be effectively blended using MergeKit to create a single model that performs competitively across diverse tasks without extensive retraining?

2. Detailed Approach

The project will focus on implementing **model blending using the MergeKit framework**, an open-source Python toolkit that enables merging of pre-trained or fine-tuned transformer models through weight arithmetic and interpolation techniques.

2.1. Research and Preparation

- Review existing literature and research papers on model merging techniques, including *Task Arithmetic*, *Spherical Linear Interpolation (SLERP)*, *TIES*, and *Dare* methods.
- Analyze prior benchmark results from the **Open LLM Leaderboard** to select suitable candidate models for blending (e.g., LLaMA, Mistral, or Falcon models with similar architectures and parameter sizes).

2.2. Implementation Steps

1. **Environment Setup**
   - Install MergeKit on a local or rented GPU environment.

- Configure dependencies in Python (using pip install -e . within the MergeKit repository).

2. **Model Selection and Download**
   - Choose two or more models with compatible architectures and parameter sizes (e.g., LLaMA 7B variants) from Hugging Face.
   - Use Git LFS to download model repositories locally.

3. **Merge Configuration**
   - Create a YAML configuration file specifying:
     - Base model and data type (dtype)
     - Merge method (e.g., Task Arithmetic, SLERP)
     - Model weights and contribution ratios

4. **Model Blending and Evaluation**
   - Execute MergeKit's merge command to create the blended model.
   - Load the resulting model using *Text Generation WebUI* or *Transformers library* to perform inference tests.
   - Evaluate the model's performance using standard reasoning and language benchmarks (ARC, HellaSwag, MMLU, TruthfulQA, and GSM8K).

5. **Optimization and Experimentation**
   - Compare results across different merge methods.
   - Analyze the impact of varying weight ratios and density parameters.
   - Examine model coherence and generalization to unseen prompts.

6. **Benchmark Submission**
   - Upload the final model to Hugging Face and optionally submit it to the Open LLM Leaderboard for public evaluation.

3. System Requirements

Hardware Requirements:

- Local system with at least 16 GB RAM (suitable for merging smaller models up to 7B parameters)
- For larger experiments: GPU instance (e.g., A100 or similar) rented via cloud platforms such as *MK Compute*, *Google Colab Pro*, or *AWS EC2*

Software Requirements:

- Operating System: macOS, Linux, or Windows
- Python 3.10+
- MergeKit toolkit (from GitHub)

- Git LFS for model downloads
- Hugging Face Transformers & Hub libraries
- Text Generation WebUI for model inference

4. Learning Outcomes

Through this project, I aim to gain both conceptual and practical understanding of emerging Deep Learning techniques that optimize the use of pre-trained models. Key learning outcomes include:

- **Understanding Model Weight Manipulation:** Learn how model parameters represent learned knowledge and how arithmetic operations on these weights can influence model behavior.
- **Hands-On Experience with MergeKit:** Develop proficiency in configuring, executing, and troubleshooting large-scale model blending workflows.
- **Evaluation and Benchmarking:** Learn to use standardized LLM benchmarks to quantitatively measure model performance, reasoning, and factual accuracy.
- **Ethical and Technical Awareness:** Understand issues like *data contamination* and *overfitting* in benchmark evaluations and explore how blending can affect model generalization.
- **Applied AI Research Skills:** Strengthen the ability to conduct reproducible experiments, interpret performance metrics, and present results in a research-oriented format.

Conclusion

This project will provide a practical exploration into the rapidly growing field of LLM optimization through model blending. By merging multiple fine-tuned models using MergeKit, I aim to demonstrate how blending can produce cost-effective, high-performance models while deepening my technical understanding of transformer architectures and weight-space manipulation.