Towards an approach combining Knowledge Graphs and Prompt Engineering for Merging Large Language Models

Furel De Consol TEGUIMENE YENDJI
Department of Computer Science
University of Yaounde I
consol.teguimene@facsciences-uy1.cm

Fidel AZANZI JIOMEKONG
Department of Computer Science, University of Yaounde I
TIB – Leibniz Information Centre for Science and Technology, Hannover, Germany
fidel.jiomekong@facsciences-uy1.cm

Sanju TIWARI
Universidad Autonoma de Tamaulipas, Mexico
sanju.tiwari.2007@gmail.com

Carick Appolinaire ATEZONG YMELE
Department of Computer Science
University of Yaounde I
carick.atezong@facsciences-uy1.cm

Janice Anta ZEBAZE
Department of Physics
University of Yaounde I
antazebaze@gmail.com

Abstract

Large Language Models (LLMs) have emerged as transformative tools in areas ranging from education to software development, but their high computing power and energy costs limit their accessibility. Deploying these powerful LLMs in developing countries presents high costs and infrastructural challenges, such as limited access to high-performance computing resources. We address this problem by proposing an approach that relies on lightweight, open-source LLMsm each optimized in sub-tasks of larger complex problems. Our methodology combines knowledge graphs (KGs), prompt engineering and LLM fusion to build a model that works efficiently on global tasks. Our results show an increase in the performance of the fused LLM of 0.32, reaching a final score of **0.63**, which is a significant improvement over the baseline. This work demonstrates that resource-efficient and open-source LLMs, when combined strategically, can provide accessible and effective AI solutions.

1 Introduction

The growing success of LLMs across a wide range of tasks has inspired various applications in fields such as education [1], food information engineering [2], and software development. However, while these models are highly effective, they remain costly and often inaccessible, particularly in developing countries where resources like consistent energy, stable network and advanced infrastructure are limited. Faced with these challenges, we propose a solution that combines smaller, specialized models,

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

each optimized for specific sub-tasks, to achieve similar performance to proprietary LLMs but at lower cost and resource demand. Our approach leverages open-source models that are cost-effective and resource-efficient, thereby reducing barriers to advanced technology access. To optimize model combinations, we employ a knowledge graph (KG) that infers the most relevant pairings based on each model's performance across comparable datasets. This work explores how KGs can guide the selection and fusion of LLMs to address specific tasks within resource constraints, maximizing both efficiency and sustainability in model deployment.

2 Methodology

2.1 Studies of the different approaches used

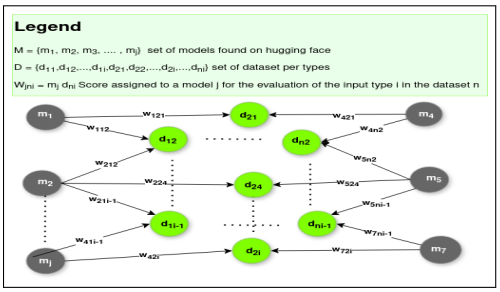
This part presents the different approaches used.

Knowledge graphs: Allowing to structure knowledge by facilitating understanding, this knowledge graph allows us to visualize the interaction between the models and the datasets on which they will be evaluated and highlighting the score. Thanks to this graph, we are able to list a large number of models tested on a set of datasets allowing us to know which model to match with which dataset (see the figure 1a).

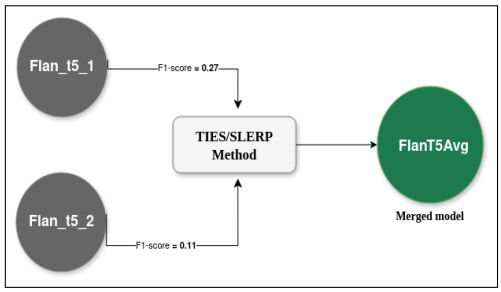
MergeKit framework: Merging models make use of several models such as SLERP, which relies on spherical interpolation to merge models; TIES and Task Arithmetic, which rely on different frameworks on the concept of task vectors for the combination of the LLMs [3]. At the output of such methods, we end up with a single model able to perform suitably on the sub tasks each model is able to accomplish (see the figure 1b).

Prompt Engineering: Its an approach that allows to carefully formulate the request made by the user enabling the model to give much more precise and orderly answers (see the figure 1c).

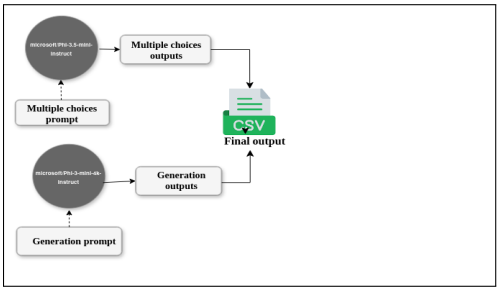
The final approach: The proposed approach combines MergeKit with prompt engineering and KGs. We choose the models to merge based on their scores displayed on a kG such that they have the same architecture and combine them through the task arithmetic method of MergeKit. Once the model is obtained, we improve its answers by injecting an input provided by a carefully constructed prompt. This approach is very efficient because having a merged model, we try to take maximum advantage of its skills through a structured request. 1dfor an overview of the process.



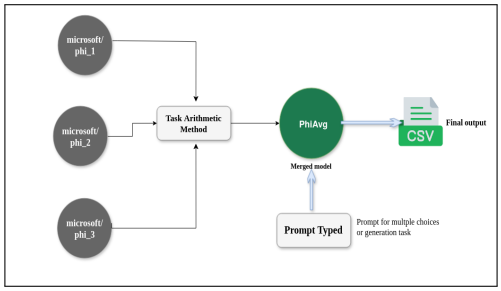
(a) LLM Knowledge Graph



(b) Mergekit process



(c) Prompt Engineering process



(d) KG-Prompt engineering LLM merging

Figure 1: Approaches used

Mathematical model : The resulting mathematical model combining KGs, prompt engineering and MergeKit is given as:

$$\text{output} = W_{\text{merged}}(P)(P, I) = \left(W_{\text{base}} + \sum_{j=1}^{T} \alpha_j \cdot \phi(P, d_j) \cdot \left(W_{\arg\max_i S_{ij}} - W_{\text{base}} \right) \right) (P, I)$$

Where: $W_{\text{merged}}(P)$ is the prompt-dependent merged model; W_{base} is the base model's parameter matrix; α_j is the importance weight for task d_j; $\phi(P, d_j)$ is the prompt weight function and $W_{\arg\max_i S_{ij}}$ is the parameter matrix of the selected model for task d_j.

2.2 Process to find the solution

The overall step to step process to implement the proposed approach is detailed in what follows:

Data Collection and Model Selection: The initial phase consists of identifying lightweight models available in open source, specialized in distinct tasks (e.g. text classification, summarization) relevant to the target complex task. These models were selected on the basis of factors like performance metrics.

Performance Assessment on Target Dataset: Each selected model was evaluated individually on the two separate data types (multiple choice and text generation). The dataset was split accordingly, allowing us to test the efficiency of each model in handling specific components of the task and to determine whether they could be retained in the merged model.

Knowledge graph construction: We created a knowledge graph (KG) to systematically represent each model and its performance. This graphical structure 2 allowed us to visualize the performance of each model on the datasets to easily choose which one to combine.

Architecture compatibility check and preparation for merging models: We checked that the models we intended to merge shared compatible architectures, which was essential for the merging process. This step also involved aligning the models with the merging approach described in the MergeKit framework .

Combining prompt engineering and merging models: After combining the selected models we designed suitable prompts which contain specific details or terms to enhance the models understanding of the required task or output.

Evaluation metrics and validation: We used evaluation metrics such as accuracy and rouge to assess the effectiveness of the combined model. Also, the results obtained were compared to the starter kit which served as benchmark for our results.
The following image represents the overall process used as the final solution to solve the problem posed.

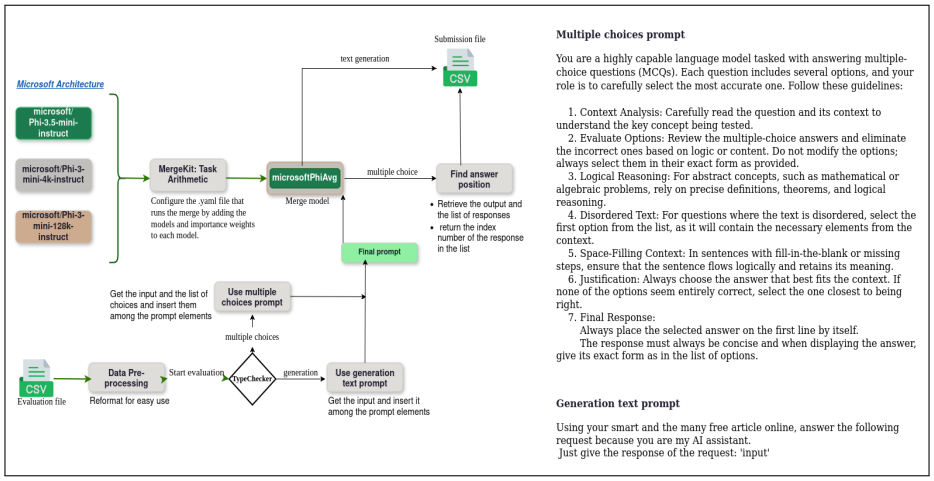


Figure 2: Merging Process

3 Experimentation and results

3.1 Experimentation environment

Due to the lack of resources, we had to use several options to meet the end of our experiment. First, we used a Dell Inc. Latitude 5580 Laptop, with an Intel® Core™ i7-7820HQ CPU @ 2.90GHz × 8, a RAM with a capacity of 16.0GiB, a graphics memory based on a NV117 / Mesa Intel® HD Graphics 630 (KBL GT2) and a hard disk of 512.1 GB. This solution was almost impossible because to start it was necessary to have an INVIDIA A6000 (48GB) which is why we moved on to the second solution. Google Colab was our second option but still not effective because it only provides 2 hours of use per day and disconnects quite quickly when you are in free mode. Finally, we used the free Kaggle environment which offered 30 hours of T4x2 GPU per week, which was very useful and allowed us to compete in the challenge.

3.2 Results

This section presents the results obtained using the different LLMs KG built and presented in the research methodology section. The legend for the long-form models that will be used in the tables is such that: **flan-t5-1**(lorahub/flan_t5_xl-wiki_qa_Is_This_True_), **flant5-2**(lorahub/flan_t5_xl-kilt_tasks_hotpotqa_complex_question), **meta-llama/Llama-2-7b-hf-1**(abcdabcd987/gsm8k-llama2-7b-lora-16), **meta-llama/Llama-2-7b-hf-2**(FinGPT/fingpt-forecaster_dow30_llama2-7b_lora), **microsoft_phi1**(microsoft/Phi-3.5-mini-instruct), **microsoft_phi2**(microsoft/Phi-3-mini-4k-instruct and **microsoft_phi3**(microsoft/Phi-3-mini-128k-instruct).

3.2.1 Starter Kit

After running the starter kit, we obtained the results shown in the following table.

Table 1: Starterkit launch results

Method	Fisrt model	Second model	Score
google/flan-t5-xl	flan-t5-1	flan-t5-2	**0.38**
meta-llama/Llama-2-7b-hf	meta-llama/Llama-2-7b-hf-1	meta-llama/Llama-2-7b-hf-2	**0.12**

3.2.2 Mergekit methods

Using the SLERP and TIES approach on the cosmos_qa and xsum datasets provided by the organizers, the outcome provided a score of **0.37** which was below the stater kit and thus not suitable for intended work. Also, SLERP could merge a limited amount of two models, which is also one of the reasons why it was left out. The selected approach uses Learning through addition, another aspect of the Task arithmetic.

Table 2: Results obtained after using two mergekit methods

Method	Fisrt model	Second model	Score
SLERP	flan-t5-1	flan-t5-2	**0.37**
TIES	flan-t5-1	flan-t5-2	**0.37**

3.2.3 Prompt Engineering

To see the impact of prompt engineering on the model's efficiency, we carried out a prediction on a model without prompt and in the second half we included the prompt. We then observe an increase in score of up to **0.25** on the individual models and **0.32** on merged models as depicted in Table 3 below. These results are explained by the ability of the prompt to improve the capacity of the models through its structuring and its precision in the address content.

Table 3: Prompt Engineering results

Model name	Task	Score Without prompt	Score with prompt
microsoft_phi1	multiple choices	0.10	0.35
microsoft_phi2	generation	0.12	0.22
phi_avg_1	-	0.22	**0.57**
microsoft_phi1	multiple choices	0.10	0.35
microsoft_phi3	generation	0.10	0.21
phi_avg_2	-	0.20	**0.56**
microsoft/Qwen/Qwen2-Math-7B-Instruct	multiple choices	0.25	0.35
microsoft_phi1	generation	0.15	0.22
qwen_phi_avg	-	0.40	**0.57**
microsoft_phi1	multiple choices	0.10	0.35
meta-llama/Meta-Llama-3-8B-Instruct	generation	0.12	0.22
phi_llama_avg	-	0.22	**0.57**

3.2.4 Combination of Mergekit and prompt engineexring

Table 4: Results obtained after combining a mergekit method with prompt engineering

Mergekit method	Fisrt model	Second model	Third model	Score
Task Arithmetic	microsoft_phi1	microsoft_phi2	microsoft_phi3	**0.63**

After selecting the appropriate models from their scores, they were merged using the merge kit arithmetic task 'learning through addition' and coupled with a suitable prompt. The selected models were three Microsoft Phi Models. The combination of these models associated with the power of the prompt allows us to be efficient at the user's request and to come out with a score of **0.63**. The output is greater than that of the starter kit by **0.25**, reflecting the impact of a well defined prompt on the model's performance. Also, the KG comes in handy providing a wide range of view on potential models which can be combined for greater yields.

4 Conclusion

The need for LLMs which can be accessed and used in developing countries is becoming a growing necessity. Motivated by computational limits, network issues and the cost to access powerful models the need for alternative methods to use them becomes necessary. Consequently, we proposed a model which combines cheap models efficient in sub tasks through KGs, Prompt engineering and LLM merging. The outcome gives promising results, providing evidence on the effectiveness of the proposed approach. This work highlights the impact of these approaches in the field of LLMs. Nevertheless, future works can be done to explore other LLM merging methods which can further enhance the output along different prompt engineering techniques. Such that the access to up to date, cheap and efficient LLMs can be made available to larger audience especially in developing countries.

References

[1] Alhafni, Bashar and Vajjala, Sowmya and Bannò, Stefano and Maurya, Kaushal Kumar and Kochmar & Ekaterina. (2024) *LLMs in Education: Novel Perspectives, Challenges, and Opportunities* arXiv preprint arXiv:2409.11917.

[2] Jiomekong, Azanzi and Oelen, Allard and Auer, Soren and Anna-Lena, Lorenz and Lars & Vogt. (2024) *Food information engineering* 338–353 AI Magazine, Wiley Online Library.

[3] Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, rian Benedict, Mark McQuade. & Jacob Solawetz. (2024) *Arcee's MergeKit: A Toolkit for Merging Large Language Models* Arcee, Florida, USA.

[4] Dongfu Jiang, Xiang Ren. & Bill Yuchen Lin. (2023) *LLM-BL E N D E R: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion* Allen Institute for Artificial Intelligence, University of Southern California, Zhejiang University.