

Re-Tuning: Overcoming the Compositionality Limits of Large Language Models with Recursive Tuning

Eric Pasewark, Kyle Montgomery, Kefei Duan, Dawn Song, Chenguang Wang

July 26, 2024

Washington University in St. Louis & UC Berkeley

It is well known that language models can perform many tasks that were in the training data.

Question: how well can they solve problems that are more difficult than those in the training data?

Question: how well can they solve problems that are more difficult than those in the training data?

We explore this question in the context of "Length Generalization"

In this setup, we analyze how the language model solves problems of various lengths, where longer problems are usually more difficult.

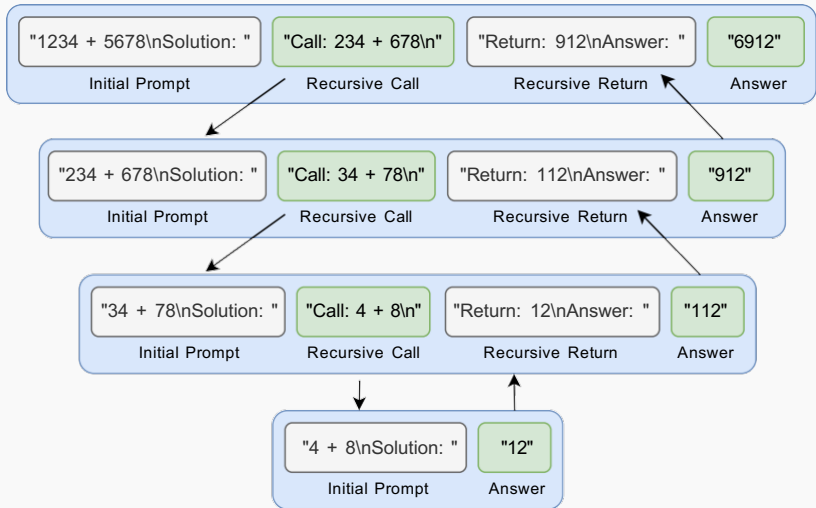
Following previous work, we train on problem examples up to a certain length, then evaluate performance on longer examples than seen in training.

Basic idea: train a language model to break a problem down into subproblems and solve these subproblems separately

Given a prompt to solve a problem, the Re-Tuning method works as follows:

1. The language model creates prompts for subproblems.
2. These prompts are extracted and sent to separate contexts, where the subproblem is solved.
3. The language model uses the subproblem solution to solve the original problem.

Re-Tuning Overview



We present graphs of performance for 3 models:

1. Baseline: Trained to simply output answer after prompt.
2. Scratchpad: Trained to output steps to solve the problem before the final answer.
3. Re-Tuning: Our new method as described.

All 3 of these are finetuned from the Llama 7B model (Touvron et al. [2023](#)).

Additionally, we present results from previous papers for comparison.

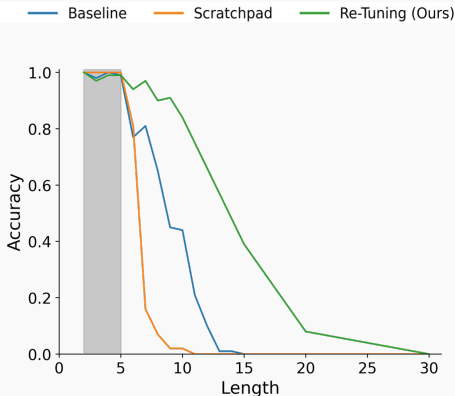
Addition: Given 2 numbers, compute their sum.

Parity: Given a list of 0s and 1s, determine if there is an even or odd amount of 1s.

Dynamic Programming: "Given a sequence of integers, find a subsequence with the highest sum, such that no two numbers in the subsequence are adjacent in the original sequence." - Dziri et al. [2023](#)

For all these problems, we train our model on instances up to a certain length, then evaluate performance on longer instances.

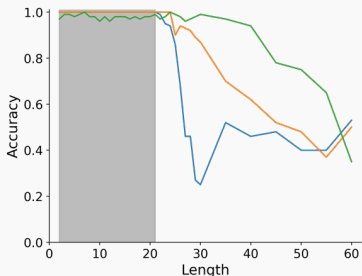
Dynamic Programming Results



Previous Result: Dziri et al. [2023](#) train a language model on the dynamic programming problem up to length 5, where it achieves 100% accuracy. At length 6 it has about 20% accuracy and by length 10 it has 0% accuracy.

Parity Results

— Baseline — Scratchpad — Re-Tuning (Ours)

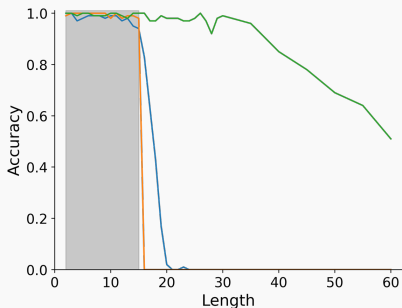


Problem Description: given a list of 0s and 1s, determine the number of 1s mod 2.

Previous Result: Anil et al. [2022](#) train a language model on parity up to length 21, where it achieves perfect accuracy. It falls to around 50% accuracy at length 40.




Addition Results



— Baseline — Scratchpad — Re-Tuning (Ours)



Previous Result: Liu and Low [2023](#) train a language model on data up to 15 digits, where it achieves 100% accuracy. Accuracy falls to 0% at 21 digits.

References

-  Anil, Cem et al. (2022). **“Exploring Length Generalization in Large Language Models”**. In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. url: <https://openreview.net/forum?id=zSkYVeX7bC4>.
-  Dziri, Nouha et al. (2023). ***Faith and Fate: Limits of Transformers on Compositionality***. arXiv: [2305.18654 \[cs.CL\]](https://arxiv.org/abs/2305.18654).
-  Liu, Tiedong and Bryan Kian Hsiang Low (2023). ***Goat: Fine-tuned LLaMA Outperforms GPT-4 on Arithmetic Tasks***. arXiv: [2305.14201 \[cs.LG\]](https://arxiv.org/abs/2305.14201).

-  Nye, Maxwell et al. (2021). ***Show Your Work: Scratchpads for Intermediate Computation with Language Models.*** arXiv: [2112.00114 \[cs.LG\]](https://arxiv.org/abs/2112.00114).
-  Touvron, Hugo et al. (2023). ***LLaMA: Open and Efficient Foundation Language Models.*** arXiv: [2302.13971 \[cs.CL\]](https://arxiv.org/abs/2302.13971).