Learning Multi-step Reasoning by solving Arithmetic Tasks

Tianduo Wang and Wei Lu

StatNLP Research Group, Singapore University of Technology and Design

Introduction

- **Making language models (LMs) perform multi-step reasoning is a valuable, yet challenging objective.**

While LMs are powerful in many NLP tasks, there are works showing that LMs' abilities to perform complex multi-step reasoning on math tasks can still be improved.

- **Chain-of-thought (CoT) prompting elicits large LMs' multi-step reasoning abilities.**

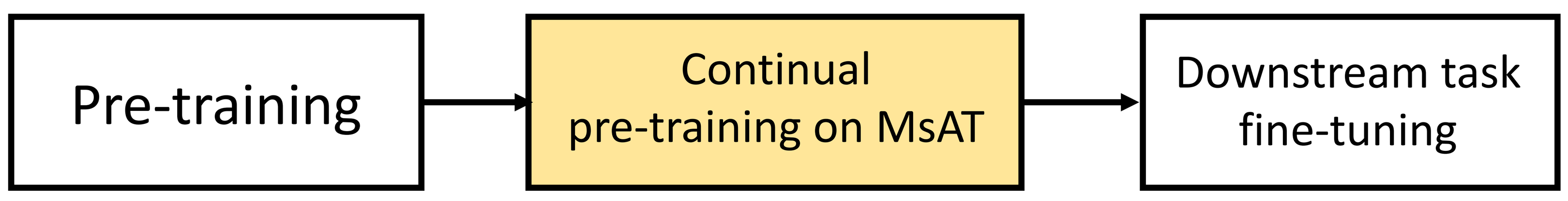
The success of CoT prompting can be attributed to large LMs' ability to decompose a complex problem into several intermediate steps. However, it is believed that such ability only emerge from LMs with sufficient parameters (> 100 B).

- **Can we inject the multi-step reasoning ability into smaller LMs (e.g., RoBERTa) in a post-hoc way?**

Reasoning ability injection

- **Small LMs are lack of numerical reasoning ability.**
- **Therefore, we propose to inject such ability into a LM by continually pre-training it on a novel synthetic seq2seq task.**

Our method essentially appends a continual pre-training stage before fine-tuning LMs on downstream tasks. We propose a synthetic pre-training task MsAT (**M**ulti-**s**tep **A**rithmetic **T**asks) for reasoning ability injection.



Multi-step Arithmetic Task (MsAT)

- **MsAT is a synthetic Seq2Seq task that is composed of arithmetic questions.**

The input of an example from MsAT describes an arithmetic question that only contains symbols and numbers. The output is a piece of executable **Python** code that solves the question step-by-step.

- **Example from MsAT**

Input:

$\underbrace{\text{W=2. Y=8. Z=15. V=3.}}_{\text{Question context}}$ $\underbrace{\text{(X - Y) / W + V = Z.}}_{\text{Equation}}$ $\underbrace{\text{X=?}}_{\text{Question variable}}$

Output:

$N_0 = 15.\ N_1 = 3.\ C_0 = N_0 - N_1.$ (step 1)
$N_2 = 2.\ C_1 = C_0 * N_2.$ (step 2)
$N_3 = 8.\ Ans = C_1 + N_3.$ (step 3)

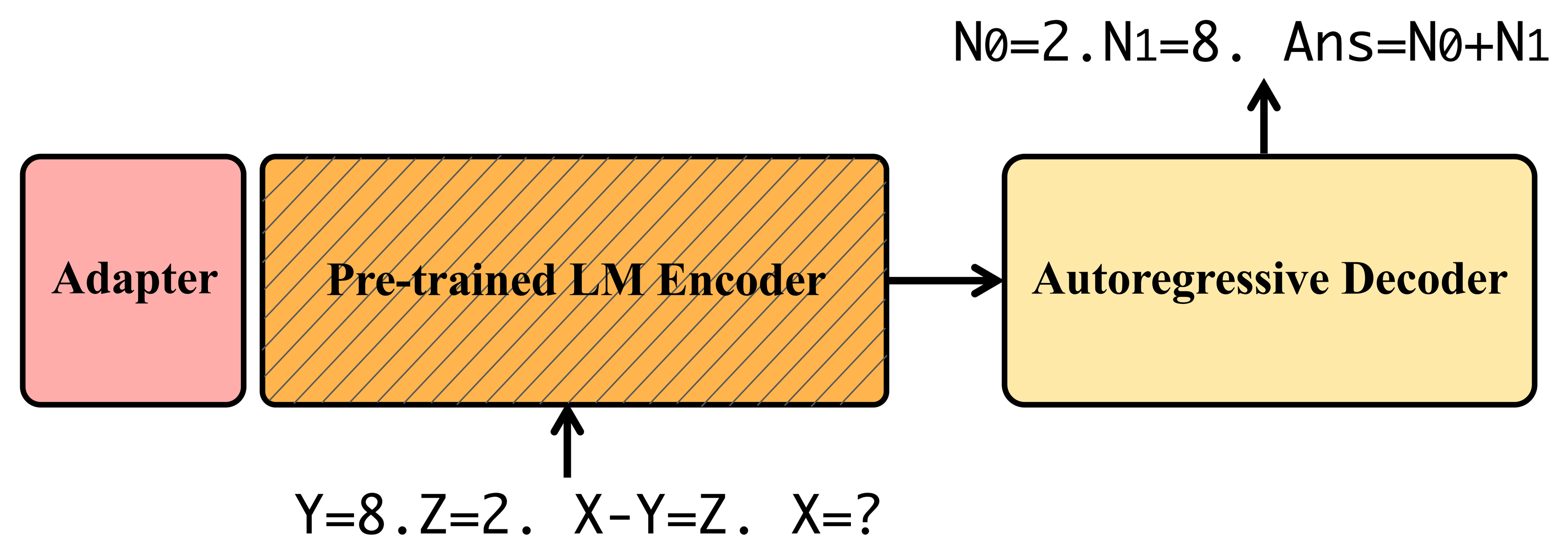
MsAT pre-training

- **LMs are pre-trained to solve arithmetic tasks by writing code instead of making explicit computation.**

This is because LMs' computation results are not reliable. The generated code can be compiled and executed by an external Python interpreter.

- **To avoid catastrophic forgetting by adapter-tuning**

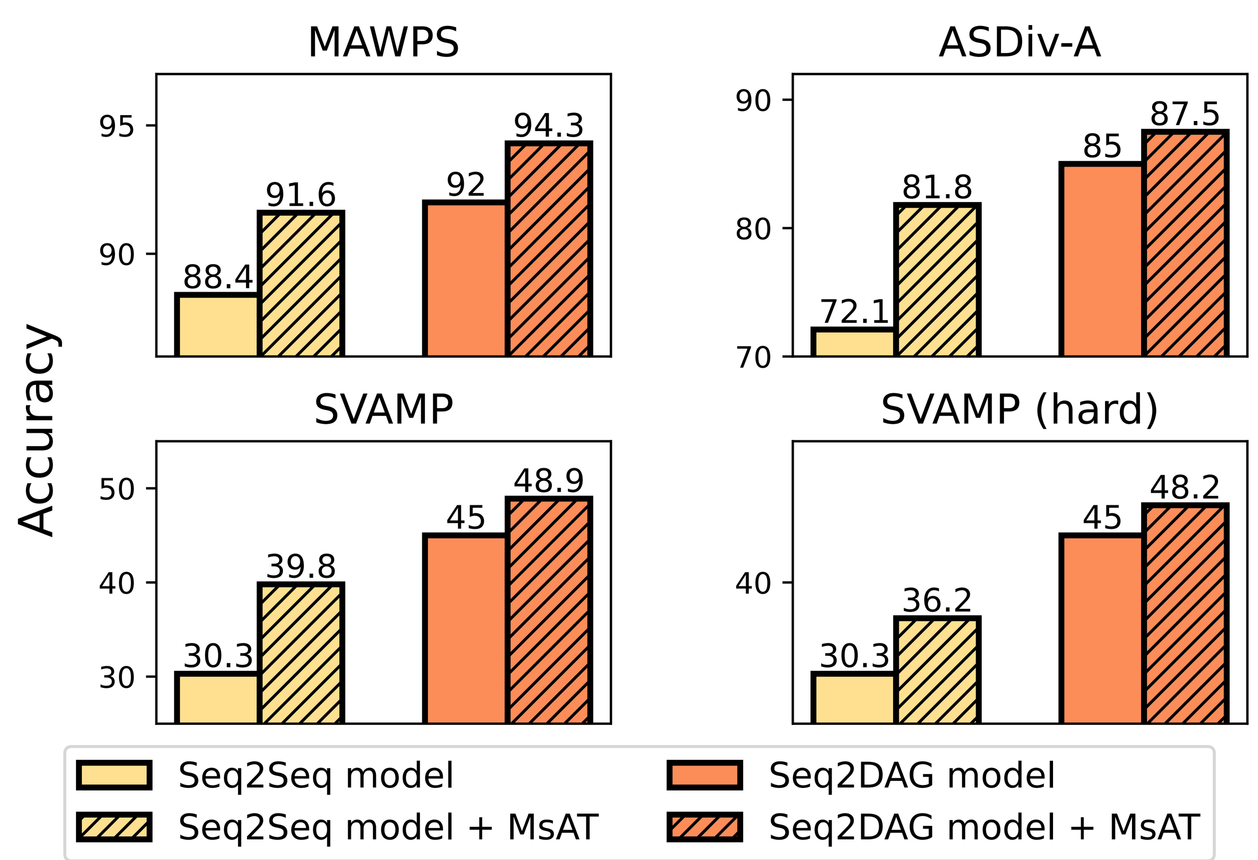
Directly tuning all parameters of a pre-trained LM on MsAT can lead to catastrophic forgetting. Hence, we apply adapter-tuning to maintain LMs' language prowess as much as possible.



Experiments

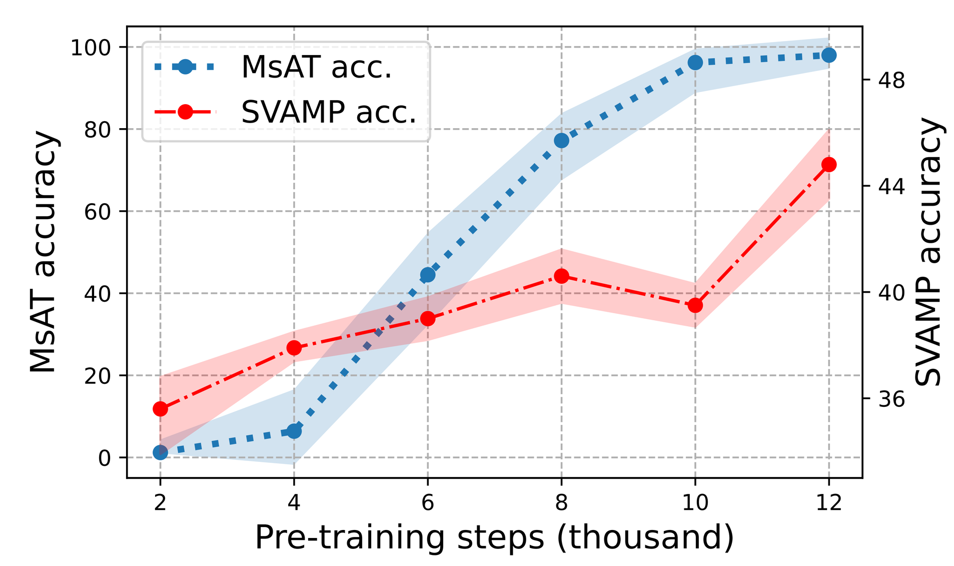
- **We evaluate our pre-training method with two backbone models on four datasets.**

Backbone models:
(1) Seq2Seq model: RoBERTa + Transformer decoder
(2) Seq2DAG model: RoBERTa + Directed Acyclic Graph decoder



- **Pre-training analysis**

We plot the pre-training trajectory as well as the effect of pre-training steps on the performance of SVAMP.



- **Our code is available on GitHub:**

https://github.com/TianduoWang/MsAT