Review of LLaMA PRO
Progressive LLaMA with Block Expansion

Vasilescu Andreea

University of Bucharest
Faculty of Mathematics and Informatics
Masters of Natural Language Processing

January 2024

Table of Contents

Table of Contents

Introduction

LLaMA-Pro is a progressive version of the original model with 8.3 billion parameters. The intended use is designed for NLP taks, programming, mathematics and general language tasks.

Aim of the study

The researchers took the LLaMA 7B as base and they expand it it LLaMA PRO, then they gave instruction tunes and created the Instruct as first result. As a second result, they took the same LLaMA 7B as base, and expand it to Chat.
How did they do it? They took the base model, add new layers with the scope of teaching new stuff, while focusing on the catastrophic forgetting.

Visual representation

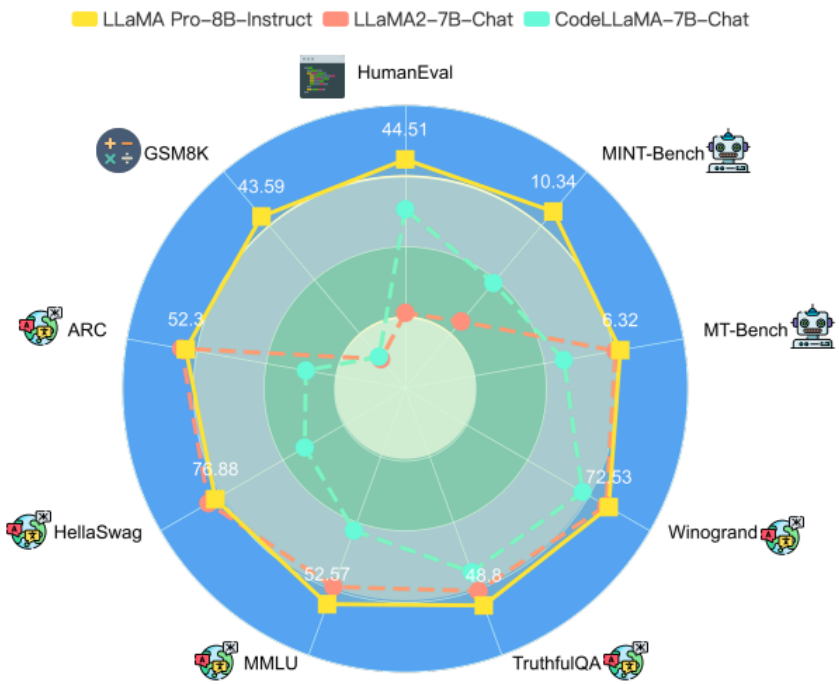


Figure 1: LLAMA PRO - INSTRUCT delivers state-of-the-art performance across a wide variety of tasks, ranging from general language to specific domains, superior to existing models from the LLaMA series.

Model Architecture

The architecture of LLaMA to has the input layer which has token embedding for the input, and then there are the output tokens and it is passed to blocks of the transformer. Transformer blocks have attention mechanism and normalization.
They took the blocks, duplicate them and train only the ones that have been duplicated.
They only train the newly added layers, while freezing the others (0 parameters). Authors affirm that they retain the old knowledge while adding new knowledge.
They take every end layer, make sure that it outputs 0 and then they copy it and train just with the new one, while the rest of them are fron

Model Architecture

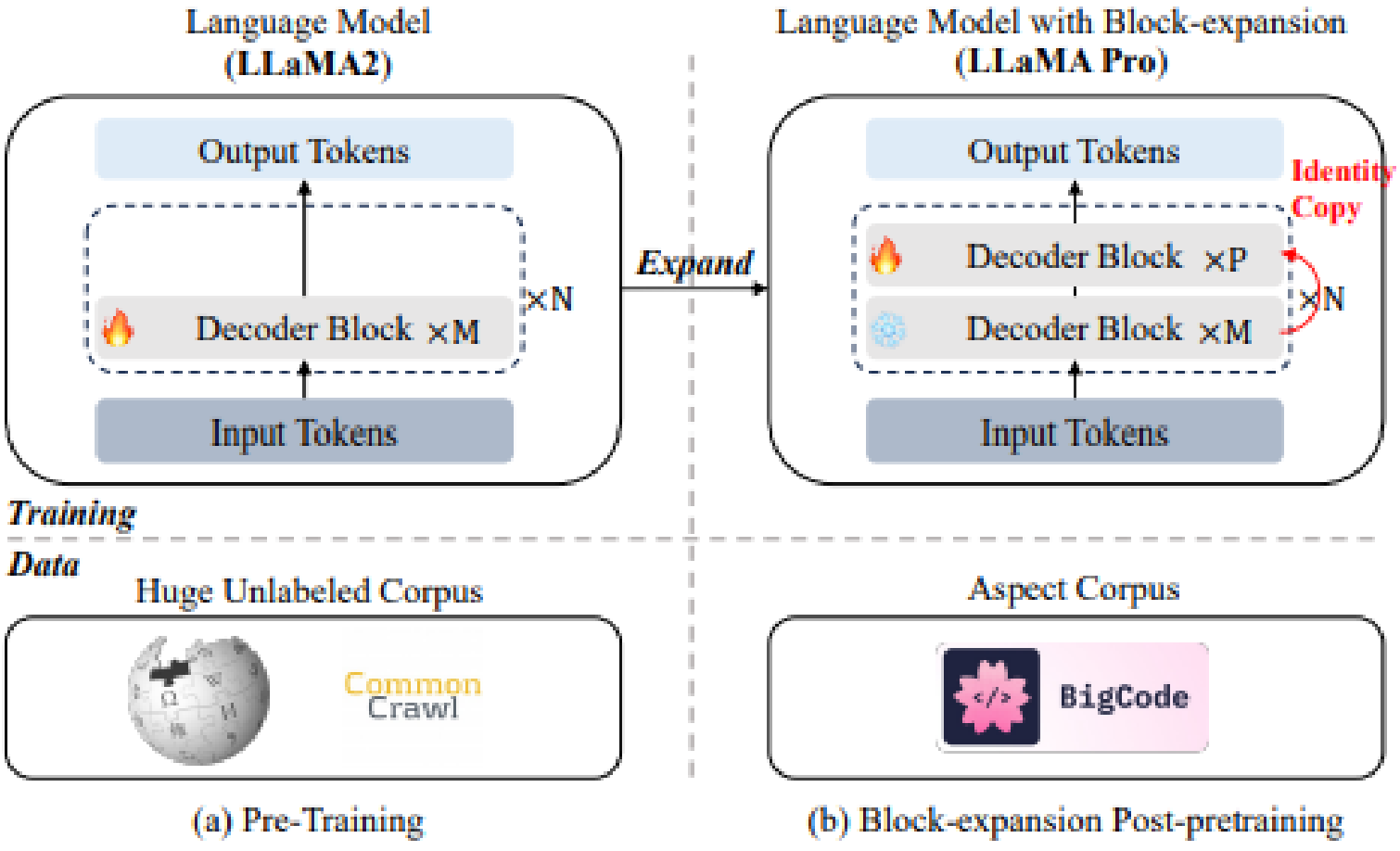
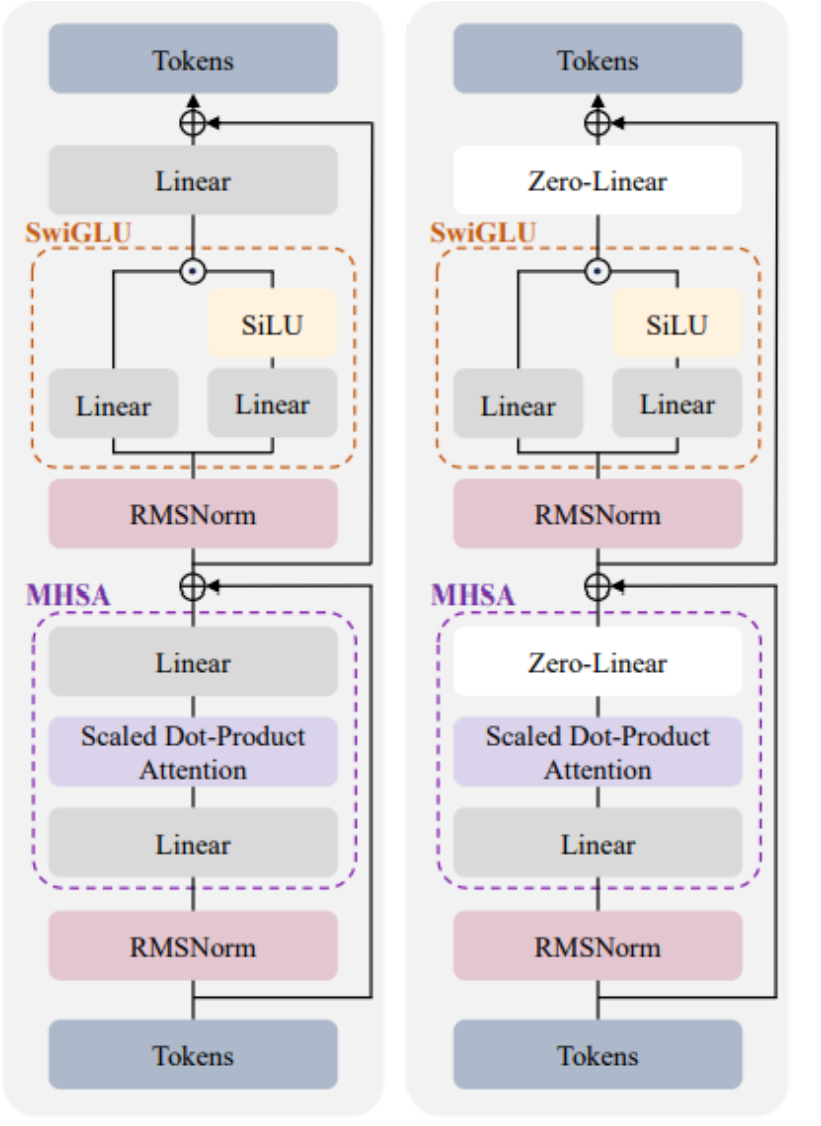


Figure: Expansion of LLaMA2 to LLaMA Pro [**?**]

Pre-training

- The authors pre-train the LLaMA PRO's expanded blocks on 80B tokens using open-source code and math data for 2830 GPU Hours (16 NVIDIA H800 GPUs for about 7 days).
- Further, they perform supervised instruction tuning (fully fine-tuning of all the blocks, aka SFT) on LLAMA PRO with approximately 80M tokens, yielding LLAMA PRO - INSTRUCT.
- It is noted that pre-trained models produced by their block expansion method are well-compatible with the subsequent SFT techniques without specific modification.

Training



(a) LLaMA Block (b) LLaMA Block after Identity Copy

Pre-processing

- On the left it would be the original layer, and then on the right it will be the newly resulted one. They are making sure that the weight matrix should be 0 so they can give a output to be sure the entire result is a copy, to ensure all is a residual connection of the initialization.
- On the training they will allow all weights on the (b) to be updated.
- If I would have to summarize everything, they go to an already existing layer, copy it with the method explained above, freeze anything else, and train. From the paper I understand that they modify blocks of four to five.

Results

This progressive model is at least as good as the already existing ones.

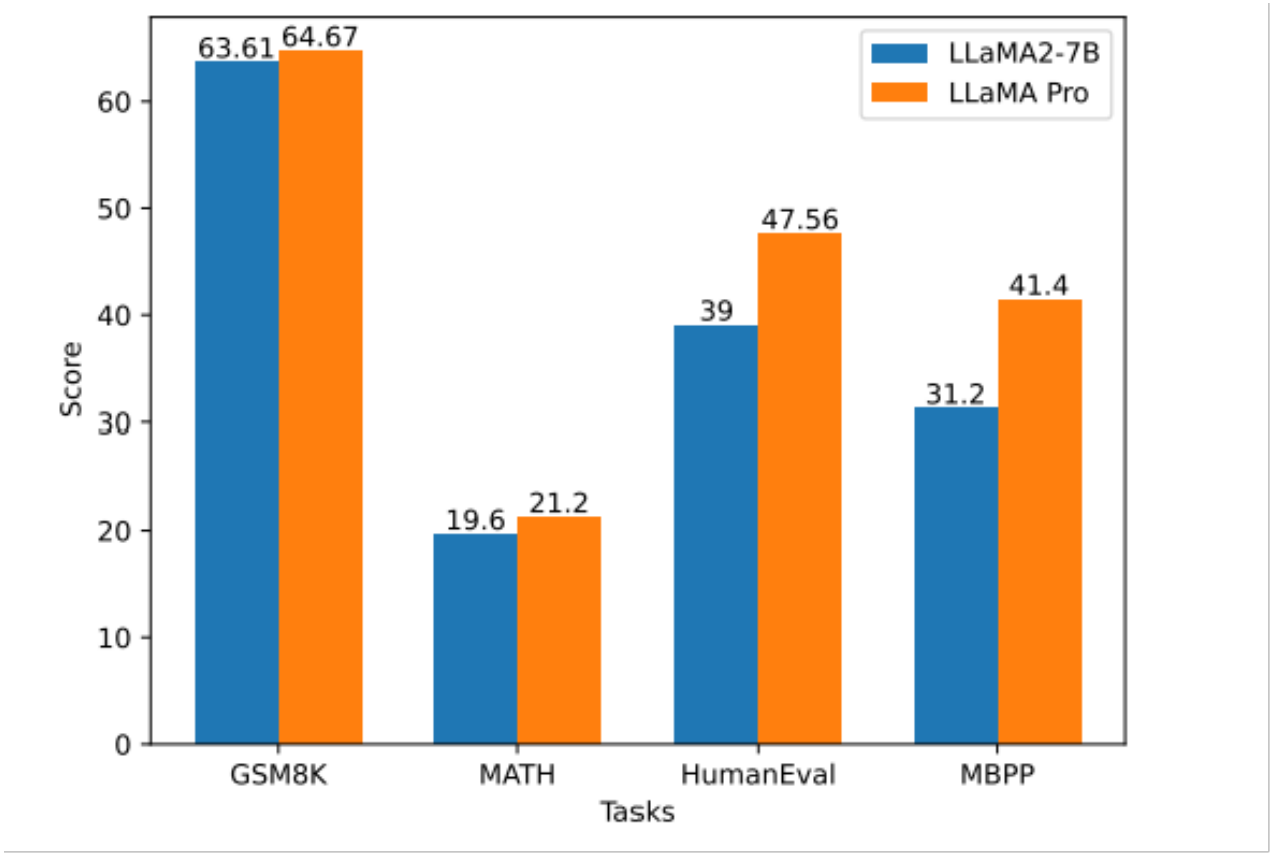


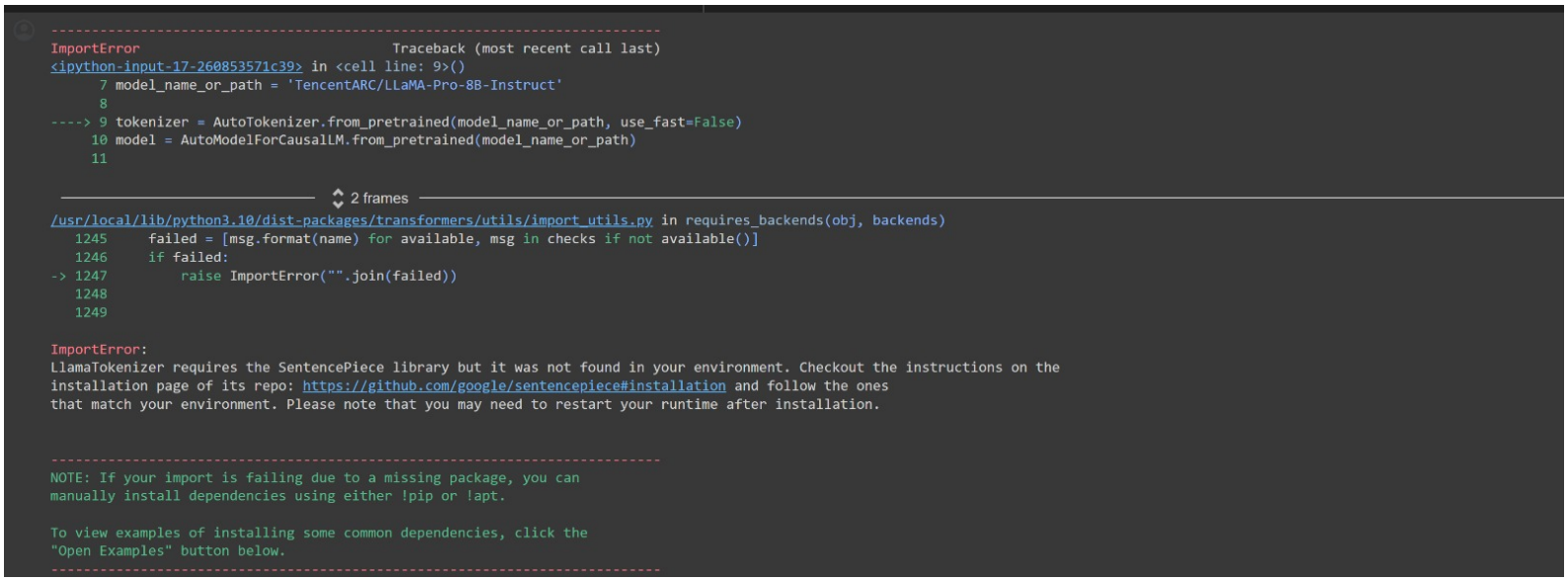
Figure: [?]

- The authors have clearly explained what they did.
- They have explained the general principle of the method they used.
- The article was comprehensive, the key message was recognizable and it has lots of drawings and tables.
- I would've liked to have more about the tests they did to be sure they retain the initial data.

Test LLaMA-Pro

I have tried to test the LLaMA-Pro and see how it goes, but I have experienced some issues:

- Initially I wanted to use google colab, but the first issue I have encountered was with gradio. After fixing it, I encountered the following:



```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-17-260853571c39> in <cell line: 9>()
      7 model_name_or_path = 'TencentARC/LLaMA-Pro-8B-Instruct'
      8
----> 9 tokenizer = AutoTokenizer.from_pretrained(model_name_or_path, use_fast=False)
     10 model = AutoModelForCausalLM.from_pretrained(model_name_or_path)
     11

                                  2 frames
/usr/local/lib/python3.10/dist-packages/transformers/utils/import_utils.py in requires_backends(obj, backends)
   1245     failed = [msg.format(name) for available, msg in checks if not available()]
   1246     if failed:
-> 1247         raise ImportError("".join(failed))
   1248
   1249

ImportError:
LlamaTokenizer requires the SentencePiece library but it was not found in your environment. Checkout the instructions on the
installation page of its repo: https://github.com/google/sentencepiece#installation and follow the ones
that match your environment. Please note that you may need to restart your runtime after installation.


---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

Test LLaMA-Pro 2

- Second approach was to download LM studio.