**Pre-Training LLMs – Upstage: Sung Kim and Lucy Park**

**Introduction**

What is pre-training?

Pre-training a Large Language Model is a process of taking a model, generally a transformer neural network, and training it on a large corpus of text using supervised learning, so that it learns to repeatedly predict the next token given an input prompt.

Pre-training = first step of training an LLM before any fine-tuning to have it follow instructions, or further alignment to human preferences is carried out.

What is a base model?

The output of a pretraining is known as a base model, and will cover both training from scratch (meaning from randomly initialized weights), as well as taking a model that is pre-trained and continuing the pre-training process on your own data.

Constrains in pre-training from scratch

Training a LLM from scratch is computationally expensive (a minimum cost of maybe about $1k for the smallest models and up to tens of thousands of dollars to hundreds of thousands of dollars for maybe a billion parameter scale model), requiring multiple state-of-the-art GPUs and training runs that can last weeks or months. For this reason, most developers won’t pretrain a model from scratch. They take an existing model, and either use prompting or use fine-tuning to adapt it to their own tasks.

Situations where pre-training may still be preferred

However, there are still some situations where pre-training a model may be required or preferred.

Upstage: has been doing pre-training for its customers. Some customers are building models for tasks in specific domains like legal, healthcare, and e-commerce. Others need models with stronger abilities in specific languages such as Thai and Japanese.

Depth Upscaling

New training methods are making more efficient pre-training possible, like Depth Upscaling, which uses 2 or more sets of existing models to build larger models. Because of this technological improvement, there are more and more interest in pre-training.

Depth upscaling creates a new, larger model by duplicating layers of a smaller pre-trained model. The new model is then further pre-trained resulting in a better, larger model than the original. Models created in this way can be pre-trained with up to 70% less compute than traditional pre-training, representing a large cost saving.

Is pre-training suitable for you?

Whether pre-training is suitable for your work depends on several factors, such as whether a model might already be available that might work for your task without pre-training, and what data you have available, as well as the compute resources you have access to, both for training and serving. And lastly, the privacy requirements you may have, which may also implicate regulatory compliance requirements.

What will be learnt in this course

In this course, you would learn all of the necessary steps to pre-train a model from scratch, from gathering and preparing training data, to configuring a model and training it. You will:

* Start by looking at some use cases where pre-training a model is the best option to get good performance.
* Discuss the difference between pre-training and fine-tuning.
* Walk through data preparation steps that are required to pre-train a model.
* Explore how to gather data from the internet, or existing repositories like HuggingFace.
* Look at the steps to obtain high quality training data, including deduplication, filtering on the length of text examples, and language cleaning.
* Explore some options for configuring model architecture.
* See how to modify Meta’s Llama models to create larger or smaller models, and then look at a few options for initializing weights, wither randomly or from other models.
* See hoe to train a model using the open source HuggingFace library and actually run a few steps of training to observe how the loss decreases as the training progresses.

This course uses smaller models with just a few million parameters to keep things lightweight enough to run on a CPU, but you will be able to use the code from the lessons to scale to both larger datasets and models, and also to train on GPUs.

**Lesson 1: Why pre-training?**

Explore a few options where pre-training is the best option to get good performance. You will also try generating some text with different versions of the same model to see how the performance differs between a base general model, a fine-tuned model, and a specialized pre-trained model.

Multi-stage Continual Training

A diagram of a model

AI-generated content may be incorrect.

Fine-tuning – Make them better at following instructions

Alignment – Align with human preferences to make them safe and helpful.

The model only has knowledge of the content that was in the training data. So if you want the model to have more knowledge you have to do more training on more data. Additional fine-tuning or alignment training is useful to teach the model a new behavior or avoiding a particular topic.

A diagram of a model

AI-generated content may be incorrect.

People often try to add new knowledge without pre-training, focusing on fine-tuning the model with smaller datasets. However, this doesn’t work in every situation, especially if the new knowledge is not well represented in the base model. In these cases, additional pre-training is required to get good performance.

Jupyter Notebook here

Price Comparison

A screenshot of a calculator

AI-generated content may be incorrect.

**Lesson 2: Data Preparation**

Data used by Pre-training an Autoregressive text generation model: Vast amount of unstructured text.

Data used in fine-tuning is highly structured (Instruction-response pairs). The goal of fine-tuning is to get the model to behave in a certain way, or to get good at completing a specific task.

Pre-training is like reading many books. Fine-tuning is like a practice exam.

A screenshot of a computer test

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.

Data Quality is very important in pre-training LLMs. If there are issues with pre-training data, then the resulting LLM would not perform well. Need to ensure that the training data is with high quality.

A screen shot of a data cleaning

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Lesson 3: Packaging Data for Pretraining**

Need to data manipulation.

Data Packaging: Tokenizing + Packing

Tokenizing: Breaking each text into smaller, meaningful units, which are called tokens.

* Tokenization breaks text into smaller units called tokens (numbers in a list sequence). It depends on the vocabulary and the tokenization algorithm of the model. Each model has a specific tokenizer.

Packing: Packing tokens into the maximum sequence length to improve training efficiency.

A screenshot of a packing box

AI-generated content may be incorrect.