Ch01 - Understanding large language Models

Video Lecture:

Developing an LLM: Building, Training, Finetuning

REFERENCES:
1. Build an LLM from Scratch book: https://amzn.to/4fqvn0D
2. Build an LLM from Scratch repo:

https://www.youtube.com/watch?v=kPGTx4wcm_w

Notes from book

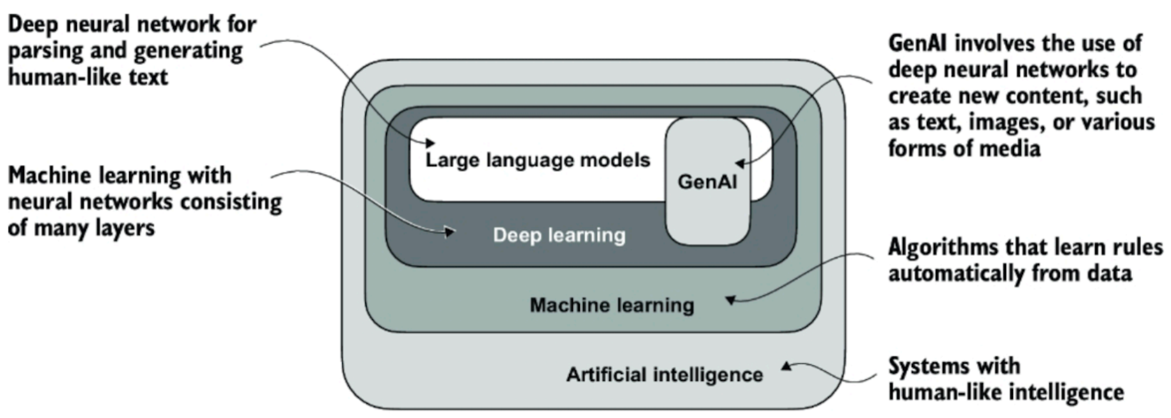
> The success of LLMs can be attributed to **transformer architecture** (which is part of most of LLMs) and **large amount of data.**

What is an LLM?

LLM is a neural network designed to understand, generate and respond to human-like text.

> LLMs utilize an architecture called the ***transformer***, which allows them to pay selective attention to different parts of the input when making predictions, making them especially adept at handling the nuances and complexities of human language.

LLMs also comes under GenAI as it can generate the text.


Deep neural network for parsing and generating human-like text
Machine learning with neural networks consisting of many layers
Large language models
GenAI
Deep learning
Machine learning
Artificial intelligence
GenAI involves the use of deep neural networks to create new content, such as text, images, or various forms of media
Algorithms that learn rules automatically from data
Systems with human-like intelligence

Stages of building and using LLMs ?

LLMs has two-main stages : Pretraining and Fine-tuning

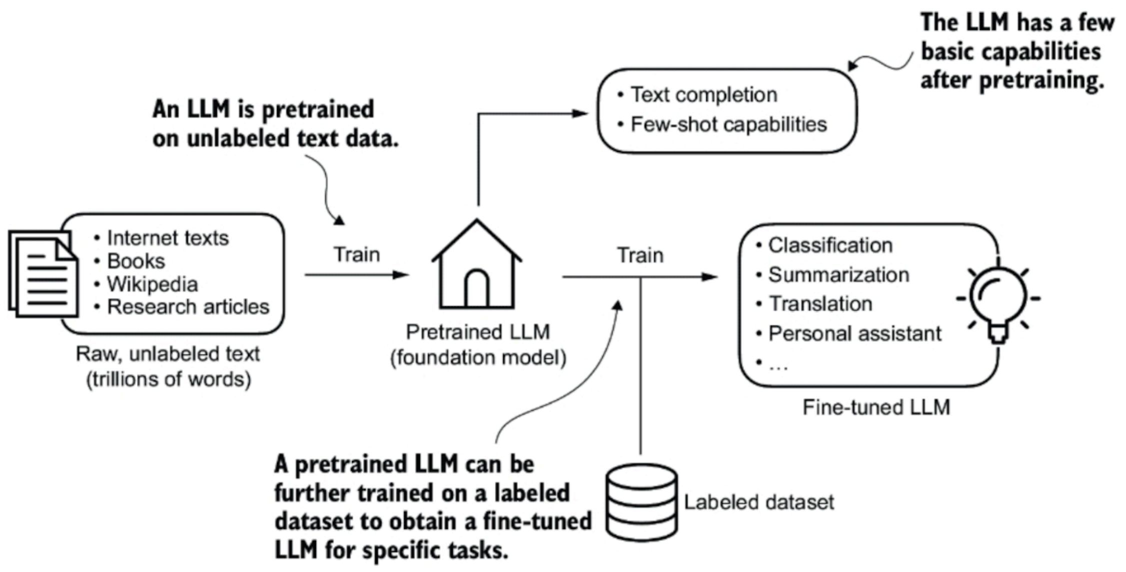

The LLM has a few basic capabilities after pretraining.
• Text completion
• Few-shot capabilities
An LLM is pretrained on unlabeled text data.
• Internet texts
• Books
• Wikipedia
• Research articles
Raw, unlabeled text (trillions of words)
Train
Pretrained LLM (foundation model)
Train
• Classification
• Summarization
• Translation
• Personal assistant
• ...
Fine-tuned LLM
A pretrained LLM can be further trained on a labeled dataset to obtain a fine-tuned LLM for specific tasks.
Labeled dataset

Figure 1.3 Pretraining an LLM involves next-word prediction on large text datasets. A pretrained LLM can then be fine-tuned using a smaller labeled dataset.

> **NOTE:** Readers with a background in machine learning may note that labeling information is typically required for traditional machine learning models and deep neural networks trained via the conventional supervised learning paradigm. This is not the case for the pretraining stage of LLMs. In this phase, *LLMs use self-supervised learning,*

> *where the model generates its own labels from the input data.*

Stage1: Pre-training

Creating initial LLMs are also called as base or foundation Models. A typical example is ChatGPT-3 base or foundation model which are capable of completing the sentences and also it has few shot setting too which allows it complete tasks with very few examples.

Stage2 : Fine-tuning

After training the model on large text dataset to predict the next word, it can be also trained on labeled data for specific tasks also known as ***fine-tuning***.

Popular categories to fine-tune the LLMs

1. Instruction fine-tuning : Instruction and answers pairs. eg. query to translate with translated text.
2. classification fine-tuning: text with labels, eg. email → spam or not spam.

Introducing the transformer Architecture

▼ Most modern LLMs rely on transformer architecture, paper "Attention is All You Need" in 2017 introduced it. (https://arxiv.org/abs/1706.03762).

▼ We must understand the transformer, which was initially developed for machine translations from English to german and french.

▼ Simplified version of that architecture is illustrated below:

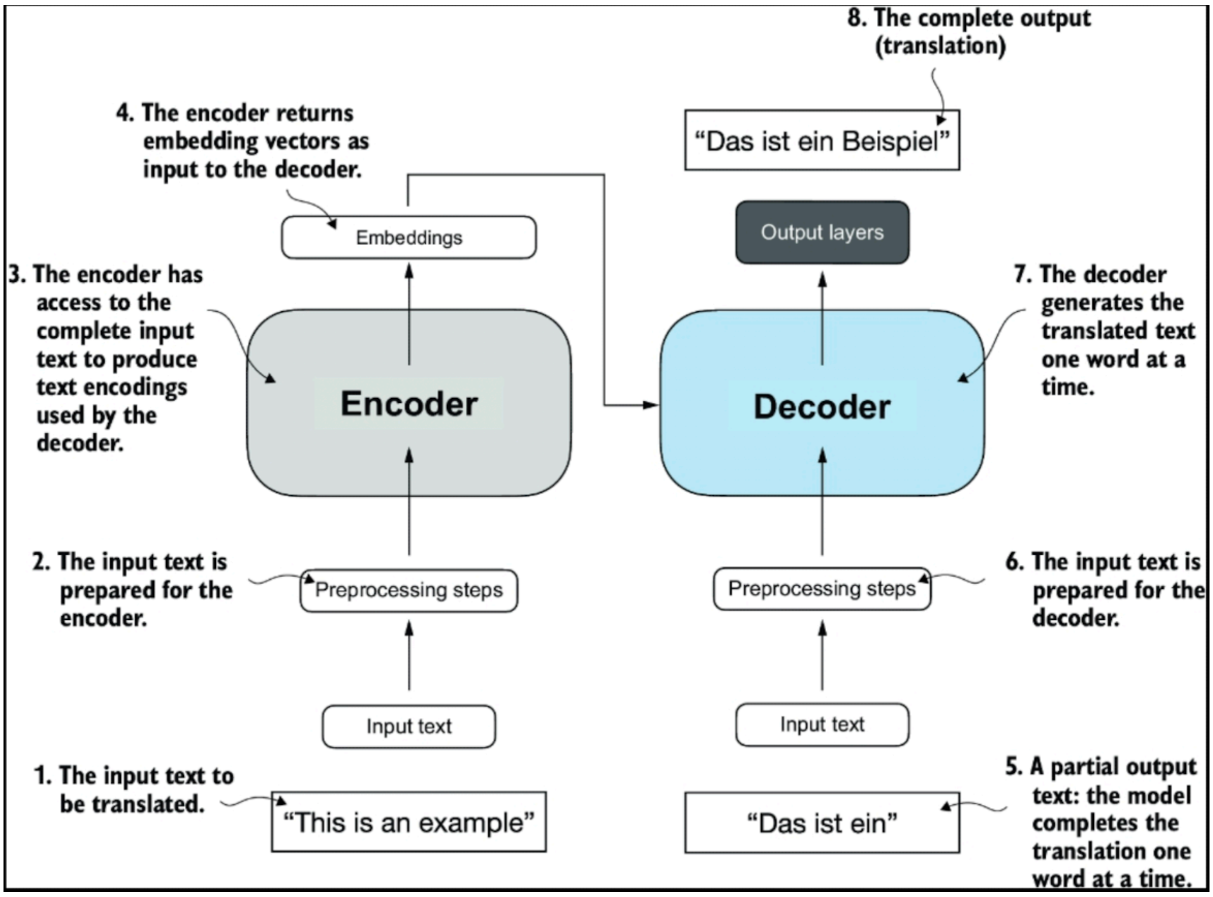

8. The complete output (translation)
"Das ist ein Beispiel"
4. The encoder returns embedding vectors as input to the decoder.
Embeddings
Output layers
3. The encoder has access to the complete input text to produce text encodings used by the decoder.
Encoder
Decoder
7. The decoder generates the translated text one word at a time.
2. The input text is prepared for the encoder.
Preprocessing steps
Preprocessing steps
6. The input text is prepared for the decoder.
Input text
Input text
1. The input text to be translated.
"This is an example"
"Das ist ein"
5. A partial output text: the model completes the translation one word at a time.

Figure 1.4 A simplified depiction of the original transformer architecture, which is a deep learning model for language translation. The transformer consists of two parts: (a) an encoder that processes the input text and produces an embedding representation (a numerical representation that captures many different factors in different dimensions) of the text that the (b) decoder can use to generate the translated text one word at a time. This figure shows the final stage of the translation process where the decoder has to generate only the final word ("Beispiel"), given the original input text ("This is an example") and a partially translated sentence ("Das ist ein"), to complete the translation.

The transformer architecture consist of two main submodules:

1. **Encoder :** Process the input text data and covert it into the numerical representation or vector which captures the contextual information.
2. Decoder: Take these encoded vectors and produce the output text.

Both encoder and decoder have multiple layers connected with so-called **self-attention mechanism.**

> **Self-attention Mechanism** : A key element of transformer and LLMs is itself Self attention mechanism, which weighs the importance of works or tokens with respect to each others. This mechanism allows model to captures the long range dependencies and contextual relationships from the input data.

Later the variants of transformers such as : ***BERT : Bidirectional Encoder Representations of Transformers*** and ***GPT : Generative Pre-trained Transformers***, built on transformers to adapt to specific tasks.

BERT:

BERT and its variants specialize in masked word prediction, where the model predicts masked or hidden words in a given sentence, as shown in figure 1.5. This unique training strategy equips BERT with strengths in text classification tasks, including sentiment prediction and document categorization. As an application of its capabilities, as of this writing, X (formerly Twitter) uses BERT to detect toxic content.

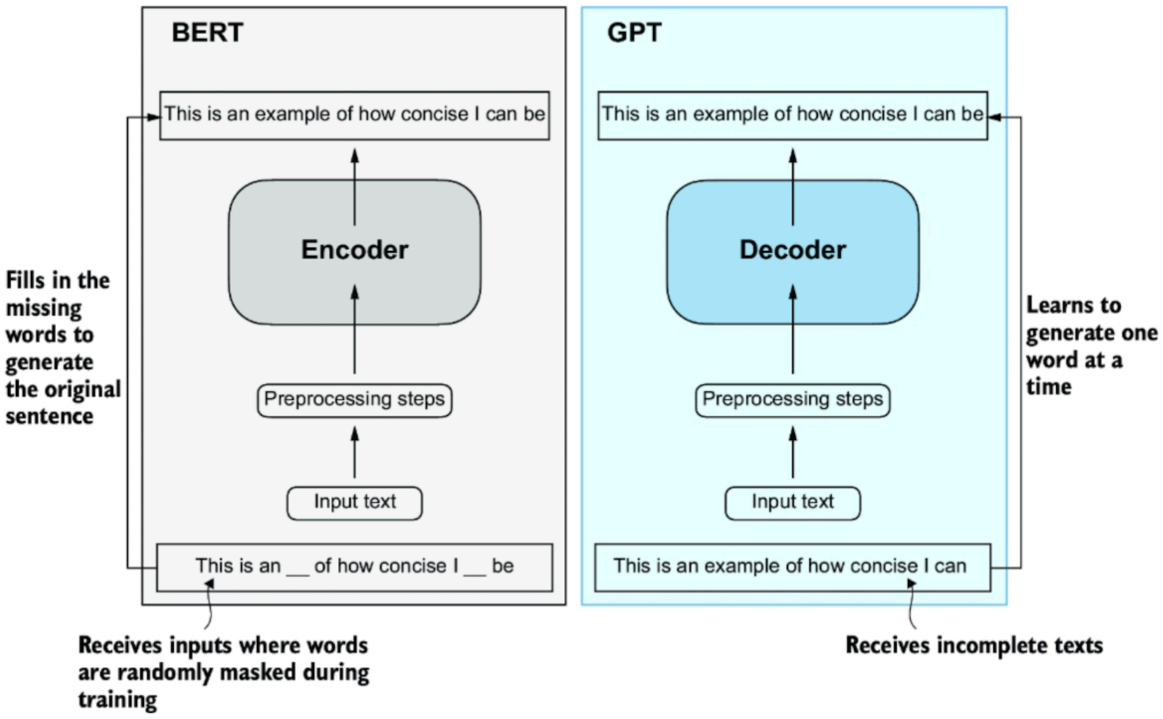

BERT
This is an example of how concise I can be
Encoder
Preprocessing steps
Input text
This is an __ of how concise I __ be
Fills in the missing words to generate the original sentence
Receives inputs where words are randomly masked during training
GPT
This is an example of how concise I can be
Decoder
Preprocessing steps
Input text
This is an example of how concise I can
Learns to generate one word at a time
Receives incomplete texts

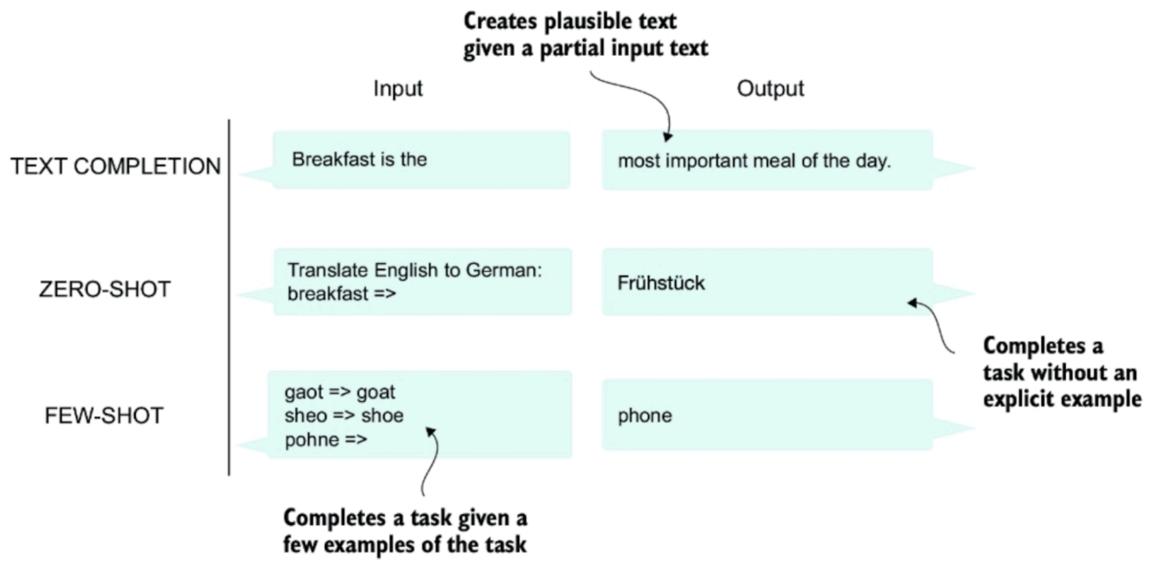
Figure 1.5 A visual representation of the transformer's encoder and decoder submodules. On the left, the encoder segment exemplifies BERT-like LLMs,

which focus on masked word prediction and are primarily used for tasks like text classification. On the right, the decoder segment showcases GPT-like LLMs, designed for generative tasks and producing coherent text sequences.

GPT-3 (2020) :

GPT, on the other hand, focuses on the decoder portion of the original transformer architecture and is designed for tasks that require generating texts. This includes machine translation, text summarization, fiction writing, writing computer code, and more.

GPT is good in zero shot and few-shot learning also.


Creates plausible text given a partial input text
Input
Output
TEXT COMPLETION
Breakfast is the
most important meal of the day.
ZERO-SHOT
Translate English to German: breakfast =>
Frühstück
Completes a task without an explicit example
FEW-SHOT
gaot => goat
sheo => shoe
pohne =>
phone
Completes a task given a few examples of the task

Datasets used in GPT-3 pre-training:

Dataset name	Dataset description	Number of tokens	Proportion in training data
CommonCrawl (filtered)	Web crawl data	410 billion	60%
WebText2	Web crawl data	19 billion	22%
Books1	Internet-based book corpus	12 billion	8%
Books2	Internet-based book corpus	55 billion	8%
Wikipedia	High-quality text	3 billion	3%

- Subset of above data used , 300 billions tokens used out of 499 billions tokens. Authors of GPT3 paper did not mention the reason why.

- Size : 410 billions tokens → 570 GB, later Meta’s LLaMa include arxive research papers (92 GB) and stack exchnage code (78GB)
- GPT-3 did not share the dataset but comparitive data is shared by

  Dolma: An Open Corpus of Three Trillion Tokens for LLM Pretraining Research by Soldaini et al. 2024 (https://arxiv.org/abs/2402.00159).
- Pretraining GPT-3 like models is quite expensive , GPT-3 training cost 4.6 Billions dollars.
- Good news is that we have open source trained models, which could be fine-tuned with small labeled datasets.

A closer look at the GPT-3 architecture:

- paper by openAI: “Improving Language Understanding by Generative Pre-Training” (https://mng.bz/x2qg)
- ChatGPT: GPT-3 was fine tuned using the large structured data using the paper InstructGPT by open AI, paper : (https://arxiv.org/abs/2203.02155). for tasks such as spelling correction, classification, or language translation. This is actually very remarkable given that GPT models are pretrained on a relatively simple next-word prediction task.

> GPT-3 (Decoder only model) Since decoder-style models like GPT generate text by predicting text one word at a time, they are considered a type of autoregressive model. Autoregressive models incorporate their previous outputs as inputs for future predictions.

💡 Transformer repeat the layers of endcoder and decoder 6 times but GPT repeat the 96 transformers layers and has 175 billions parameters in total.

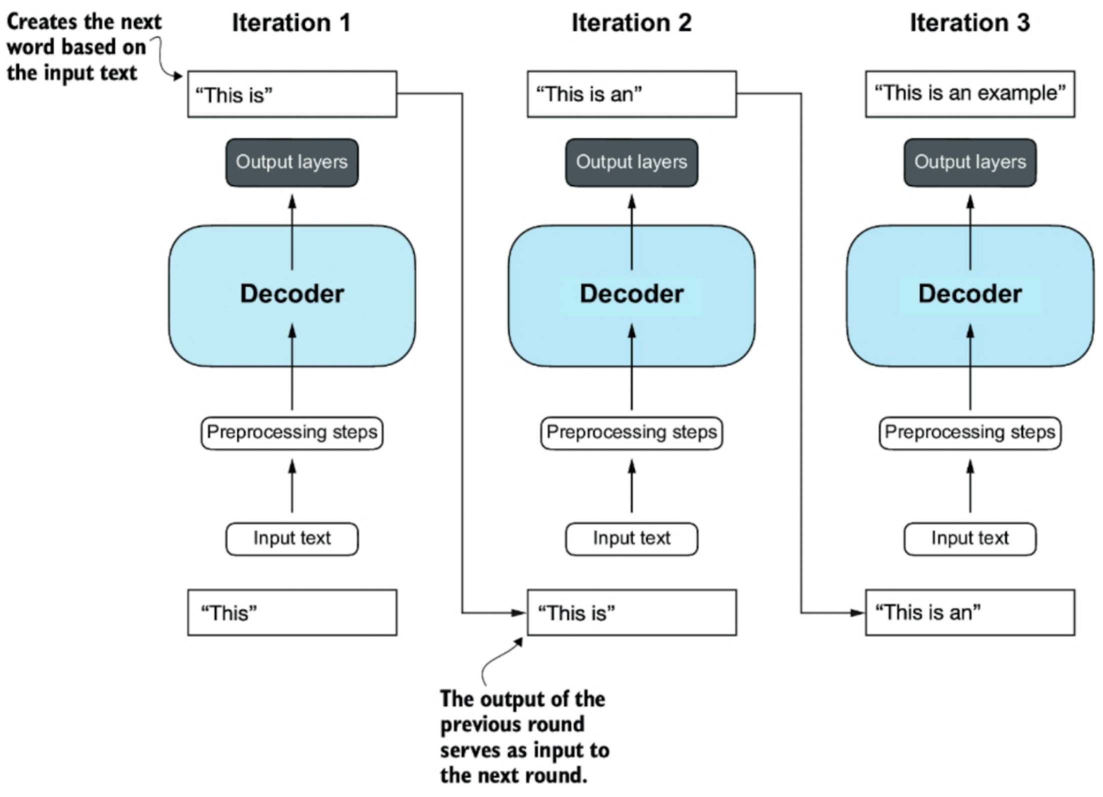


Figure 1.8 The GPT architecture employs only the decoder portion of the original transformer. It is designed for unidirectional, left-to-right processing, making it well suited for text generation and next-word prediction tasks to generate text in an iterative fashion, one word at a time.

💡 The ability to perform tasks that the model wasn't explicitly trained to perform is called an **emergent behavior**.

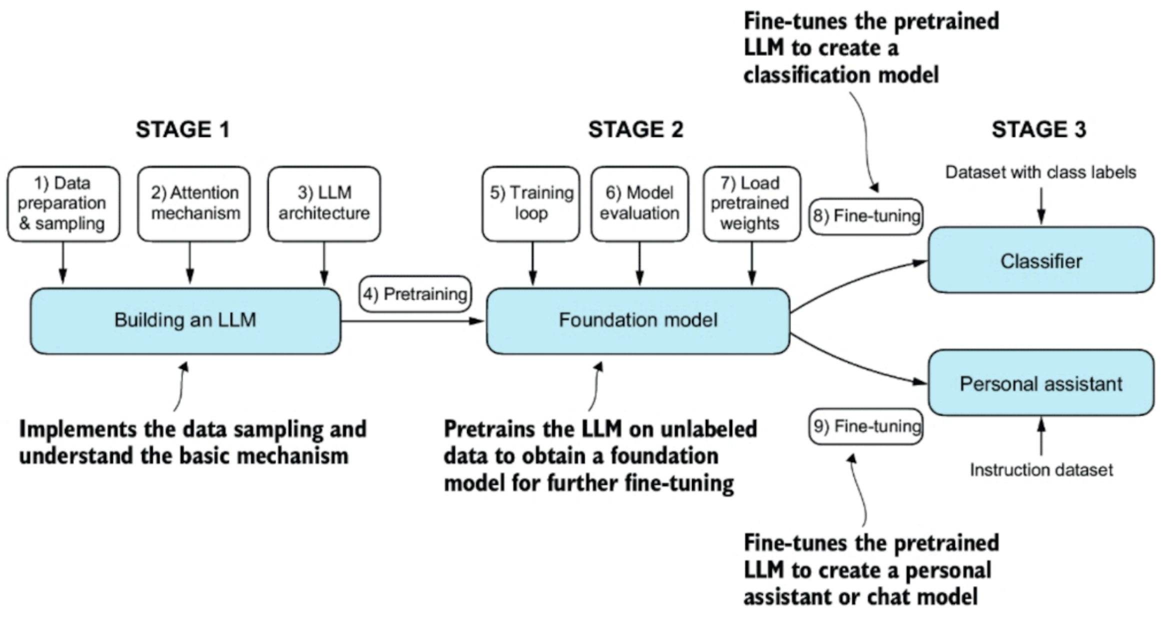

Fine-tunes the pretrained LLM to create a classification model
STAGE 1
STAGE 2
STAGE 3
1) Data preparation & sampling
2) Attention mechanism
3) LLM architecture
5) Training loop
6) Model evaluation
7) Load pretrained weights
8) Fine-tuning
Dataset with class labels
Classifier
Building an LLM
4) Pretraining
Foundation model
Personal assistant
9) Fine-tuning
Instruction dataset
Implements the data sampling and understand the basic mechanism
Pretrains the LLM on unlabeled data to obtain a foundation model for further fine-tuning
Fine-tunes the pretrained LLM to create a personal assistant or chat model

Figure 1.9 The three main stages of coding an LLM are implementing the LLM architecture and data preparation process (stage 1), pretraining an LLM to create a foundation model (stage 2), and fine-tuning the foundation model to become a personal assistant or text classifier (stage 3).

Summary

- Previously mostly relied on explicit rule-based systems and simpler statistical methods. The advent of LLMs introduced new deep learning-driven approaches that led to advancements in understanding, generating, and translating human language.
- Modern LLMs are trained in two main steps:
    - First, they are pretrained on a large corpus of unlabeled text by using the prediction of the next word in a sentence as a label.
    - Then, they are fine-tuned on a smaller, labeled target dataset to follow instructions or perform classification tasks.
- LLMs are based on the transformer architecture. The key idea of the transformer architecture is an attention mechanism that gives the LLM selective access to the whole input sequence when generating the output one word at a time.
- The original transformer architecture consists of an encoder for parsing text and a decoder for generating text.

- LLMs for generating text and following instructions, such as GPT-3 and ChatGPT, only implement decoder modules, simplifying the architecture.
- Large datasets consisting of billions of words are essential for pretraining LLMs.
- While the general pretraining task for GPT-like models is to predict the next word in a sentence, these LLMs exhibit emergent properties, such as capabilities to classify, translate, or summarize texts.
- Once an LLM is pretrained, the resulting foundation model can be fine-tuned more efficiently for various downstream tasks.
- LLMs fine-tuned on custom datasets can outperform general LLMs on specific tasks.

Raschka, Sebastian. Build a Large Language Model (From Scratch) (Function). Kindle Edition.

Chapter-1 -Code Setup