

In stage 1, we will learn about the fundamental data preprocessing steps and code the attention mechanism at the heart of every LLM. Next, in stage 2, we will learn how to code and pretrain a GPT-like LLM capable of generating new texts. We will also go over the fundamentals of evaluating LLMs, which is essential for developing capable NLP systems.

Pretraining an LLM from scratch is a significant endeavor, demanding thousands to millions of dollars in computing costs for GPT-like models. Therefore, the focus of stage 2 is on implementing training for educational purposes using a small dataset. In addition, I also provide code examples for loading openly available model weights.

Finally, in stage 3, we will take a pretrained LLM and fine-tune it to follow instructions such as answering queries or classifying texts—the most common tasks in many real-world applications and research.

I hope you are looking forward to embarking on this exciting journey!

Summary

- LLMs have transformed the field of natural language processing, which 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.



Working with text data

This chapter covers

- Preparing text for large language model training
- Splitting text into word and subword tokens
- Byte pair encoding as a more advanced way of tokenizing text
- Sampling training examples with a sliding window approach
- Converting tokens into vectors that feed into a large language model

So far, we've covered the general structure of large language models (LLMs) and learned that they are pretrained on vast amounts of text. Specifically, our focus was on decoder-only LLMs based on the transformer architecture, which underlies the models used in ChatGPT and other popular GPT-like LLMs.

During the pretraining stage, LLMs process text, one word at a time. Training LLMs with millions to billions of parameters using a next-word prediction task yields models with impressive capabilities. These models can then be further fine-tuned to follow general instructions or perform specific target tasks. But before we can implement and train LLMs, we need to prepare the training dataset, as illustrated in figure 2.1.