Large Language Models – Mô Hình Ngôn Ngữ Lớn

Nguyễn Quản Bá Hồng*

Ngày 14 tháng 3 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Large Language Models – Mô Hình Ngôn Ngữ Lớn.*
  PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/large_language_model/NQBH_large_language_model.pdf.
  TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/large_language_model/NQBH_large_language_model.tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Basics LLMs

1.1 [Ras24]. Sebastian Raschka. Build A Large Language Model (From Scratch)

[125 Amazon ratings]

- Amazon review. Learn how to create, train, & tweak LLMs by building 1 from ground up! In *Build a Large Language Model (from Scratch)* bestselling author Sebastian Raschka guides you step by step through creating your own LLM. Each stage is explained with clear text, diagrams, & examples. Go from initial design & creation, to pretraining on a general corpus, & on to fine-tuning for specific tasks.

  Build a Large Language Model (from Scratch) teaches how to:

  - Plan & code all parts of an LLM
  - Prepare a dataset suitable for LLM training
  - Fine-tune LLMs for text classification & with your own data
  - Use human feedback to ensure your LLM follows instructions
  - Load pretrained weights into an LLM

  Build a Large Language Model (from Scratch) takes you inside AI black box to tinker with internal systems that power generative AI. As work through each key stage of LLM creation, develop an in-depth understanding of how LLMs work, their limitations, & their customization methods. Your LLM can be developed on an ordinary laptop, & used as your own personal assistant.

  About technology. Physicist Richard P. Feynman reportedly said, "I don't understand anything I can't build." Based on this same powerful principle, bestselling author Sebastian Raschka guides you step by step as build a GPT-style LLM that can run on laptop. This is an engaging book that covers each stage of process, from planning & coding to training & fine-tuning.

*A Scientist & Creative Artist Wannabe. E-mail: nguyenquanbahong@gmail.com. Bến Tre City, Việt Nam.

About book. *Build a Large Language Model (From Scratch)* is a practical & eminently-satisfying hands-on journey into foundations of generative AI. Without relying on any existing LLM libraries, code a base model, evolve it into a text classifier, & ultimately create a chatbot that can follow your conversational instructions. & really understand it because built it yourself!

What's inside.

- Plan & code an LLM comparable to GPT-2
- Load pretrained weights
- Construct a complete training pipeline
- Fine-tune your LLM for text classification
- Develop LLMs that follow human instructions

- About reader. Readers need intermediate Python skills & some knowledge of ML. LLM you create will run on any modern laptop & can optionally utilize GPUs.

- About author. Sebastian Raschka is a Staff Research Engineer at Lighting AI, where he works on LLM research & develops an open-source software. The technical editor on this book was David Caswell.

- Editorial Reviews.
  - "The most comprehensive book I've seen on building LLMs. Highly recommended!" – Raul Ciotescu, CTO, Netzinkubator Software
  - "A clear, hands-on guide that empowers readers to build their own models & explore cutting edge of AI." – Guillermo Alcántara, Project manager, PepsiCo Global
  - "Must-have resource for quickly getting up to speed on LLMs. Whether you're new to field or looking to deepen your knowledge, it's perfect guide." – Walter Reade, Staff Developer Relations Engineer, Kaggle/Google
  - "A fantastic resource for diving into LLMs – a must-read for anyone eager to get hands-on!" – Dr. Vahid Mirjalili, Senior Data Scientist, FM Global

  Pipeline: 3 main stages of coding a LLM are implementing LLM architecture & data preparation process (stage 1), pretraining an LLM to create a foundation model (stage 2), & fine-tuning foundation model to become a personal assistant or text classifier (stage 3). Each of these stages is explored & implemented in this book.

- Preface. Always been fascinated with language models. > 1 decade ago, journey into AI began with a statistical pattern classification class, which led to 1st independent project: developing a model & web application to detect mood of a song based on its lyrics.

  Fast forward to 2022, with release of ChatGPT, LLMs have taken world by storm & have revolutionized how many of us work. These models are incredibly versatile, aiding in tasks e.g. checking grammar, composing emails, summarizing lengthy documents, & much more. This is owed to their ability to parse & generate human-like text, which is important in various fields, from customer service to content creation, & even in more technical domains like coding & data analysis.

  As their name implies, a hallmark of LLMs: they are "large" – very large – encompassing millions to billions of parameters. (For comparison, using more traditional ML or statistical methods, Iris flower dataset can be classified with > 90% accuracy using a small model with only 2 parameters.) However, despite large size of LLMs compared to more traditional methods, LLMs don't have to be a black box.

  In this book, will learn how to build an LLM 1 step at a time. By end, will have a solid understanding of how an LLM, like ones used in ChatGPT, works on a fundamental level. Believe: developing confidence with each part of fundamental concepts & underlying code is crucial for success. This not only helps in fixing bugs & improving performance but also enables experimentation with new ideas.

  Several years ago, when I started working with LLMs, had to learn how to implement them hard way, sifting through many research papers & incomplete code repositories to develop a general understanding. With this book, hope to make LLMs more accessible by developing & sharing a step-by-step implementation tutorial detailing all major components & developments phases of an LLM.

  Strongly believe: best way to understand LLMs: code one from scratch – & will see that this can be fun too! Happy reading & coding!

- Acknowledgments. Writing a book is a significant undertaking – Viết 1 cuốn sách là một công việc quan trọng.

- About this Book. *Build a Large Language Model (From Scratch)* was written to help you understand & create your own GPT-like LLMs from ground up. It begins by focusing on fundamentals of working with text data & coding attention mechanisms & then guides through implementing a complete GPT model from scratch. Book then covers pretraining mechanism as well as fine-tuning for specific tasks e.g. text classification & following instructions. By end of this book, have a deep understanding of how LLMs work & skills to build your own models. While models you'll create are smaller in scale compared to large foundational models, they use same concepts & serve as powerful educational tools to grasp core mechanisms & techniques used in building state-of-art LLMs.

- Who should read this book. *Build a Large Language Model (From Scratch)* is for ML enthusiasts, engineers, researchers, students, & practitioners who want to gain a deep understanding of how LLMs work & learn to build their own models from scratch. Both beginners & experienced developers will be able to use their existing skills & knowledge to grasp concepts & techniques used in creating LLMs.

  What sets this book apart is its comprehensive coverage of entire process of building LLMs, from working with datasets to implementing model architecture, pretraining on unlabeled data, & fine-tuning for specific tasks. As of this writing, no other resource provides such a complete & hands-on approach to building LLMs from ground up.

  To understand code examples in this book, should have a solid grasp of Python programming. While some familiarity with ML, DL, & AI can be beneficial, an extensive background in these areas is not required. LLMs are a unique subset of AI, so even if relatively new to field, will be able to follow along.

  If have some experience with deep neural networks, may find certain concepts more familiar, as LLMs are built upon these architectures. However, proficiency in PyTorch is not a prerequisite. Appendix A provides a concise introduction to PyTorch, equipping with necessary skills to comprehend code examples throughout book.

  A high school-level understanding of mathematics, particularly working with vectors & matrices, can be helpful as explore inner workings of LLMs. However, advanced mathematical knowledge is not necessary to grasp key concepts & ideas presented in this book.

  Most important prerequisite is a strong foundation in Python programming. With this knowledge, will be well prepared to explore fascinating world of LLMs & understand concepts & code examples presented in this book.

- How this book is organized: A roadmap. This book is designed to be read sequentially, as each chap builds upon concepts & techniques introduced in prev ones. Book is divided into 7 chaps that cover essential aspects of LLMs & their implementation.
  * Chap. 1 provides a high-level introduction to fundamental concepts behind LLMs. It explores transformer architecture, which forms basis for LLMs e.g. those used on ChatGPT platform.
  * Chap. 2 lays out a plan for building an LLM from scratch. It covers process of preparing text for LLM training, including splitting text into word & subword tokens, using byte pair encoding for advanced tokenization, sampling training examples with a sliding window approach, & converting tokens into vectors that feed into LLM.
  * Chap. 3 focuses on attention mechanisms used in LLMs. It introduces a basic self-attention framework & progresses to an enhanced self-attention mechanism. Chap also covers implementation of a causal attention module that enables LLMs to generate 1 token at a time, masking randomly selected attention weights with dropout to reduce overfitting & stacking multiple causal attention modules into a multihead attention module.
  * Chap. 4 focuses on coding a GPT-like LLM that can be trained to generate human-like text. It covers techniques e.g. normalizing layer activations to stabilize neural network training, adding shortcut connections in deep neural networks to train models more effectively, implementing transformer blocks to create GPT models of various sizes, & computing number of parameters & storage requirements of GPT models.
  * Chap. 5 implements pretraining process of LLMs. It covers computing training & validation set losses to assess quality of LLM-generated text, implementing a training function & pretraining LLM, saving & loading model weights to continue training an LLM, & loading pretrained weights form OpenAI.
  * Chap. 6 introduces different LLM fine-tuning approaches. It covers preparing a dataset for text classification, modifying a pretrained LLM for fine-tuning, fine-tuning an LLM to identify spam messages, & evaluating accuracy of a fine-tuned LLM classifier.
  * Chap. 7 explores instruction fine-tuning process of LLMs. It covers preparing a dataset for supervised instruction fine-tuning, organizing instruction data in training batches, loading a pretrained LLM & fine-tuning it to follow human instructions, extracting LLM-generated instruction responses for evaluation, & evaluating an instruction-fine-tuned LLM.

- About code.

- 1. Understanding LLMs.
  - 1.1. What is an LLM?
  - 1.2. Applications of LLMs.
  - 1.3. Stages of building & using LLMs.
  - 1.4. Introducing transformer architecture.
  - 1.5. utilizing large datasets.
  - 1.6. A closer look at GPT architecture.
  - 1.7. Building a large language model.
- 2. Working with text data.
  - 2.1. Understanding word embeddings.
  - 2.2. Tokenizing text.
  - 2.3. Converting tokens into token IDs.
  - 2.4. Adding special context tokens.

- 2.5. Byte pair encoding.
- 2.6. Data sampling with a sliding window.
- 2.7. Creating token embeddings.
- 2.8. Encoding word positions.

- 3. Coding attention mechanisms.
  - 3.1. Problem with modeling long sequences.
  - 3.2. Capturing data dependencies with attention mechanisms.
  - 3.3. Attending to different parts of input with self-attention.
    - A simple self-attention mechanism without trainable weights.
    - Computing attention weights for all input tokens.
  - 3.4. Implementing self-attention with trainable weights.
    - Computing attention weights step by step.
    - Implementing a compact self-attention Python class.
  - 3.5. Hiding future words with causal attention.
    - Applying a causal attention mask.
    - Masking additional attention weights with dropout.
    - Implementing a compact causal attention class.
  - 3.6. Extending single-head attention to multi-head.
    - Stacking multiple single-head attention layers.
    - Implementing multi-head attention with weight splits.

- 4. Implementing a GPT model from scratch to generate text.
  - 4.1. Coding an LLM architecture.
  - 4.2. Normalizing activations with layer normalization.
  - 4.3. Implementing a feed forward network with GELU activations.
  - 4.4. Adding shortcut connections.
  - 4.5. Connecting attention & linear layers in a transformer block.
  - 4.6. Coding GPT model.
  - 4.7. Generating text.

- 5. Pretraining on unlabeled data.
  - 5.1. Evaluating generative text models.
    - Using GPT to generate text.
    - Calculating text generation loss.
    - Calculating training & validation set losses.
  - 5.2. Training an LLM.
  - 5.3. Decoding strategies to control randomness.
    - Temperature scaling.
    - Top-k sampling.
    - Modifying text generation function.
  - 5.4. Loading & saving model weights in PyTorch.
  - 5.5. Loading pretrained weights from OpenAI.

- 6. Fine-tuning for classification.
  - 6.1. Different categories of fine-tuning.
  - 6.2. Preparing dataset.
  - 6.3. Creating data loaders.
  - 6.4. Initializing a model with pretrained weights.
  - 6.5. Adding a classification head.
  - 6.6. Calculating classification loss & accuracy.
  - 6.7. Fine-tuning model on supervised data.

- 6.8. Using LLM as a spam classifier.

- 7. Fine-tuning to follow instructions.
  - 7.1. Introduction to instruction fine-tuning.
  - 7.2. Preparing a dataset for supervised instruction fine-tuning.
  - 7.3. Organizing data into training batches.
  - 7.4. Creating data loaders for an instruction dataset.
  - 7.5. Loading a pretrained LLM.
  - 7.6. Fine-tuning LLM on instruction data.
  - 7.7. Extracting & saving responses.
  - 7.8. Evaluating fine-tuned LLM.
  - 7.9. Conclusions.
    - What's next?
    - Staying up to data in a fast-moving field.
    - Final words.

- A. Introduction to PyTorch.
- B. References & further reading.
- C. Exercise solutions.
- D. Adding bells & whistles to training loop.
- E. Parameter-efficient fine-tuning with LoRA.

2 Miscellaneous

Tài liệu

[Ras24] Sebastian Raschka. *Build A Large Language Model (From Scratch)*. 1st edition. Manning Publishing, 2024, p. 343.