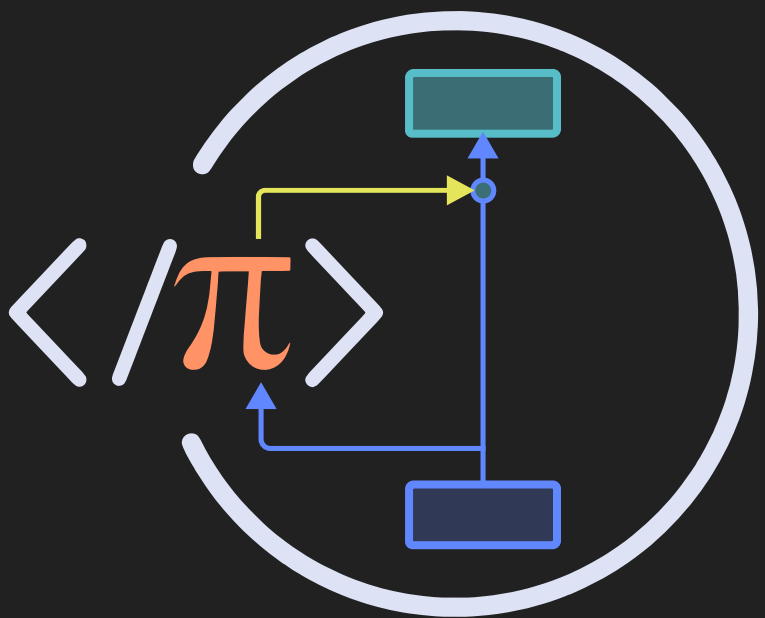
50 ML projects to understand LLMs



Investigate transformer mechanisms through data analysis, visualization, and experimentation

Mike X Cohen
SincXpress

0.1 Front matter

ISBN: 000, edition 1.
This book was written and formatted in LaTeX by Mike X Cohen.

0.2 Other books and courses by Dr. Mike X Cohen

I've written several other books, and authored 40-80-hour video-based courses on Udemy, on statistics, applied math for AI (linear algebra and calculus), and neuroscience data analysis.

Check out my website for resources, links, and discount coupons:
https://www.sincxpress.com.

0.3 Forward

The past is immutable and the present is fleeting. Forward is the only direction.

0.4 Dedication

If you're reading this, then the book is dedicated to you. I wrote this book for *you*. Now turn the page and start learning about LLMs!

Contents

Chapter 1

Introductions

1.1 Why study LLM mechanisms?

LLMs are among the most important, impactful, and cutting-edge technology in modern humanity. They are becoming increasingly common in every other piece of tech, and therefore are becoming increasingly integrated into every aspect of our professional, personal, and civic lives. It is difficult to imagine careers that AI will not impact.

Simultaneously, LLMs raise serious concerns about ethics, bias and fairness, errors in reasoning, hallucinations, and misuse (e.g., misinformation and disinformation). These concerns are exacerbated by modern LLMs being both literal and figurative "black boxes": Literal black boxes because many advanced AI systems are proprietary and the weights (trained parameters of the models) are not released to the public; and figurative black boxes because even the open-source AI models are so complicated that understanding them and developing safety guardrails has thus far proven extremely difficult.

Thus, learning about LLMs is important for several reasons, including career opportunities, improved customization, AI safety, and understanding developments in modern technology.

Mechanistic interpretability

The overarching goal of mechanistic interpretability is to understand how LLMs represent and transform sequences of text into coherent, capable, and human-like responses — basically, what's happening inside the LLM when you interact with chatbots like ChatGPT.

Shouldn't it be really easy to understand how LLMs work? After all, the equations, code, and all internal weights and activations of open-source models can be easily imported into Python.

The problem is the mind-bogglingly high-dimensional interactions across myriad computations inside the model. By analogy: Could you understand the plot of a movie by analyzing a vector of pixel brightness values?

In other words, the internal workings of LLMs are not immediately human-interpretable the way that a linear regression model ($y = mx + b$) is human-interpretable. Thus, the goal of interpretability is to discover pat-

terns in the complex, high-dimensional internal activations of LLMs that provide insights into how the models represent, process, and generate language.

Interpretability research has several applications, including the following.

Technical AI safety
: If an AI system is capable of harmful behavior or unethical decision-making, then those capabilities must be represented *somewhere* in the model. If it were possible to find the exact mechanism that allows a model to perform dangerous behavior, then that circuit could be removed before deployment. That may sound straightforward, but turns out to be extremely difficult in practice.

Improved fine-tuning
: Fine-tuning LLMs for specific applications can be time- and cost-intensive. A better understanding of internal mechanisms and representations of LLMs can lead to more efficient means of targeting the most relevant circuits in the model.

Complex systems research
: LLMs exhibit emergent behavior, meaning that simple components lead to complex patterns that cannot be predicted by examining the individual parts alone. Many biological and physical systems are complex and have emergent properties; LLMs are a fascinating model to study complex dynamics because they are human-crafted, accessible to scientific experimentation, and are increasingly pervasive in modern society.

Thus, there are many reasons to understand and interpret LLM mechanisms, ranging from technical to legal to practical to scientific to curiosity.

Will you learn everything about LLMs in this book?

Nope! The field of LLMs is large and ever-increasing. LLMs are very complex and there are new discoveries about how they work being made seemingly daily. Furthermore, not all LLMs have the same architecture, which means that even if you understood everything about GPT-2, you wouldn't automatically understand everything about BERT.

However, after working through this book, you will have the skills to learn

about and investigate different LLM architectures. That's my goal here: To teach you a core set of future-proof skills that provides a solid foundation to continue investigating LLMs and other deep-learning models.

1.2 Why use machine learning to learn about LLMs?

Machine learning (ML) is at the intersection of computer science and statistics, and involves developing and deploying algorithms that make predictions about the world by learning from examples (training data). The goal of ML is to discover patterns in datasets that can be used to predict or classify new data.

ML is a useful — and under-utilized — framework for studying LLMs. For one thing, LLMs are literally composed of simple ML algorithms (linear weighted averages and nonlinear transformations). Furthermore, using ML techniques like regression, classification, and clustering, can help reveal how concepts like grammar rules are represented inside LLMs. And finally, many people find LLMs to be intimidatingly complicated while finding ML to be much more approachable. Thus, using ML to study LLMs involves using simple tools to understand complicated tools.

My goal in this book is to give you a solid foundation upon which to improve your ML and Python coding skills, learn about LLM architecture and operations, and fast-track your future in AI research, development, or safety.

Therefore, I do not focus on current theories and nuanced interpretations of LLM mechanisms. Theories of LLMs are developing quickly — and are often quickly proven wrong. That's a good thing thing; it reflects a fast rate of progress. On the other hand, my goal in this book is to teach you transferable and future-proof skills and tools to investigate LLMs, rather than make bold interpretational claims that might be disproven by the time you read this.

1.3 Prerequisites

The obvious ML and LLM mechanisms are each challenging fields in their own right, and the combination of the two is even more difficult. I believe that having a goal and intention to learn about LLMs through ML is the single most important prerequisite for success. The points below are minor in comparison to the importance of having a reason to make even a small but consistent effort to learn a new skill.

Math You need to be comfortable with arithmetic and basic algebra. Having a linear algebra background (e.g., matrix multiplication, eigendecomposition, rank) is helpful though not necessary. Calculus is important for understanding how randomly initialized models learn from data (i.e. the gradient descent algorithm), but is not necessary for this book.

I will show and explain equations as the need arises in the projects, but I tried to make this book more of a hands-on practical approach towards studying LLMs through ML coding, as opposed to a math-dense tome with first-principles proofs.

Coding Python is currently the most important language for modern AI, and so this book uses Python exclusively.

I assume basic coding proficiency. If you're brand-new to Python, then you should probably wait to delve into this book until after taking an intro-Python course or book. There are myriad free and paid resources available for learning Python.

Before starting this book, you should be comfortable with topics including variable types, for-loops, and functions; and you should be familiar with the NumPy, Matplotlib, and PyTorch libraries. Other libraries like pandas, scipy, and statsmodels are used occasionally.

LLM architecture, training, and interpretability No prior background required! In fact, the purpose of this book is for you to learn about LLMs. Keep in mind, though, that all projects involve analyzing pretrained models. I don't discuss training, fine-tuning, or instruction tuning in this book.

Machine learning You do not need ML expertise to work through this book; I will explain all the analyses and Python implementations as they are needed in the projects. That said, having some exposure to ML or related disciplines like statistics or data science will be helpful. If you're comfortable with concepts like correlation, t-test, and clustering, then you know enough to work through this book.

1.4 Hardware and software requirements

GPU vs. CPU LLMs simply wouldn't exist without GPUs. But that doesn't mean you need top-of-the-line GPUs to learn about LLMs by investigating trained models.

Most of the projects in this book can be done on any CPU — even the CPU in your laptop. There are several projects where having GPU access is beneficial to decrease computation time from minutes to seconds, and to gain proficiency at transferring models and data between the GPU and the CPU. But those projects can still be completed with a CPU with some patience and possibly reducing the dataset sizes or using a smaller model.

When a GPU is particularly helpful, I'll mention it explicitly at the beginning of the project. If there is no text about GPU usage, then you can assume that a standard CPU is sufficient for that project.

Run the projects locally or on the cloud? I wrote all the code for this book in Google's Colab environment, which is a free cloud-based Python environment that runs interactive Jupyter notebooks. Colab is great for teaching and for learning, because you don't need to install anything; you can develop and run the code from any browser on any internet-connected computer; you don't need to worry about downloading or updating libraries; if library updates break my code, it's easy for me to update it for everyone.

Of course, it's also fine to develop and run the code on your computer using locally installed Python and libraries, and any IDE you prefer. That may require some additional effort and trouble-shooting, for example ensuring that your local Python library versions match those running on Google Colab. I will continue to ensure that my code runs in Colab, but I cannot

provide support for every possible local configuration and IDE.

Do you need a Pro subscription? Google offers free and paid-tier access to Colab. The paid plans offer longer connection times and access to better GPUs. You do not need to pay for upgrades to successfully complete all the projects in this book. That said, paid access can make your experience smoother and faster, for example if GPU access is temporarily blocked due to heavy usage.

Python and library versions As I wrote above, if library updates cause errors, I will update the code on the book's GitHub repository (link and description in Section 1.6). Thus, the best way to ensure that you can reproduce the solutions in the book is either to use Google Colab, or to check that the library versions on your local installation match those on Colab.

1.5 How to solve the projects

If you simply read this book without solving any projects, then sure, you'll learn something and I hope you find the book useful. But to gain real knowledge about ML and LLMs, and to gain flexible analytic skills that you can transfer to other applications, you need to solve projects.

I designed the projects to require some effort and creativity — and coding. They are opportunities for you to explore concepts, visualizations, and parameters in ways that are difficult or impossible to do without code.

Project "parts" Each project is focused on one theme, and is split into smaller and more focused parts. The parts are cumulative, so you cannot complete part 2 without having completed part 1.

Tasks and figures The text for each project introduces background, provides instructions, shows figures (data visualizations) for you to reproduce, and provides interpretations.

When you see a task box like this:

Task

Here are specific instructions for you to follow.

Then you should read the task instructions and look at any referenced figures, and then switch to Python and start coding. Most instructions produce Python outputs or data visualizations that you can compare against the figures in the book.

Two important things to keep in mind when reproducing the figures in the book:

1. Some analyses involve generating random numbers or randomly shuffling data. In these cases, you won't reproduce the book figure *exactly* but yours should show the same qualitative patterns.
2. Don't worry about matching the aesthetics of my figures, such as line colors, marker shapes, axis tick marks, and so on. Focus more on the analyses and the content of the visualizations.

Adjustable difficulty levels

There are several ways to solve the projects, depending on your background level of coding and ML experience, and depending on how much time you choose to invest into this book.

Solve from scratch

This is the most challenging and most time-consuming way to engage with this book — but offers the most rewarding and deepest educational experience. Here's what to do: Read the text for instructions and look at the figures. Then open a blank Python notebook and start coding from scratch. You can check your results against my solutions. Keep in mind that there are many correct ways to solve the projects; if your figures match those in the book, then it doesn't matter if your solution is different from mine.

Start from the "helper" notebooks

Each project comes with two notebook files: a "helper" and a "solution." The helper file has missing or partially completed lines of code. You can solve the project by completing the code. This is a medium level of difficulty. The helper files on in the book's GitHub repository (see Section 1.6).

Run the solutions notebooks

The solutions files are completed code that produce the figures in the book. You can run through those files one code-block at a time. But don't just run the code — look through each line to understand what it's doing, change the code to see how that affects the results, and so on.

However you choose to engage with the projects, feel free to look at my code solutions as much as you like. It's not cheating! This is especially true if you are new to Python, and if you understand the concepts but struggle with the coding syntax.

Note: Only the most important snippets of code are printed in the book. All the code files to solve the projects and produce the book figures are on the book's GitHub repository.

Do you need to solve all the projects in order?

When creating these projects, I tried to find a balance between making the projects independent of each other such that you can work on any project in any order, vs. reducing redundancies when explaining LLM and ML concepts that were introduced in previous projects. For example, Chapter 5 involves calculating cosine similarity matrices, which I explain in Chapter 3.

Thus, I do recommend solving the projects in order. But it's fine to skip projects if you have limited time and want to focus on specific aspects of LLM processing. When there are prerequisite dependencies, I state this explicitly in the project description (along the lines of "you need to have solved project x before starting this one").

Questions? Join the Discord discussion

I have a Discord server with a dedicated channel for this book. It's free to join; the invitation link is on the book's GitHub repository (link in the next section).

You can join the server to discuss project solutions or find "accountability partners" to work through the book and solutions together with other readers.

1.6 Getting and using the book code

The code for this book is available at
https://github.com/mikexcohen/ML4LLM_book
If you are comfortable with git, then you can clone this repository to sync the files locally. If you have no idea what that previous sentence means, then don't worry! You can get all the code without knowing anything about git, and without needing to log in, sign up, give an email address, pay, or anything else.

Simply go to that URL, look for the green button that says "Code", click on that button, and look for the link that says "Download zip." This will download one zip file that contains all the code material that you will need for this entire book. Unpack that zip file on your computer.

Then, upload the files to drive.google.com. Once the notebook files are in your Google Drive, you can open them using Colab.

Modifying my code Yes, please do! I very much hope that you see my code not as immutable text, but instead as a source of inspiration for you to modify, adapt, and explore.

Reposting my code I am occasionally asked whether I allow people to post my code, or modifications to it, on websites, blogs, GitHub, or the like. My answer is *yes!* Absolutely, please feel free to use and share my code as you like. I ask only that you cite the source at the top of the code file, including the url to the GitHub repo. Something like this:

```
This code is modified from Mike X Cohen's book
"50 ML projects to understand LLMs"
https://github.com/mikexcohen/ML4LLM_book
```

1.7 AI assistance

I find that the more time goes on, the less I use AI for coding and writing. I am not morally opposed to AI-assistance, nor am I intellectually superior