Notes on Chip Huyen's book *AI Engineering*

notes by Jacob Williams • last updated 2025-04-20

"Chapter 1. Introduction to Building AI Applications with Foundation Models"

- "Eloundou et al. (2023) has excellent research on how exposed different occupations are to AI."
- author's large list of AI tools
- "I find it strange that ads are personalized because we know everyone is different, but education is not."
- "In research, people have also found that they can use a group of conversational bots to simulate a society, enabling them to conduct studies on social dynamics (Park et al., 2023)."
- Apple's guidance on using ML

- Microsoft's crawl-walk-run approach
- On how harder it is to achieve fully satisfactory results than just initially promising ones: https://arxiv.org/abs/2305.14233, https://www.linkedin.com/blog/engineering/generative-ai/musings-on-building-a-generative-ai-product

Even when building on foundation models, your startup's advantage could be data:

> Big companies likely have more existing data. However, if a startup can get to market first and gather sufficient usage data to continually improve their products, data will be their moat. Even for the scenarios where user data can't be used to train models directly, usage information can give invaluable insights into user behaviors and product shortcomings, which can be used to guide the data collection and training process

"Chapter 2. Understanding Foundation Models"

- Chinchilla scaling law: "They found that for compute-optimal training, you need the number of training tokens to be approximately 20 times the model size."
- There's a section on the difficulty of hyperparameter tuning given the infeasibility of doing many training runs. Would be interesting to read more about this. Some links: https://lukemetz.com/difficulty-of-extrapolation-nn-scaling/, https://arxiv.org/abs/2206.07682, https://x.com/jaschasd/status/1756930242965606582
- "The rate of training dataset size growth is much faster than the rate of new data being generated (Villalobos et al., 2022)…"
- "Longpre et al. (2024) observed that between 2023 and 2024, the rapid crescendo of data restrictions from web sources rendered over 28% of the most critical sources in the popular public dataset C4 fully restricted from use. Due to changes in its Terms of Service and crawling restrictions, a full 45% of C4 is now restricted."
- "Techniques for preference finetuning include *reinforcement learning from human feedback* (RLHF) (used by GPT-3.5 and Llama 2), DPO (Direct Preference Optimization) (used by Llama 3), and *reinforcement learning from AI feedback* (RLAIF) (potentially used by Claude)."
- Neat-looking nonprofit: https://laion.ai/
- "Some companies find it okay to skip reinforcement learning altogether. For example, Stitch Fix and Grab find that having the reward model alone is good enough for their applications. They get their models to generate multiple outputs and pick the ones given

high scores by their reward models. This approach, often referred to as the *best of N* strategy, leverages how a model samples outputs to improve its performance."

- "…you can use beam search to generate a fixed number of most promising candidates (the beam) at each step of sequence generation."
- "OpenAI also trained verifiers to help their models pick the best solutions to math problems (Cobbe et al., 2021). They found that using a verifier significantly boosted the model performance. *In fact, the use of verifiers resulted in approximately the same performance boost as a 30× model size increase.*"
- "There are currently two hypotheses about why language models hallucinate."
    - "self-delusion" https://arxiv.org/abs/2110.10819
    - "mismatch between the model's internal knowledge and the labeler's internal knowledge" https://www.alignmentforum.org/posts/BgoKdAzogxmgkuuAt/behavior-cloning-is-miscalibrated, https://www.youtube.com/watch?v=hhiLw5Q_UFg

"Chapter 3. Evaluation Methodology"

- "…investment in evaluation lags behind other areas in the AI space…"
- some interconvertible language model metrics: cross-entropy, perplexity, bits-per-character, bits-per-byte
- higher perplexity = "the more uncertainty the model has in predicting what comes next in a given dataset"
- "Perplexity might not be a great proxy to evaluate models that have been post-trained using techniques like SFT and RLHF. … A language model's perplexity typically increases after post-training."
- "…quantization…can also change a model's perplexity in unexpected ways."
- "…if a model's perplexity on a benchmark's data is low, this benchmark was likely included in the model's training data…"
- Joint embedding models: CLIP, ULIP, ImageBind
- some advice on having models perform scoring:
    - "It's been reported that AI judges work better with classification than with numerical scoring systems."

    - “For numerical scoring systems, discrete scoring seems to work better than continuous scoring. Empirically, the wider the range for discrete scoring, the worse the model seems to get. Typical discrete scoring systems are between 1 and 5.”
    - “If you use a scoring system between 1 and 5, include examples of what a response with a score of 1, 2, 3, 4, or 5 looks like, and if possible, why a response receives a certain score.”
- “Many teams use AI judges as guardrails in production to reduce risks, showing users only generated responses deemed good by the AI judge.”
- “In some cases, evaluation can take up the majority of the budget, even more than response generation.”
- biases some AI models have while judging: self-bias, first-position bias, verbosity bias
- “…you may use a cheap in-house model to generate responses and GPT-4 to evaluate 1% of the responses.”
- “While no one has admitted to me that they tried to game the [Chatbot arena] ranking, several model developers have told me that they’re convinced their competitors try to game it.”
- “Comparative evaluation is relatively hard to game, as there’s no easy way to cheat, like training your model on reference data. For this reason, many trust the results of public comparative leaderboards more than any other public leaderboards.”

“Chapter 4. Evaluate AI Systems”

- “**evaluation-driven development** means defining evaluation criteria before building” (emphasis added)
- “I believe that evaluation is the biggest bottleneck to AI adoption.”
- Nice breakdown and example of evaluation metrics: “Imagine you ask a model to summarize a legal contract. At a high level, **domain-specific capability** metrics tell you how good the model is at understanding legal contracts. **Generation capability** metrics measure how coherent or faithful the summary is. **Instruction-following capability** determines whether the summary is in the requested format, such as meeting your length constraints. **Cost and latency** metrics tell you how much this summary will cost you and how long you will have to wait for it.” (emphasis added)
- “What Evidence Do Language Models Find Convincing?”“Long-Form Factuality in Large Language Models”

- “…studies (Feng et al., 2023; Motoki et al., 2023; and Hartman et al., 2023) have shown that models, depending on their training, can be imbued with political biases. For example, OpenAI’s GPT-4 is more left-winged and libertarian-leaning, whereas Meta’s Llama is more authoritarian…”
- instruction-following capability evaluations: IFEval and InfoBench
- “If you use an open source model that infringes on copyrights, the infringed party is unlikely to go after the model developers, and more likely to go after you. However, if you use a commercial model, the contracts you sign with the model providers can potentially protect you from data lineage risks.”
- “Another reason that might cause open source models to lag behind is that open source developers don’t receive feedback from users to improve their models, the way commercial models do.”
- “Some leaderboards might exclude an important but expensive benchmark. For example, HELM (Holistic Evaluation of Language Models) Lite left out an information retrieval benchmark (MS MARCO, Microsoft Machine Reading Comprehension) because it’s expensive to run. Hugging Face opted out of HumanEval due to its large compute requirements—you need to generate a lot of completions.”
- WinoGrande is a fascinating benchmark - “solve challenging pronoun resolution problems”
- LOL: “Pretraining on the Test Set Is All You Need”
- detecting data contamination: n-gram overlapping (“impossible without access to the training data”), perplexity (“less accurate but much less resource-intensive”)
- To see if you have an adequate number of samples to do evaluation, try making multiple ‘bootstraps’ which are sample sets of the same size as the original but selected from it with replacement, and see if performance is similar across these. Also, see OpenAI’s guidance on how many samples you need

“Chapter 5. Prompt Engineering”

- “Prompt experiments should be conducted with the same rigor as any ML experiment, with systematic experimentation and evaluation.”
- “Most models, including GPT-4, empirically perform better when the task description is at the beginning of the prompt. However, some models, including Llama 3, seem to perform better when the task description is at the end of the prompt.”

- “The model might have been post-trained to pay more attention to the system prompt, as shared in the OpenAI paper “The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions” (Wallace et al., 2024). Training a model to prioritize system prompts also helps mitigate prompt attacks…”
- “Tools that aim to automate the whole prompt engineering workflow include OpenPrompt (Ding et al., 2021) and DSPy (Khattab et al., 2023). At a high level, you specify the input and output formats, evaluation metrics, and evaluation data for your task. These prompt optimization tools automatically find a prompt or a chain of prompts that maximizes the evaluation metrics on the evaluation data. Functionally, these tools are similar to autoML (automated ML) tools that automatically find the optimal hyperparameters for classical ML models.”
- “DeepMind’s Promptbreeder (Fernando et al., 2023) and Stanford’s TextGrad (Yuksekgonul et al., 2024) are two examples of AI-powered prompt optimization tools.”
- “Several tools have proposed special .prompt file formats to store prompts. See Google Firebase’s Dotprompt, Humanloop, Continue Dev, and Promptfile.”
- “Let’s say you trick a model into spitting out what looks like its system prompt. How do you verify that this is legitimate? More often than not, the extracted prompt is hallucinated by the model.”
- A couple types of indirect prompt injection attacks
    - “Passive phishing”, e.g.: “Imagine an attacker inserts code to install malware into an innocuous-looking public GitHub repository. If you use an AI model to help you write code, and this model leverages web search to find relevant snippets, it might discover this repository.”
    - “Active injection”, e.g.: “Imagine that you use a personal assistant to read and summarize emails for you. An attacker can send you an email with malicious instructions. When the assistant reads this email, it can confuse these injected instructions with your legitimate instructions.”
- “There are benchmarks that help you evaluate how robust a system is against adversarial attacks, such as Advbench (Chen et al., 2022) and PromptRobust (Zhu et al., 2023). Tools that help automate security probing include Azure/PyRIT, leondz/garak, greshake/llm-security, and CHATS-lab/persuasive_jailbreaker.”
- “Microsoft has a great write-up on how to plan red teaming for LLMs.”

“Chapter 6. RAG and Agents”

- term-based retrieval (e.g. Elasticsearch) vs embedding-based retrieval

- “Due to the importance of vector search, many algorithms and libraries have been developed for it. Some popular vector search libraries are *FAISS* (Facebook AI Similarity Search) (Johnson et al., 2017), Google’s *ScaNN* (Scalable Nearest Neighbors) (Sun et al., 2020), Spotify’s *Annoy* (Bernhardsson, 2013), and *Hnswlib* (Hierarchical Navigable Small World) (Malkov and Yashunin, 2016).”
- Zilliz series on vector search https://zilliz.com/learn/vector-index
- “The MTEB benchmark (Muennighoff et al., 2023) evaluates embeddings for a broad range of tasks including retrievals, classification, and clustering.”
- “The ANN-Benchmarks website compares different ANN algorithms on multiple datasets using four main metrics, taking into account the trade-offs between indexing and querying.”
- “Different algorithms can be used in sequence. First, a cheap, less precise retriever, such as a term-based system, fetches candidates. Then, a more precise but more expensive mechanism, such as k-nearest neighbors, finds the best of these candidates. This second step is also called *reranking*.”
- agents should plan first, have that plan validated (maybe by an AI judge), then execute
- “An open question is how well foundation models can plan. Many researchers believe that foundation models, at least those built on top of autoregressive language models, cannot. Meta’s Chief AI Scientist Yann LeCun states unequivocally that autoregressive LLMs can’t plan (2023). In the article “Can LLMs Really Reason and Plan?” Kambhampati (2023) argues that LLMs are great at extracting knowledge but not planning.”
- “Even if AI can’t plan, it can still be a part of a planner. It might be possible to augment an LLM with a search tool and state tracking system to help it plan.”
- “I suspect that in the long run, FM agents and RL agents will merge.”
- ReAct and Reflexion - approaches that iteratively try things and reflect on results, contrast with rigid plan-following

“Chapter 7. Finetuning”

- “Similarly, with supervised finetuning, you can also finetune a model to predict the next token or fill in the blank. The latter, also known as *infilling finetuning*, is especially useful for tasks such as text editing and code debugging. You can finetune a model for infilling even if it was pre-trained autoregressively.”

- “You can finetune a big model to make it even better, but finetuning smaller models is much more common. Smaller models require less memory, and, therefore, are easier to finetune. They are also cheaper and faster to use in production.”
- “Both finetuning and prompting experiments require systematic processes. Doing prompt experiments enables developers to build an evaluation pipeline, data annotation guideline, and experiment tracking practices that will be stepping stones for finetuning.”
- “One benefit of finetuning, before prompt caching was introduced, was that it can help optimize token usage. … With prompt caching, where repetitive prompt segments can be cached for reuse, this is no longer a strong benefit. … However, the number of examples you can use with a prompt is still limited by the maximum context length. With finetuning, there’s no limit to how many examples you can use.”
- “…*while finetuning can enhance a model’s performance on a specific task, it may also lead to a decline in performance in other areas.*”
- optimizers affect the memory requirements for training: “An Adam optimizer stores two values per trainable parameter.”
- “While FP64 is still used in many computations—as of this writing, FP64 is the default format for NumPy and pandas—it’s rarely used in neural networks because of its memory footprint. FP32 and FP16 are more common. Other popular floating point formats in AI workloads include *BF16* (BFloat16) and *TF32* (TensorFloat-32). BF16 was designed by Google to optimize AI performance on TPUs and TF32 was designed by NVIDIA for GPUs.”
- “Strictly speaking, it’s quantization only if the target format is integer. However, in practice, quantization is used to refer to all techniques that convert values to a lower-precision format.”
- “Weight quantization is more common than activation quantization, since weight activation tends to have a more stable impact on performance with less accuracy loss.”
- PEFT = parameter-efficient finetuning: “by inserting additional parameters into the model in the right places, you can achieve strong finetuning performance using a small number of trainable parameters”. That’s an “adapter-based” or “additive” method of PEFT; there are also “soft prompt-based methods [which] modify how the model processes the input by introducing special trainable tokens”.
- “Rather than trying to reduce LoRA’s number of parameters, you can reduce the memory usage more effectively by quantizing the model’s weights, activations, and/or gradients during finetuning. An early promising quantized version of LoRA is QLoRA (Dettmers et al., 2023).”
- multiple approaches to model merging

- “While you can linearly combine any set of models, *linear combination is the most effective for models finetuned on top of the same base model.* In this case, linear combination can be viewed through the concept of *task vectors*. The idea is that once you’ve finetuned a model for a specific task, subtracting the base model from it should give you a vector that captures the essence of the task.”
- you can follow a “progression path” or a “distillation path” when finetuning
- “Depending on the base model and the task, full finetuning typically requires at least thousands of examples and often many more. PEFT methods, however, can show good performance with a much smaller dataset. If you have a small dataset, such as a few hundred examples, full finetuning might not outperform LoRA.”

“Chapter 8. Dataset Engineering”

- “data-centric AI” vs “model-centric AI”
- single-turn vs multi-turn data - “For instance, when given a query, a model may need to first clarify the user’s intent before addressing the task.”
- quality, coverage, quantity
- “In short, if you have a small amount of data, you might want to use PEFT methods on more advanced models. If you have a large amount of data, use full finetuning with smaller models.”
- “If no improvement is observed with small data, a bigger dataset will rarely do the trick.”
- “However, be careful before concluding that finetuning with a small dataset doesn’t improve a model. Many things, other than data, can impact finetuning’s results, such as the choice of hyperparameters (e.g., the learning rate is too high or too low), data quality, poorly crafted prompts, etc. *In the vast majority of cases, you should see improvements after finetuning with 50–100 examples.*”
- data augmentation vs synthesis: the distinction is derived-from-real vs not
- “*Sometimes, humans might have fundamental limitations that cause human-generated data to be of lower quality than AI-generated data.* … Another example is in generating complex math problems—AI can generate questions that are far more complex than what an average human expert might conceive.”
- “The level of detail in the Llama 3 paper (Dubey et al., 2024) makes it an excellent case study for instruction data synthesis.”
- “…if you want synthetic data to mimic real data, its quality can be measured by how difficult it is to distinguish between the two. You could train an AI content detector to

identify AI-generated data—if it's easy to differentiate between real and synthetic data, the synthetic data isn't good."

- "In every project I've worked on, *staring at data for just 15 minutes usually gives me some insight that could save me hours of headaches.*"

"Chapter 9. Inference Optimization"

- "Anecdotally, I find that people coming from a system background (e.g., optimization engineers and GPU engineers) use *memory-bound* to refer to *bandwidth-bound*, and people coming from an AI background (e.g., ML and AI engineers) use to memory-bound to refer to memory capacity-bound."
- "…inference for image generators like Stable Diffusion is typically compute-bound, whereas inference for autoregression language models is typically memory bandwidth-bound."
- "An NVIDIA H100 running at its peak for a year consumes approximately 7,000 kWh. For comparison, the average US household's annual electricity consumption is 10,000 kWh."
- pruning
- speculative decoding; related: inference with reference
- parallel decoding, e.g. Medusa, Jacobi decoding
- KV cache, and techniques for reducing its memory requirements
- optimized kernels
- static, dynamic, and continuous / in-flight batching
- prompt caching
- replica and pipeline parallelism

"Chapter 10. AI Engineering Architecture And User Feedback"

- model routers & model gateways
- exact vs semantic caching
- "Many teams train a classifier to predict whether a query should be cached."

- "*Compared to other caching techniques, semantic caching's value is more dubious because many of its components are prone to failure.*"

- "…since user feedback is a crucial source of data for continuously improving AI models, more AI engineers are now becoming involved in the process to ensure they receive the data they need. …compared to traditional ML engineering, AI engineering is moving closer to product. This is because of both the increasing importance of data flywheel and product experience as competitive advantages."

The first part of the chapter incrementally builds up to this overall architecture diagram (this is copied from the book):