AI engineering

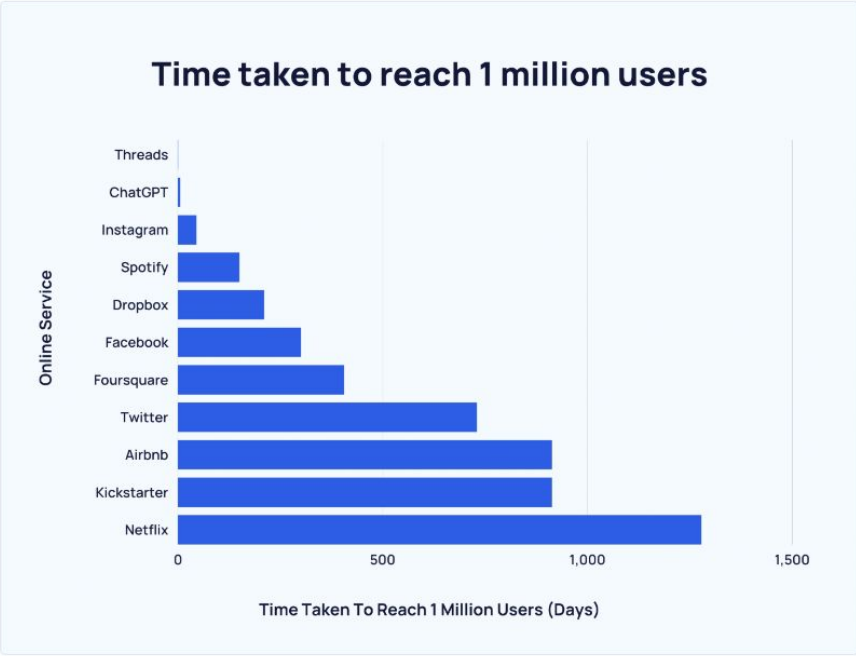
Columbia University
Fall 2025

Introduction to the course

The golden era of AI

It’s a fantastic moment to be in AI

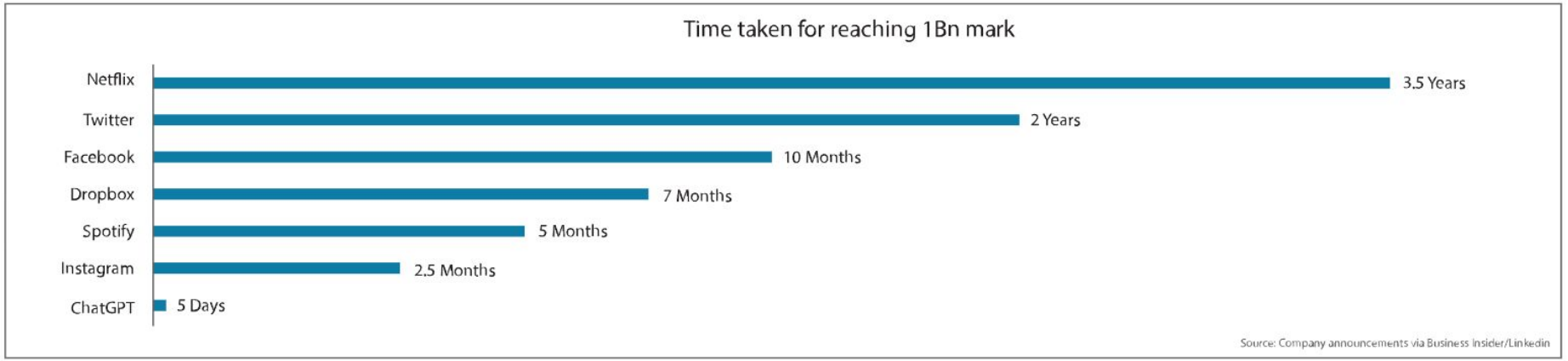
- ChatGPT usage: In April 2025, ChatGPT had approximately 800 million weekly users, with users sending 2.5 billion prompts per day by mid-2025.
- Generative AI impact: It’s expected to generate US $1.3 trillion in global economic impact annually by 2030



It’s a fantastic moment to be in AI

- **AI market growth**
  Expected to rise from US $294 billion (2025) to US $1.77 trillion by 2032, with a 29.2% CAGR.

- **AI market forecast**
  Another credible estimate projects AI growing from US $197 billion (2023) to around US $1.81 trillion by 2030, with a 36.6% CAGR.



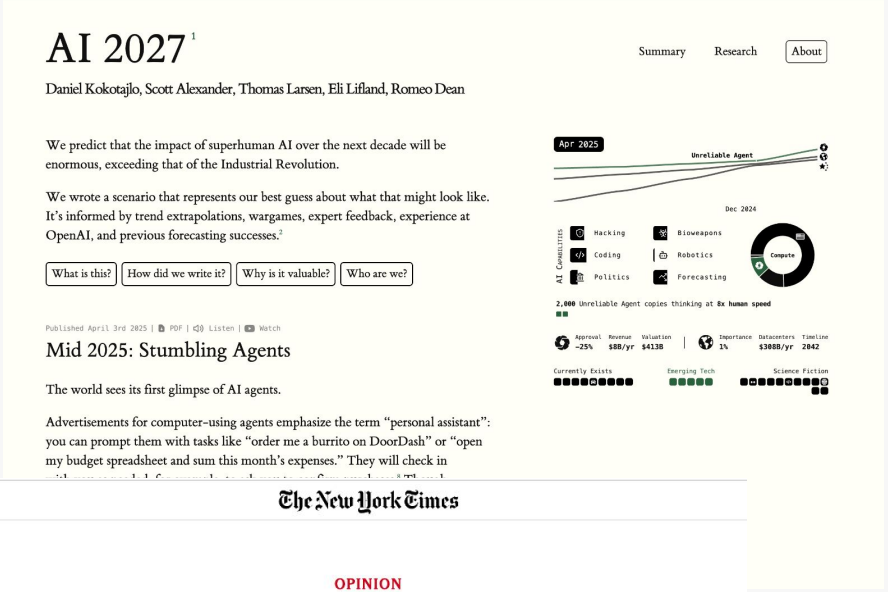
It’s a fantastic moment to be in AI

- GDP boost potential: AI could increase global GDP by up to 14% by 2030, representing an additional ~US $15.7 trillion.

The future is…we don’t really know, actually

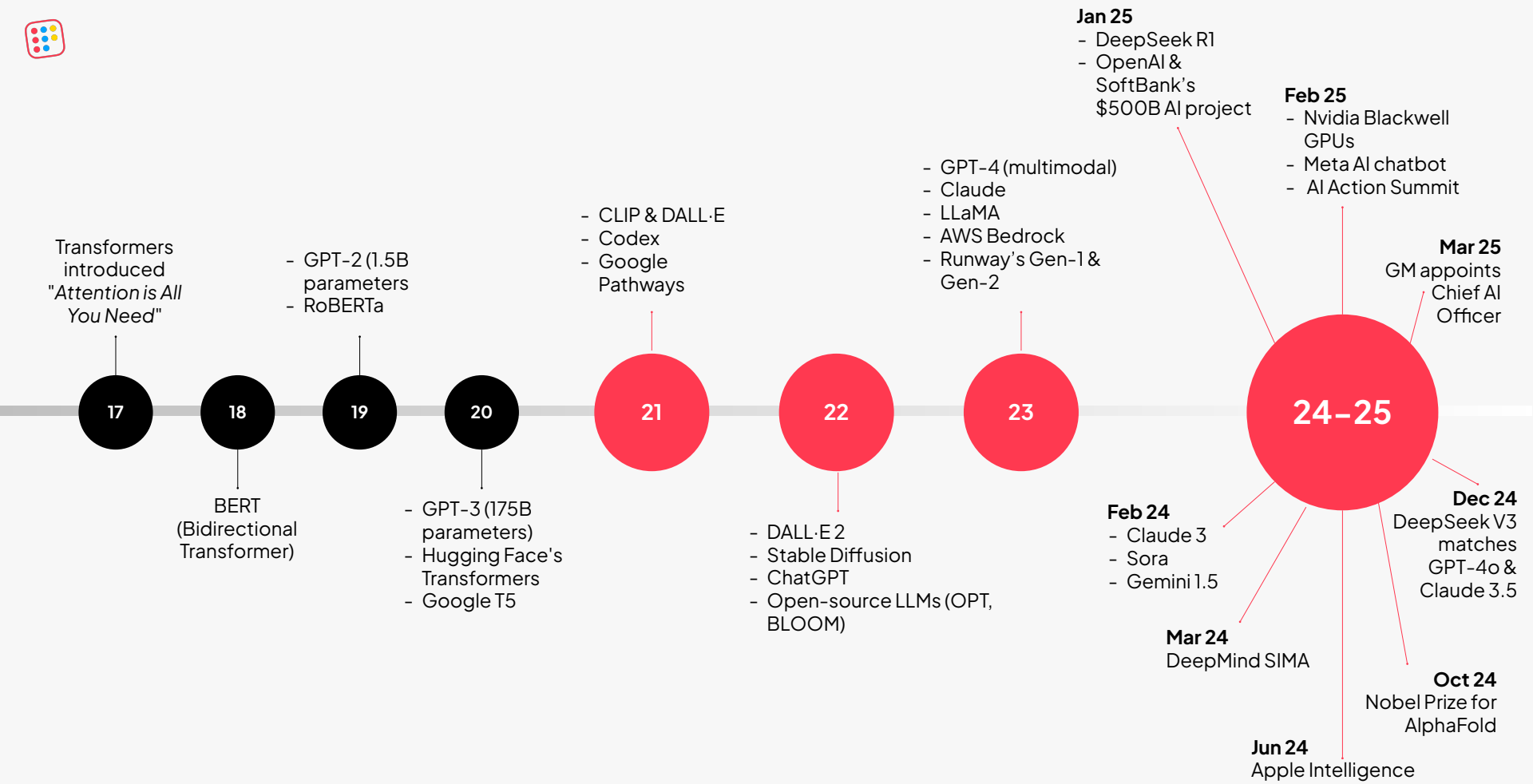
- AI 2027 projects explosive AI progress, culminating in AGI by late 2027:
  - Mid-2025: “Stumbling agents” begin performing simple digital tasks (scheduling, web automation)
  - 2026-2027: Emergence of “Neuralese” (latent reasoning language), enabling rapid self-improvement and shared knowledge across AI agents
  - Late 2027: Deployment of superhuman agents (e.g., “Agent-3-mini”) widely disrupts labor markets; AGI capabilities surpass human cognitive reach

- However, AI researchers estimate only a 10% chance of machines outperforming humans in all tasks by 2027, and 50% by 2047.



Large Language Models

- LLMs = very large deep learning models that are pre-trained on vast amounts of data to predict next tokens in a sequence.
- Based on transformer architecture (Vaswani et al., 2017).
- Capabilities: reasoning, summarization, translation, code generation, tool use.
- Examples: GPT-4, Claude, Gemini, LLaMA.

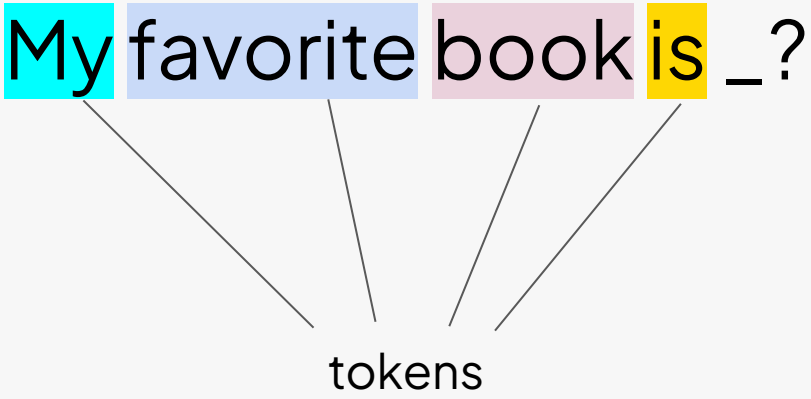

17
Transformers introduced "*Attention is All You Need*"
18
BERT (Bidirectional Transformer)
19
- GPT-2 (1.5B parameters
- RoBERTa
20
- GPT-3 (175B parameters)
- Hugging Face's Transformers
- Google T5
21
- CLIP & DALL·E
- Codex
- Google Pathways
22
- DALL·E 2
- Stable Diffusion
- ChatGPT
- Open-source LLMs (OPT, BLOOM)
23
- GPT-4 (multimodal)
- Claude
- LLaMA
- AWS Bedrock
- Runway's Gen-1 & Gen-2
24–25
Jan 25
- DeepSeek R1
- OpenAI & SoftBank's $500B AI project
Feb 25
- Nvidia Blackwell GPUs
- Meta AI chatbot
- AI Action Summit
Mar 25
GM appoints Chief AI Officer
Dec 24
DeepSeek V3 matches GPT-4o & Claude 3.5
Oct 24
Nobel Prize for AlphaFold
Jun 24
Apple Intelligence
Mar 24
DeepMind SIMA
Feb 24
- Claude 3
- Sora
- Gemini 1.5

The rise of AI engineering

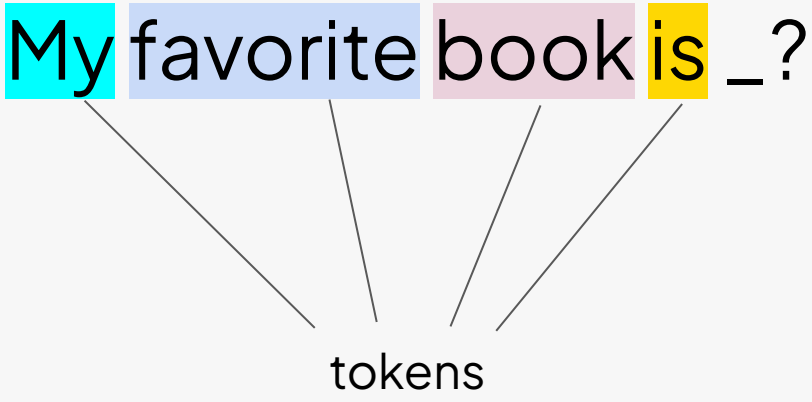
- Shift from AI research to AI engineering: building reliable, scalable applications
- Focus areas: prompt/data pipelines, cloud deployment, monitoring (cost, latency, safety)
- AI engineering is to ML research what software engineering is to computer science

From LMs to LLMs

Language models are completion machines



Language models are completion machines

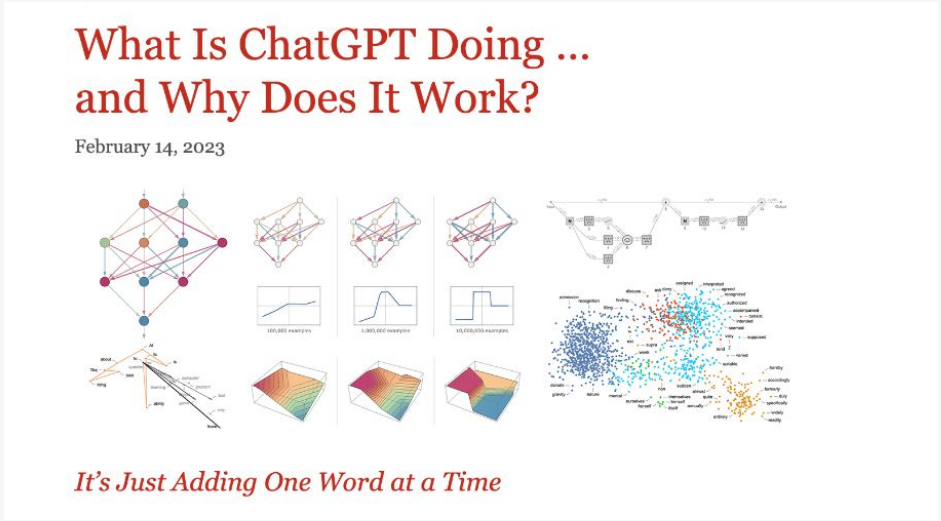


Moby Dick	4.1%
War and Peace	3.9%
the one	1.4%
written	2.2%

Pick on of these, but with a bit of randomness

Language models are completion machines

- Both traditional LMs and today's LLMs do the same core thing: **predict the next token given context**.
- More specifically: **decode by sampling** or **taking argmax** to complete the sequence.



Find the best parameters

- We want the parameters that maximize the likelihood of the training data.
- Equivalent to “make the model assign high probability to real text.”

Why log? Because probabilities are small and multiplying them becomes unstable. Log turns products into sums, making optimization tractable.

$$\text{LM/LLM objective: } \max_{\theta} \sum_{t} \log p_\theta(x_t \mid x_{<t})$$

The parameters of the model

- Billions of numbers (weights) in a neural network.
- Training = adjusting *theta* to make predictions better.

sum over all positions in the text sequence

- Every word/token in the training data contributes to the training objective.
- The model learns from predicting each next token correctly.

This is the heart of the LM: predict the probability that the model with parameters θ assigns to token x_t given the context - that is the token that came before $x_{<t}$.

From language models to large language models

- **Masked LM** (BERT, 2018): predicts hidden tokens. Great for understanding and classification tasks.
- **Autoregressive LM** (GPT, 2018+): predicts the next token. Better for generation.
- Today's LLMs mostly **autoregressive** → fluent, flexible outputs.

Figure 1-2 shows these two types of language models.

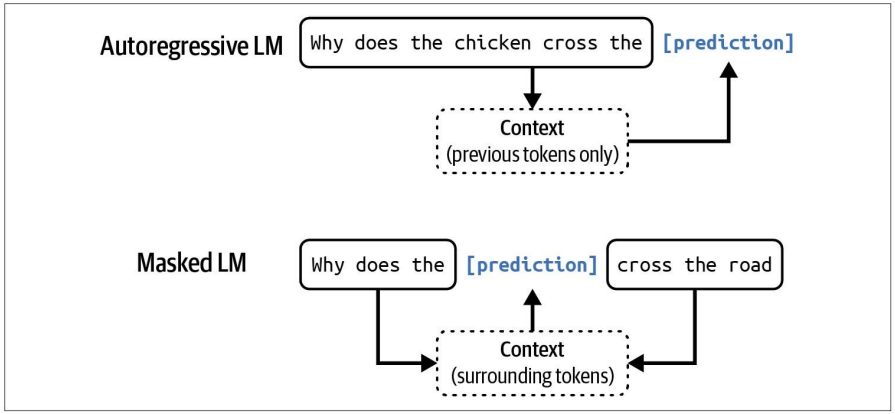


Figure 1-2. Autoregressive language model and masked language model.

So what's different with LLMs?

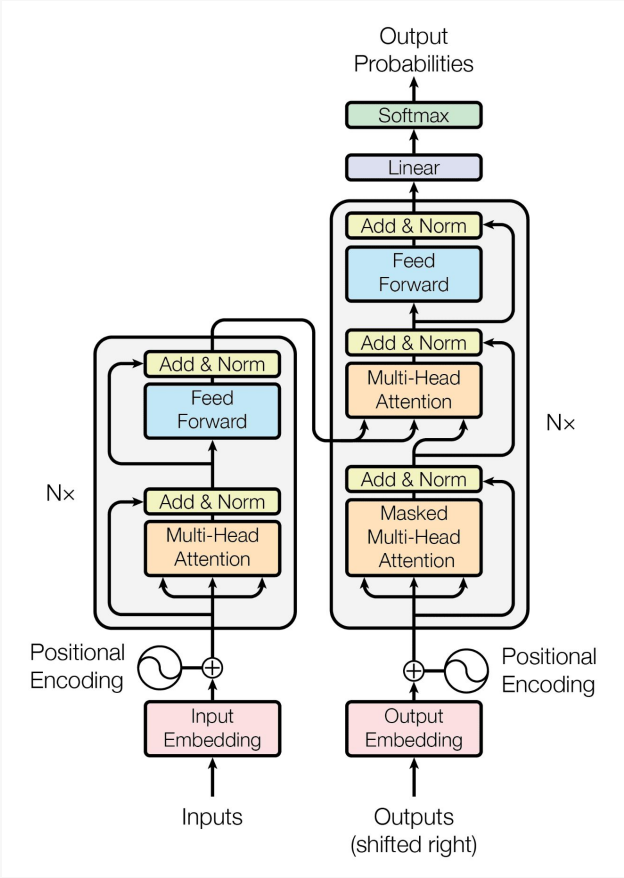
- **Scale**
  Much larger parameters, data, and compute → richer learned distributions, emergent capabilities.

- **Architecture**
  From n-grams and RNN/LSTM to transformers with self-attention and long context windows.

- **Training recipe**
  Self-supervised pretraining + instruction tuning + preference optimization (RLHF/DPO) → chat/task behavior.

From language models to large language models

- **N-grams (1990s–2000s):** statistical models with limited context.
- **RNNs & LSTMs (2010s):** introduced memory, but struggled with long sequences.
- Breakthrough: **transformer architecture (2017)** → attention mechanism scales to long context.



Training recipe: self-supervision

- Self-supervised learning: training on massive unlabeled corpora.
- Cost is exponentially lower: no human labels → unprecedented scaling becomes possible.
- Data = internet text, code, images.
- Changed how we think about datasets: *quantity & diversity trump handcrafted labels*.

Input: "The cat sat on the **[MASK]"**

Label: "mat" (comes directly from the original sentence).

<BOS>	the
<BOS>, the,	cat
<BOS>, the, cat	sat
<BOS>, the, cat, sat	on
<BOS>, the, cat, sat, on	the
<BOS>, the, cat, sat, on, the	mat

Training set

Is all we need just a bigger boat?

- **Size does matter in terms of data AND model**
  - Rapid growth in size: From 117M parameters (GPT-1, 2018) to 175B (GPT-3, 2020) to estimates of 1.8T (GPT-4, 2023).
  - Scaling laws (Kaplan et al., 2020): Model performance improves predictably with more data, parameters, and compute.
  - Emergent abilities (Wei et al., 2022): Beyond certain thresholds, qualitatively new capabilities appear (e.g., reasoning, coding).

- **But scaling is costly:**
  - Compute- and data-optimal scaling (Chinchilla, 2022) shows we need ~20 tokens per parameter to train efficiently.
  - Bottlenecks ahead: data scarcity and energy consumption.
  - **Industry reality**: Small, efficient open-source models (Mistral, LLaMA) are gaining adoption because they are cheaper to train and run, even if they don't always hit state-of-the-art accuracy.

Large language models

Traditional Language Models	Large Language Models (LLMs)
Trained on small datasets with limited compute → **narrow capabilities**	Trained on massive datasets with huge compute → **qualitatively new capabilities**
Task-specific (e.g., translation, classification, spam detection, etc.)	**General-purpose** (translation, reasoning, summarization, coding, multi-domain)
Scale = incremental improvements	Scale = emergent abilities (reasoning, tool use, code generation)
Each model trained for a single task	Foundation models: one model adapted to many tasks

Take a look at:

Wei et al., *Emergent Abilities of Large Language Models* (2022) arXiv

Kaplan et al., *Scaling Laws for Neural Language Models* (2020) arXiv

Multi-modal models

- Foundation models encapsulate knowledge about the world
- And data comes in several modalities
- Historically we don’t do a lot with unstructured data
- Foundation models will increasingly be multi-modal

An example: from CLIP to Fashion CLIP

scientific reports

Explore content · About the journal · Publish with us

nature › scientific reports › articles › article

Article | Open access | Published: 08 November 2022

Contrastive language and vision learning of general fashion concepts

Patrick John Chia ✉, Giuseppe Attanasio, Federico Bianchi, Silvia Terragni, Ana Rita Magalhães, Diogo Goncalves, Ciro Greco & Jacopo Tagliabue

Scientific Reports **12**, Article number: 18958 (2022) | Cite this article

21k Accesses | **32** Citations | **14** Altmetric | Metrics

A Publisher Correction to this article was published on 23 January 2023

This article has been updated

Abstract

The steady rise of online shopping goes hand in hand with the development of increasingly complex ML and NLP models. While most use cases are cast as specialized supervised learning problems, we argue that practitioners would greatly benefit from general and transferable representations of products. In *this* work, we build on recent developments in contrastive learning to train *FashionCLIP*, a *CLIP*-like model adapted for the fashion industry. We demonstrate the effectiveness of the representations learned by *FashionCLIP* with extensive tests across a variety of tasks, datasets and generalization probes. We argue that adaptations of large pre-trained models such as CLIP offer new perspectives in terms of scalability and sustainability for certain types of players in the industry. Finally, we detail the costs and environmental impact of training, and release the model weights and code as open

AI engineering

The AI engineer

Three layers of responsibility

1. **Application Development** – prompt design, context management, user interfaces, evaluation loops.

2. **Model Development** – fine-tuning, dataset engineering, inference optimization.

3. **Infrastructure and LLMOps** – serving models, managing compute, monitoring, scaling.

Model adaptation and application development

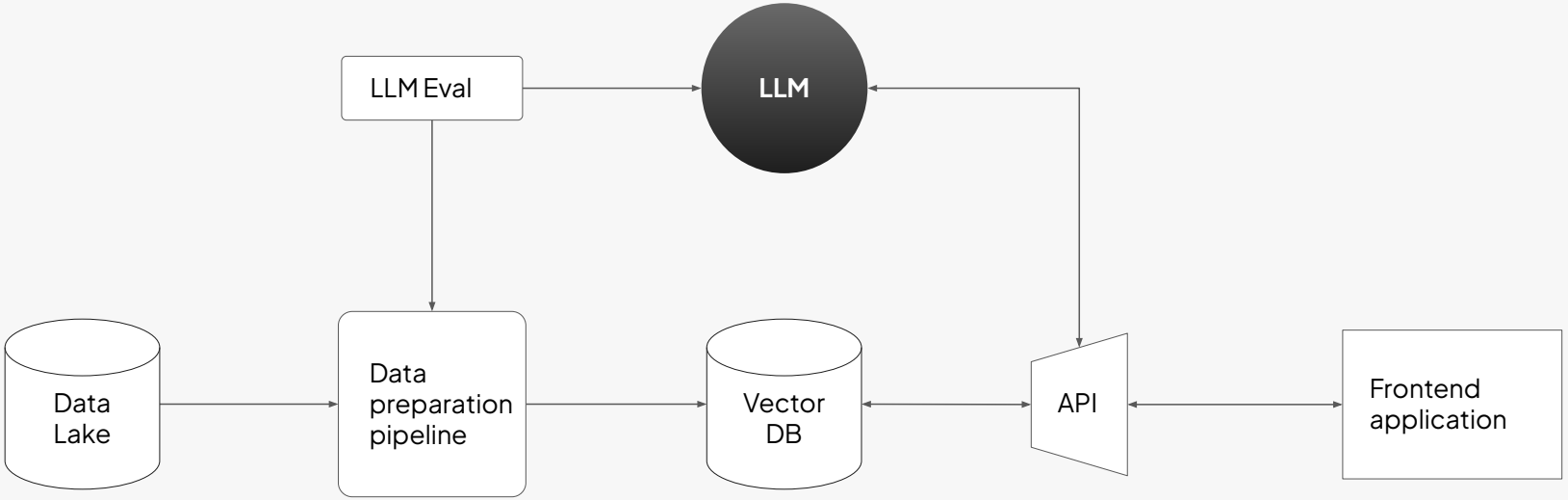
Key differences from ML engineering:

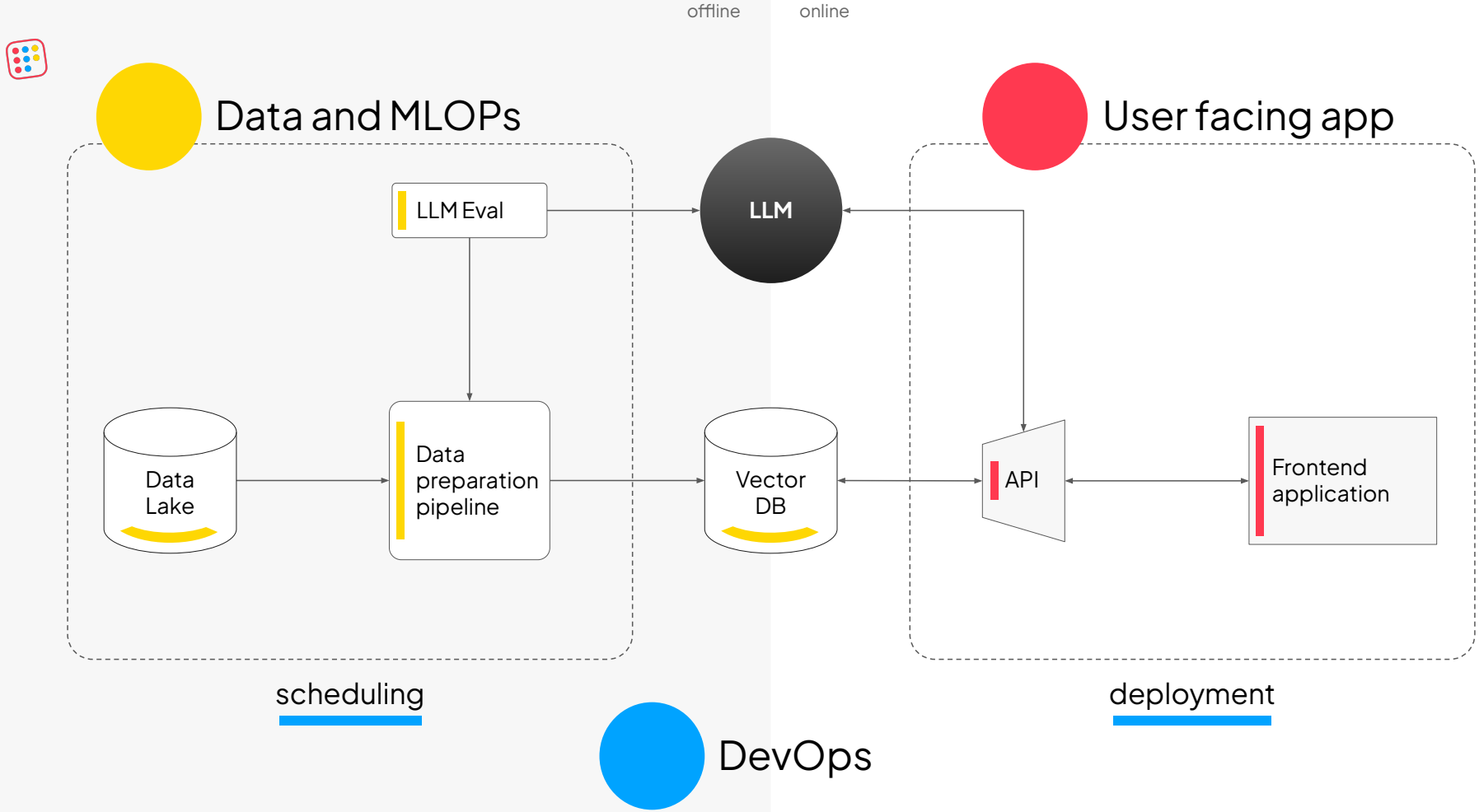
- Work with foundation models trained elsewhere. → Prompt engineering, context engineering, sampling, fine-tuning
- Handle larger, more compute-intensive models (GPU scaling, efficiency). → LLMOps and DataOps
- Manage open-ended outputs → much harder evaluation, testing and guardrails. → Eval, Eval, Eval!

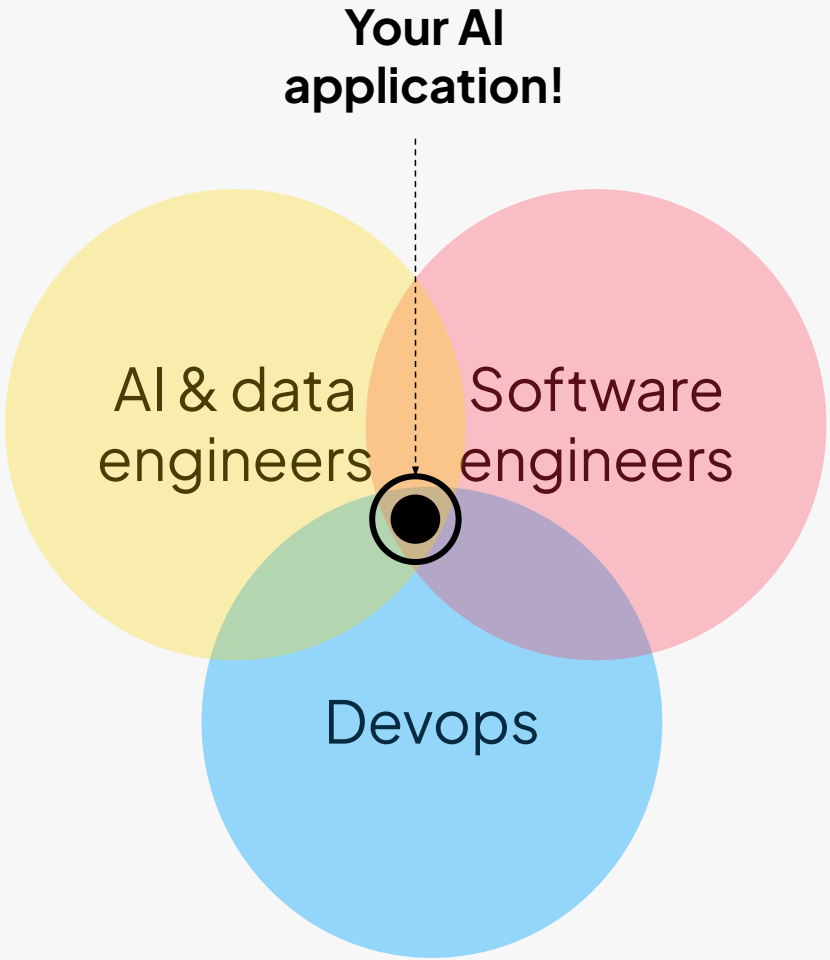
Model adaptation and application development

Closer to product & full-stack work:

- AI engineers increasingly collaborate on UX, feedback loops, and product decisions.
- Many come from web dev / full-stack backgrounds, not only ML.


LLM Eval
LLM
Data Lake
Data preparation pipeline
Vector DB
API
Frontend application


offline
online
Data and MLOPs
User facing app
LLM Eval
LLM
Data Lake
Data preparation pipeline
Vector DB
API
Frontend application
scheduling
deployment
DevOps


Your AI application!
AI & data engineers
Software engineers
Devops

Historically, productionizing ML has been pretty hard

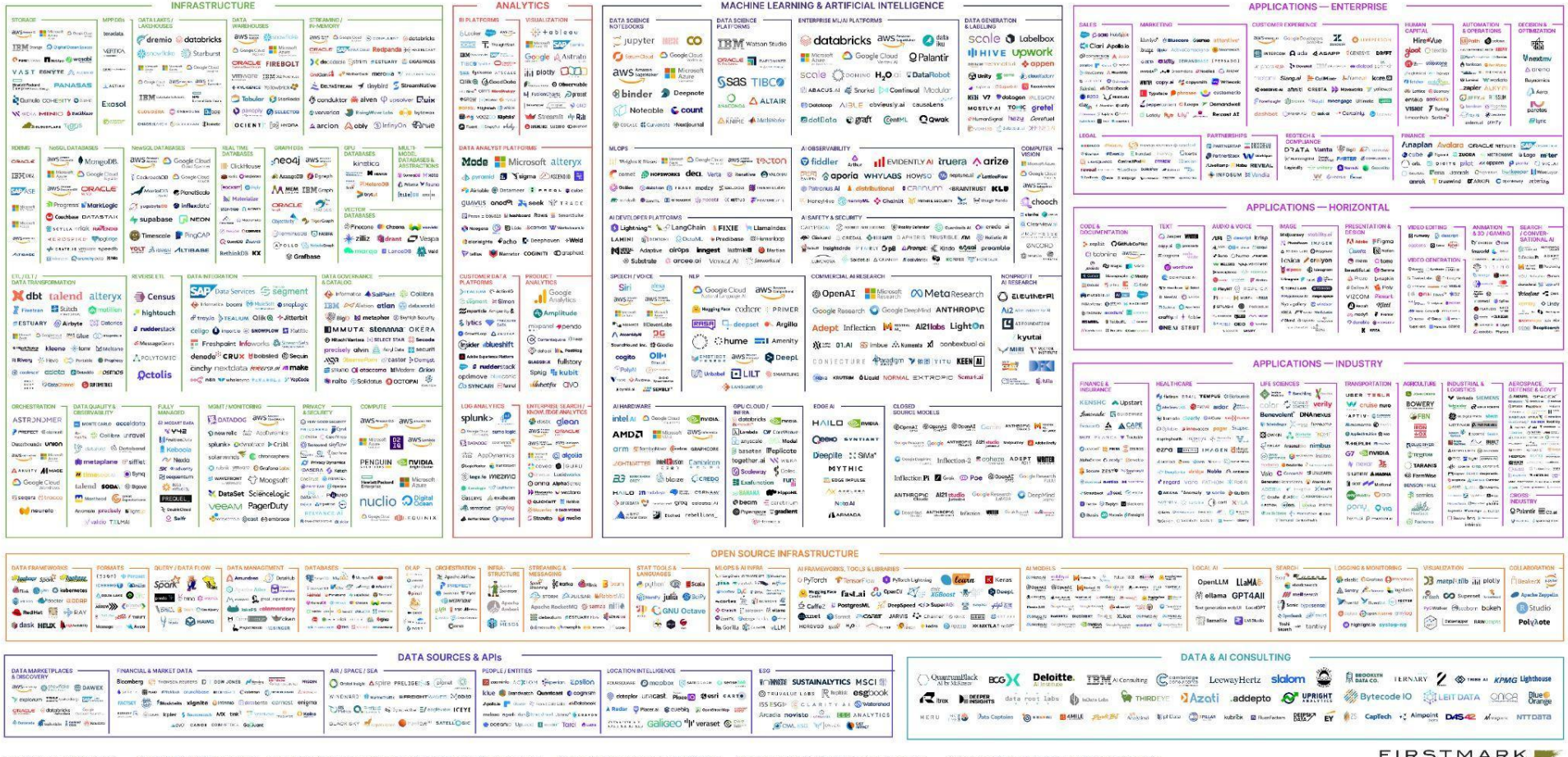
I am a data scientist, I work with files and notebooks

Data

Notebooks on my laptop and Python scripts

DevOps

THE 2024 MAD (MACHINE LEARNING, ARTIFICIAL INTELLIGENCE & DATA) LANDSCAPE



Version 1.0 - March 2024 Blog post: mattturck.com/MAD2024 Interactive version: MAD.firstmarkcap.com Comments? Email MAD2024@firstmarkcap.com

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

It still is, but LLMs made it much easier to get started

- Product mentality is a must-have.
- IT mentality is a big no.
- So:
- Invest in FinOps! Make your your costs easy to monitor and control.
- Invest in Developer Experience! Make your infrastructure simple so your product developers can participate.

Paradigm shift in product development

Circa 2015–2023*

- Invest in data
- Invest in Analytics
- Invest in Data Science
- Invest in ML and AI
- Deploy AI applications
- Measure ROI

Today

- Put together a prototype
- Measure rough ROI
- Invest in data
- Invest in Analytics
- Invest in Data Science
- Invest in ML and AI

Application development

Data science + model development + MLOps

Data

Application development

LLM APIs

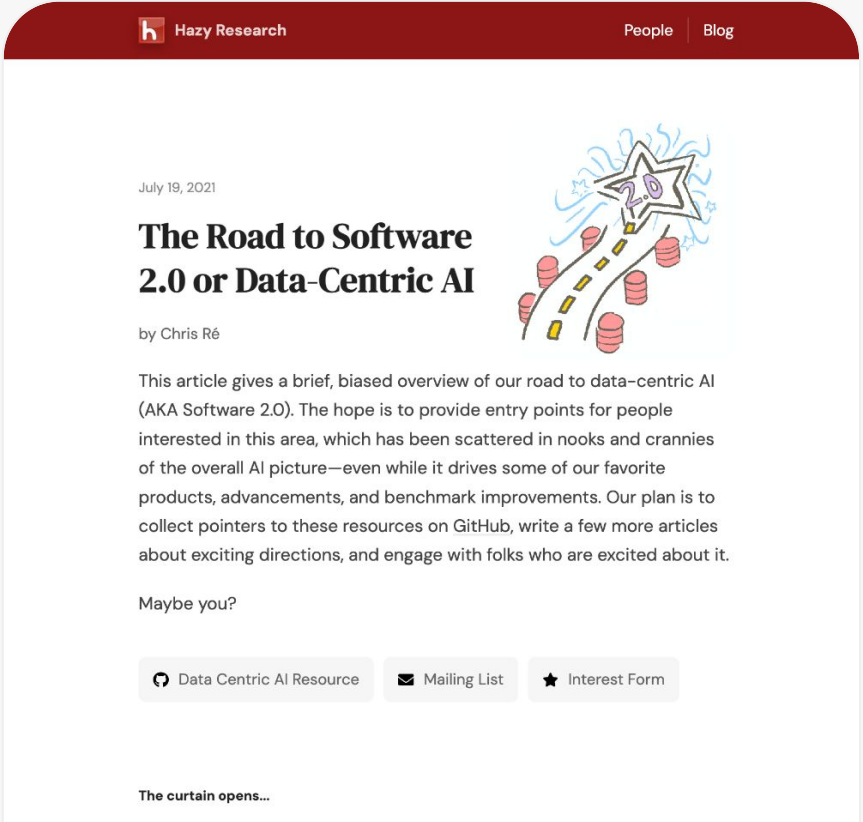
Data

AI will not fix your data problem

- **Data strategy** - your data touch engineering, product, security compliance and legal. You might want to think about it thoroughly.
- **Data quality** - if AI has high ROI for you and you have some special that only you have, using OpenAI won't cut it.
- **Data infrastructure** - data infrastructure is historically super hard and an IT domain. Simplify and put it in the hands of application builders.

Data centric AI

- The models are a commodity.
- Textual public data also is.
- Your data is what makes you special.



Why should you care about data?

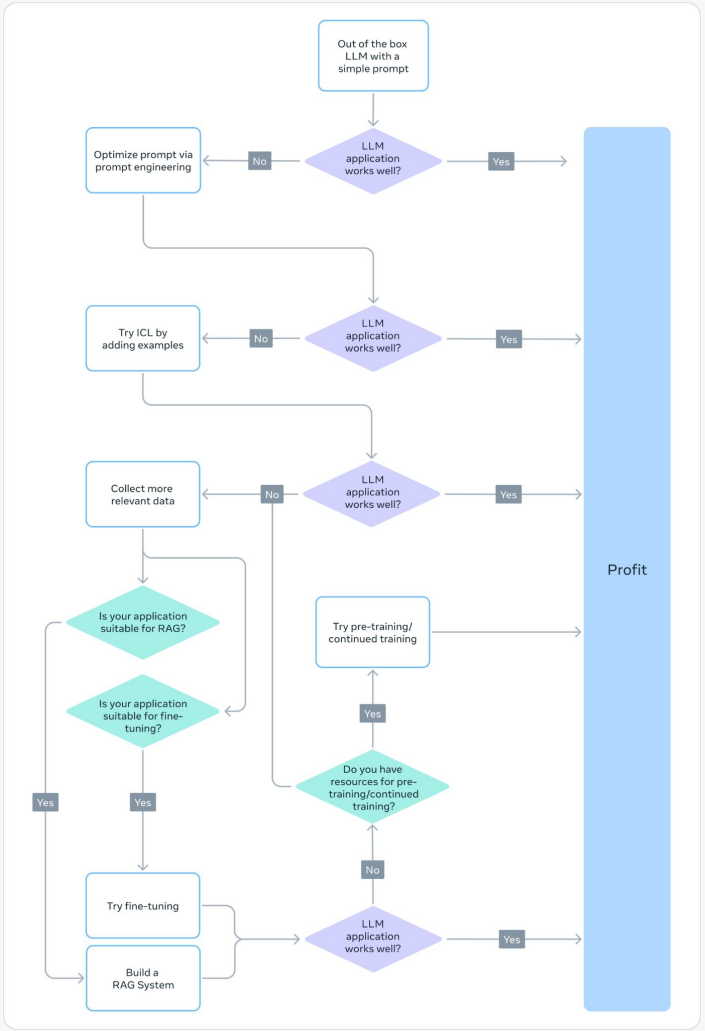
- **Pretraining is expensive and hard. Nobody really needs that, except those who build foundational models.**
  - 50B to ~300B parameters, $5 to $50M for end-to-end training, there are probably 2000 people in the world that really know how to do this.
  - Done yearly.
  - Quantity over quality
  - Kind of the same for everybody.
- **Prompt engineering, post-training, fine-tuning, context engineering is where most of the improvement is.**
  - Fine-tuning on labelled data - 100k to 500k tokens.
  - Done daily or weekly.
  - Quality over quantity.
  - Your specific data matter!

Why should you care about data?

- **Differentiate** – Use data specific to your business.
- **Integrate** – Use accurate, up-to-date information.
- **Govern** – Ensure auditability, compliance, and trust.
- **Improve** – Use feedback data to improve AI systems.

Buy vs. build

- Spend as much time as you can understanding the use case.
- Identify weak vs. strong constraints.
- Start simple and determine ROI.
- Do not - I repeat do not - do what Google does.
- Assume you'll operate at a reasonable scale.



Source: https://ai.meta.com/blog/adapting-large-language-models-llms/

Problem Domain	Why not LLMs
Recommender Systems	Sensitive to prompt ordering → unstable, high variance (arXiv)
Time Series Forecasting	No better than simpler models; lack temporal structure and are computationally expensive (arXiv, NeurIPS, ResearchGate)
Anomaly Detection	Specialized models outperform LLMs; LLMs require workaround and are less reliable (ResearchGate, ScienceDirect, MIT News)
Common Challenges	Structured data, precision needs, temporal consistency, cost constraints, and stability **requirements.**

Recommender systems

- Most of ML applications out there are actually recommender systems.
- Recsys are a big deal as they are ubiquitous in our life:
  - Movies
  - Books
  - News
  - Shopping
  - People
- The Recsys market is predicted to reach around [15 BN by 2026.]