Introduction to Building AI Applications with Foundation Models

CSCI 4440U | Topics in Computer Science
AI-Powered Software Engineering

Cristiano Politowski

Language Models

A language model encodes statistical information about one or more languages.

This information tells us how likely a word is to appear in a given context.

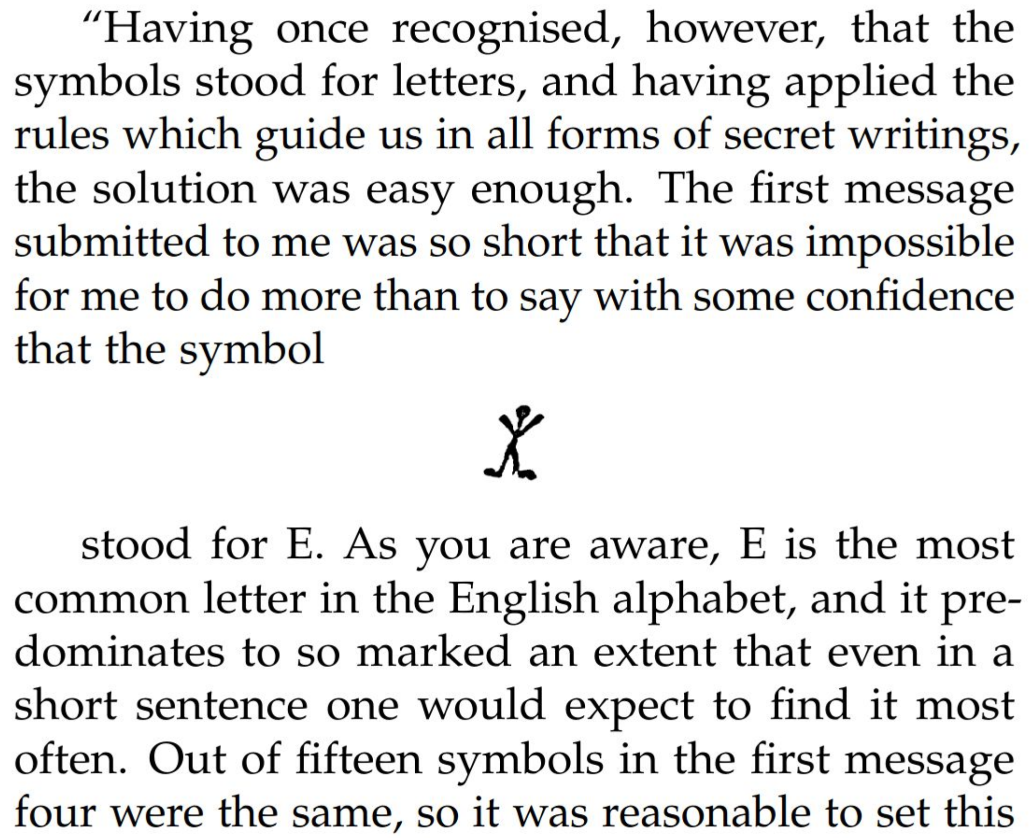
For example, given the context **“My favorite color is __”**, a language model should predict **“blue”** more often than **“car”**.

The Adventure of the Dancing Men

(1905) “The Adventure of the Dancing Men”, Sherlock Holmes.

(1951) Claude Shannon used more sophisticated statistics to decipher enemies’ messages during the Second World War: **“Prediction and Entropy of Printed English”**.

Many concepts introduced in this paper, including *entropy*, are still used for language modeling today.



“Having once recognised, however, that the symbols stood for letters, and having applied the rules which guide us in all forms of secret writings, the solution was easy enough. The first message submitted to me was so short that it was impossible for me to do more than to say with some confidence that the symbol

stood for E. As you are aware, E is the most common letter in the English alphabet, and it predominates to so marked an extent that even in a short sentence one would expect to find it most often. Out of fifteen symbols in the first message four were the same, so it was reasonable to set this

Token

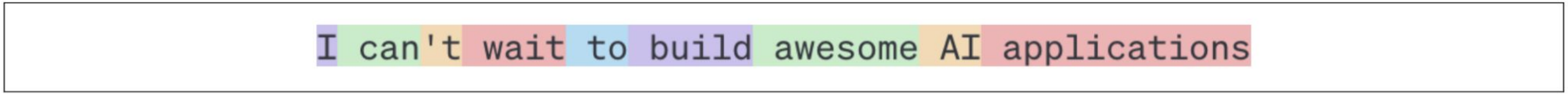
The basic unit of a language model is **token**.

A token can be a character, a word, or a part of a word (like -tion), depending on the model.

- For non-English languages, a single Unicode character can sometimes be represented as multiple tokens.

The process of breaking the original text into tokens is called **tokenization**.

- For GPT-4, an average token is approximately ¾ the length of a word. So, 100 tokens are approximately 75 words.
- https://platform.openai.com/tokenizer



```
I can't wait to build awesome AI applications
```

Vocabulary

The set of all tokens a model can work with is the model's vocabulary.

You can use a small number of tokens to construct a large number of distinct words, similar to how you can use a few letters in the alphabet to construct many words.

- The *Mixtral 8x7B* model has a vocabulary size of 32,000.
- GPT-4's vocabulary size is 100,256.

The tokenization method and vocabulary size are decided by model developers.

Why token instead of word or character?

1. Tokens allow the model to break words into meaningful components. For example, “cooking” can be broken into “cook” and “ing”, with both components carrying some meaning of the original word.
2. There are fewer unique tokens than unique words, this reduces the model’s vocabulary size, making the model more efficient.
3. Tokens also help the model process unknown words. For instance, a made-up word like “chatgpting” could be split into “chatgpt” and “ing”, helping the model understand its structure.

Tokens balance having fewer units than words while retaining more meaning than individual characters.

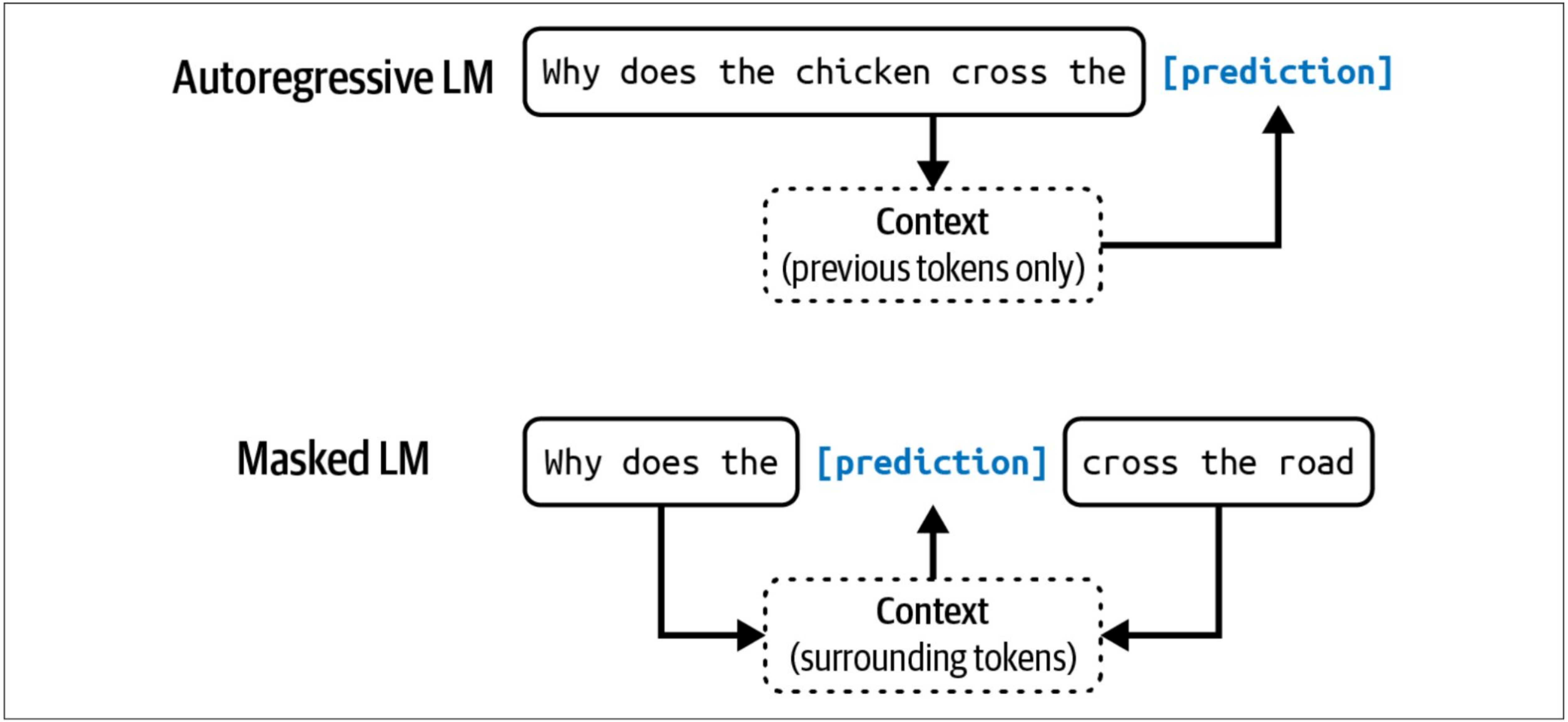
Language Models Types: Masked & Autoregressive

Masked language model: Trained to predict missing tokens **anywhere** in a sequence, using the context from both before and after the missing tokens.

- Commonly used for non-generative tasks such as: sentiment analysis, text classification, and code debugging.

Autoregressive language model (**Causal language models**): Trained to predict the **next token** in a sequence, using only the preceding tokens.

- An autoregressive model can continually generate one token after another.
- Commonly used for text generation -> much more popular.


Autoregressive LM
Why does the chicken cross the
[prediction]
Context
(previous tokens only)
Masked LM
Why does the
[prediction]
cross the road
Context
(surrounding tokens)

Generative AI

The outputs of language models are **open-ended**.

- A language model can use its fixed, finite vocabulary to construct **infinite possible outputs**.
- A model that can generate open-ended outputs is called **generative**, hence the term generative AI.

Completions are predictions, based on probabilities, and not guaranteed to be correct.

- This probabilistic nature of language models makes them both so exciting and frustrating to use.

Self-supervision

Language modeling is just **one of many ML algorithms**.

- Object detection, topic modeling, recommender systems, weather forecasting, stock price prediction, etc.

Language models can be trained using self-supervision, while many other models require supervision.

- **Supervision** refers to the process of training ML algorithms using labeled data: expensive and slow to obtain.
- **Self supervision** allows create larger datasets for models to learn from, effectively allowing models to scale up.

With supervision, you label examples to show the behaviors you want the model to learn, and then train the model on these examples. Once trained, the model can be applied to new data.

The success of AI models in the 2010s lay in supervision.

- The model that started the deep learning revolution, AlexNet (Krizhevsky et al., 2012), was supervised.
- It was trained to learn how to classify over 1 million images in the dataset ImageNet.
- It classified each image into one of 1,000 categories such as “car”, “balloon”, or “monkey”.

Self-supervision

A drawback of supervision is that data labeling is expensive and time-consuming.

- If it costs 5 cents for one person to label one image, it’d cost $50,000 to label a million images for ImageNet.
- However, not all labeling tasks are that simple. Generating Latin translations for an English-to-Latin model is more expensive.

Self-supervision helps overcome the data labeling bottleneck.

- In self-supervision, instead of requiring explicit labels, the model can infer labels from the input data.
- Language modeling is self-supervised because each input sequence provides both the labels (tokens to be predicted) and the contexts the model can use to predict these labels.
- For example, the sentence “I love street food.” gives six training samples.

Table 1-1. Training samples from the sentence "I love street food." for language modeling.

Input (context)	Output (next token)
`<BOS>`	`I`
`<BOS>, I`	`love`
`<BOS>, I, love`	`street`
`<BOS>, I, love, street`	`food`
`<BOS>, I, love, street, food`	`.`
`<BOS>, I, love, street, food, .`	`<EOS>`

Large Language Models (LLMs)

Self-supervised learning means that language models can learn from text sequences without requiring any labeling.

- Because text sequences are everywhere (in books, blog posts, articles, and Reddit comments) it’s possible to construct a massive amount of training data, allowing language models to scale up to become **LLMs**.

Self-supervision differs from unsupervision.

- In self-supervised learning, labels are inferred from the input data.
- In unsupervised learning, you don’t need labels at all.

Parameter

LLM, however, is hardly a scientific term.

A model’s size is typically measured by its number of parameters.

A parameter is a variable within an ML model that is updated through the training process.

In general the more parameters a model has, the greater its capacity to learn desired behaviors.

- June 2018: OpenAI’s first generative pre-trained transformer (GPT) - 117 million parameters
- February 2019: GPT-2 with 1.5 billion parameters, 117 million -> small.
- Today*: A model with 100 billion parameters is considered large.

Model weights to refer to all parameters.

Why do larger models need more data?

Larger models have more capacity to learn, and, therefore, would need more training data to maximize their performance.

- You can train a large model on a small dataset too, but it'd be a waste of compute.
- Similar or better results on this dataset with smaller models.

It seems counterintuitive that larger models require more training data.

- If a model is more powerful, shouldn't it require fewer examples to learn from?
- We're not trying to get a large model to match the performance of a small model using the same data. We're trying to **maximize model performance**.

Foundation Models

Foundation models mark a breakthrough from the traditional structure of AI research.

For a long time, AI research was divided by data modalities.

- Natural language processing (NLP) deals only with text.
- Computer vision deals only with vision.
- Text-only models can be used for tasks such as translation and spam detection.
- Image-only models can be used for object detection and image classification.
- Audioonly models can handle speech recognition (speech-to-text, or STT).

A model that can work with more than one data modality is also called a **multimodal model**.

A generative multimodal model is also called a **large multimodal model (LMM)**.

Foundation models to refer to both **large language models** and **large multimodal models**.

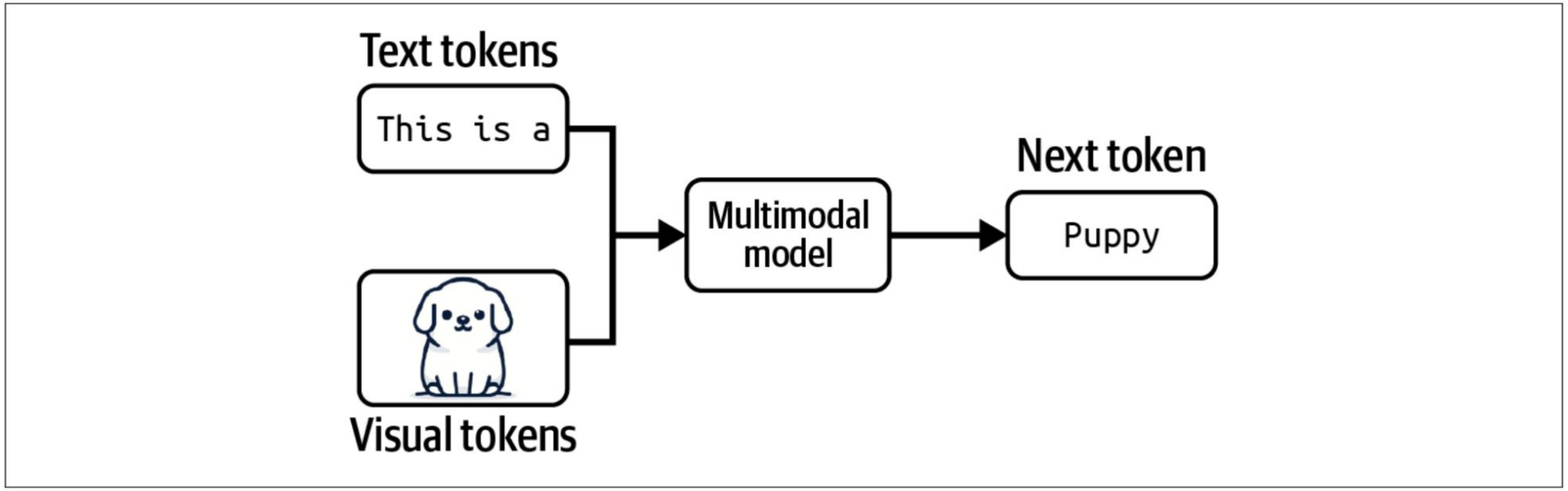


Figure 1-3. A multimodal model can generate the next token using information from both text and visual tokens.

Foundation Models

Foundation models also mark the transition from task-specific models to general purpose models.

Foundation models, thanks to their scale and the way they are trained, are capable of a wide range of tasks.

- An LLM can do both sentiment analysis and translation.

Super-Natural Instructions is a benchmark to evaluate foundation models.

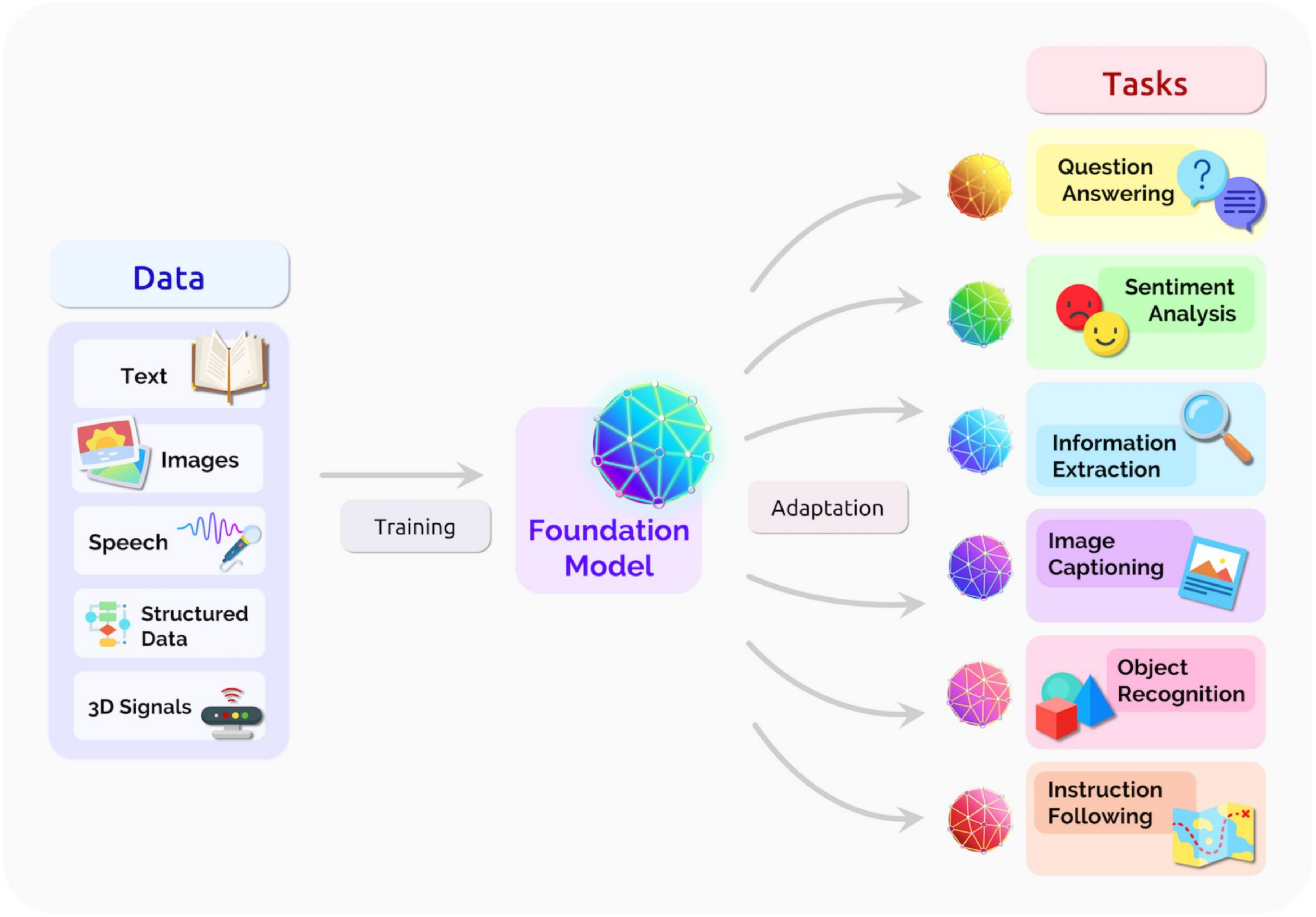
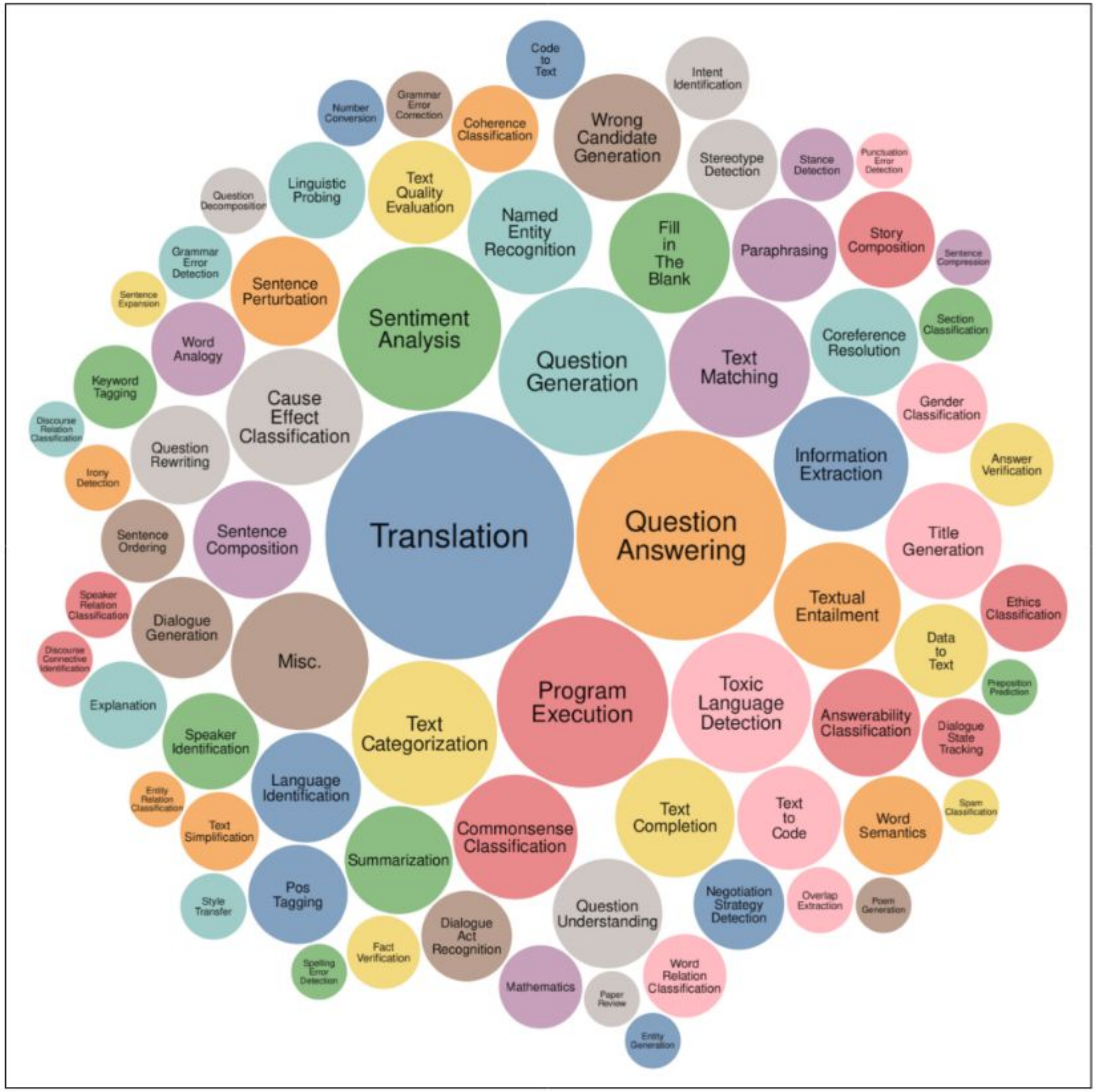


Fig. 2. A foundation model can centralize the information from all the data from various modalities. This one model can then be adapted to a wide range of downstream tasks.


Code to Text
Intent Identification
Number Conversion
Grammar Error Correction
Coherence Classification
Wrong Candidate Generation
Stereotype Detection
Stance Detection
Punctuation Error Detection
Question Decomposition
Linguistic Probing
Text Quality Evaluation
Named Entity Recognition
Fill in The Blank
Paraphrasing
Story Composition
Sentence Compression
Grammar Error Detection
Sentence Perturbation
Sentiment Analysis
Sentence Expansion
Question Generation
Text Matching
Coreference Resolution
Section Classification
Word Analogy
Keyword Tagging
Cause Effect Classification
Gender Classification
Discourse Relation Classification
Question Rewriting
Information Extraction
Answer Verification
Irony Detection
Translation
Question Answering
Title Generation
Sentence Ordering
Sentence Composition
Textual Entailment
Ethics Classification
Speaker Relation Classification
Dialogue Generation
Misc.
Data to Text
Discourse Connective Identification
Program Execution
Toxic Language Detection
Preposition Prediction
Explanation
Text Categorization
Answerability Classification
Speaker Identification
Dialogue State Tracking
Language Identification
Entity Relation Classification
Text Completion
Text to Code
Word Semantics
Spam Classification
Text Simplification
Commonsense Classification
Summarization
Pos Tagging
Question Understanding
Negotiation Strategy Detection
Overlap Extraction
Poem Generation
Style Transfer
Dialogue Act Recognition
Fact Verification
Spelling Error Detection
Mathematics
Paper Review
Word Relation Classification
Entity Generation

Prompt Engineering, RAG, and Finetune

There are multiple techniques you can use to get the model to generate what you want.

Prompt Engineering: craft detailed instructions with examples of the desirable product descriptions.

Retrieval Augmented Generation (RAG): using a database to supplement the instructions.

Finetune: further train the model on a dataset of high-quality product descriptions.

Prompt engineering, RAG, and finetuning are three very common AI engineering techniques that you can use to adapt a model to your needs.

AI engineering

AI engineering refers to the process of **building applications** on top of foundation models.

People have been building AI applications for over a decade

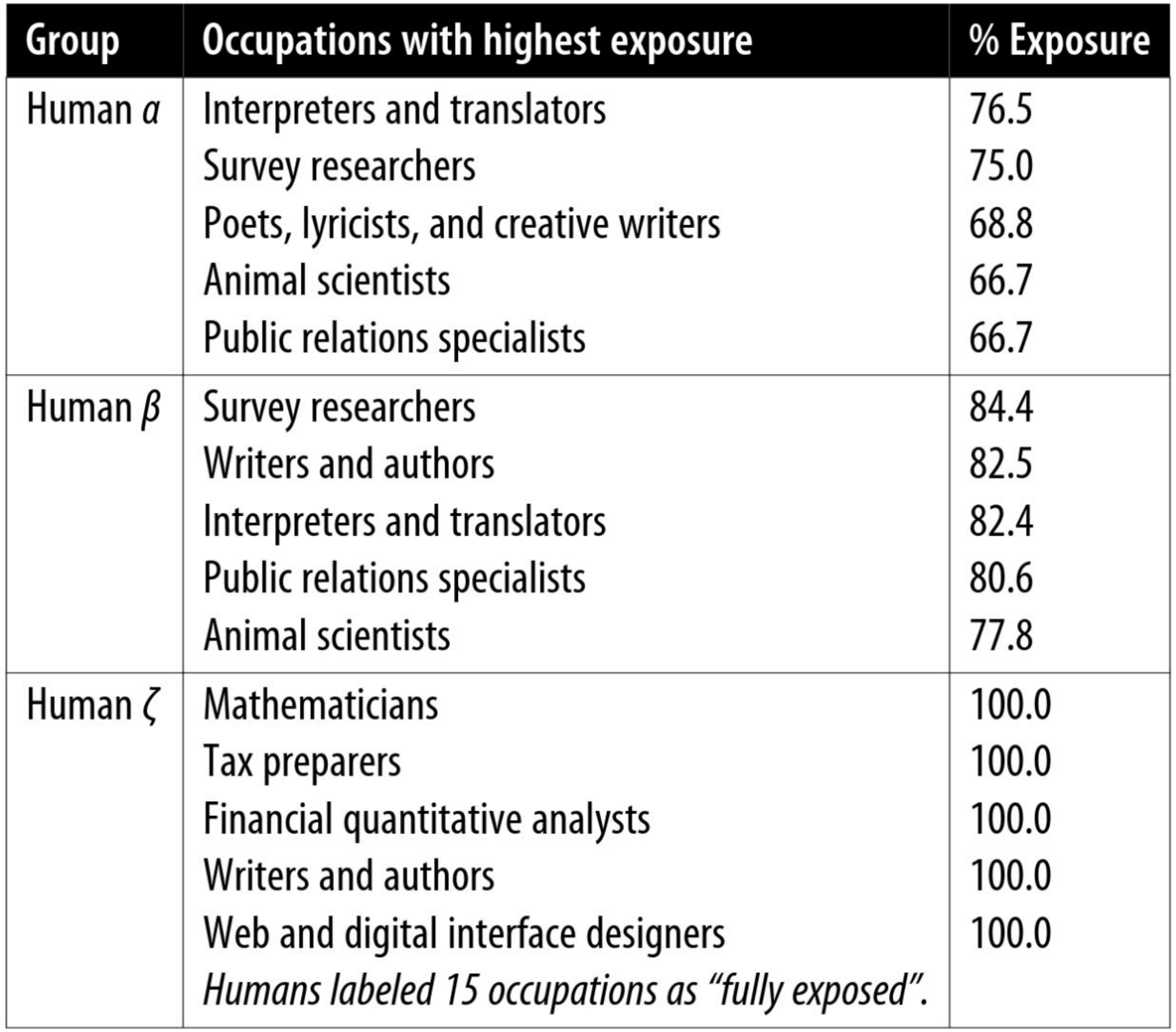
- A process often known as ML engineering or MLOps (short for ML operations).

Why do we talk about AI engineering now?

- If traditional ML engineering involves developing ML models, AI engineering **leverages existing ones**.

How exposed different occupations are to AI

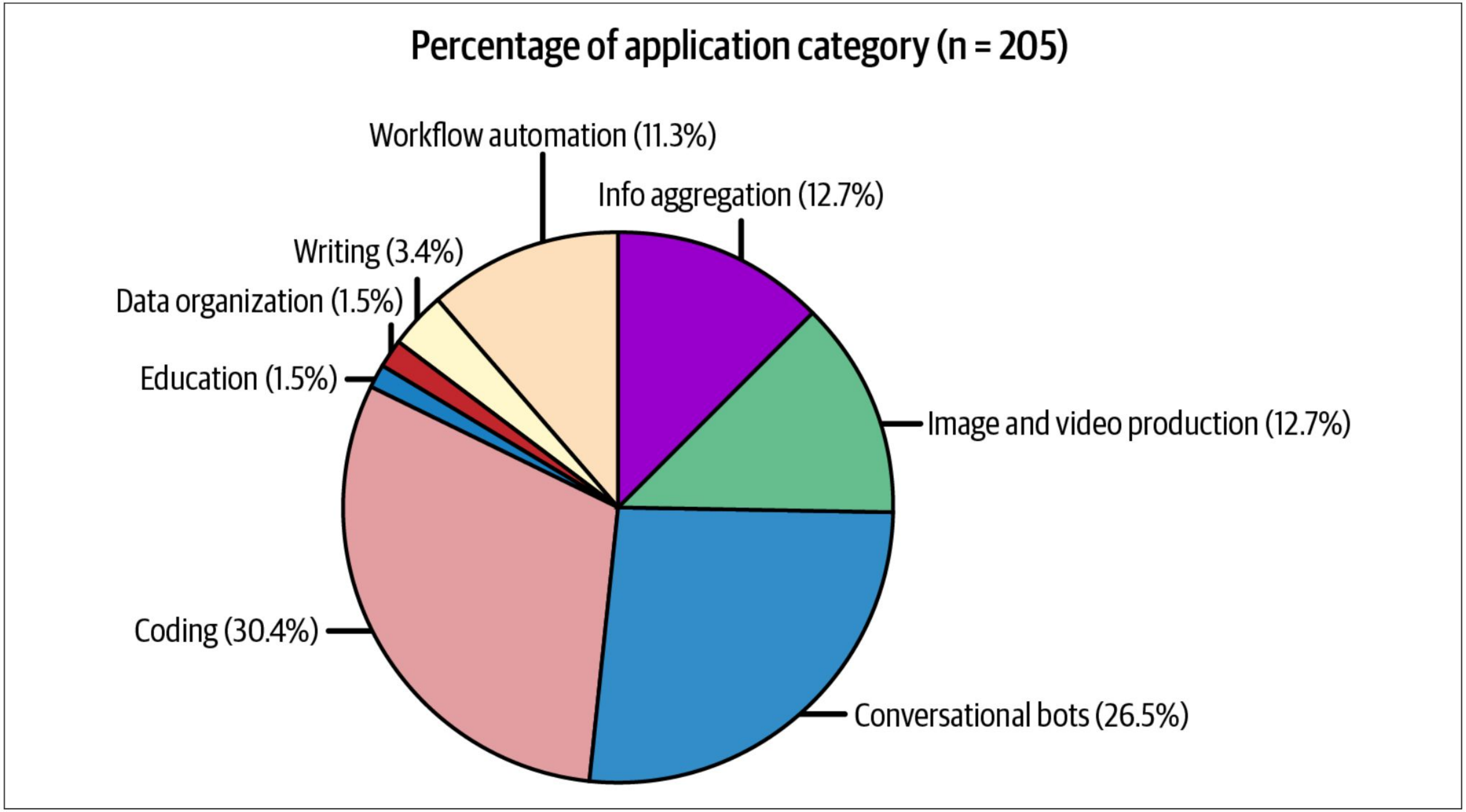
A task is exposed if AI can reduce the time needed to complete this task by at least 50%.



Group	Occupations with highest exposure	% Exposure
Human α	Interpreters and translators	76.5
	Survey researchers	75.0
	Poets, lyricists, and creative writers	68.8
	Animal scientists	66.7
	Public relations specialists	66.7
Human β	Survey researchers	84.4
	Writers and authors	82.5
	Interpreters and translators	82.4
	Public relations specialists	80.6
	Animal scientists	77.8
Human ζ	Mathematicians	100.0
	Tax preparers	100.0
	Financial quantitative analysts	100.0
	Writers and authors	100.0
	Web and digital interface designers	100.0
	Humans labeled 15 occupations as "fully exposed".	

huyenchip.com/llama-police

It’s a lot of fun seeing what people are building with AI


Percentage of application category (n = 205)
Workflow automation (11.3%)
Info aggregation (12.7%)
Writing (3.4%)
Data organization (1.5%)
Education (1.5%)
Image and video production (12.7%)
Coding (30.4%)
Conversational bots (26.5%)

Open-source AI Tools - Coding

- Extracting structured data from web pages and PDFs (AgentGPT)
- Converting English to code (DB-GPT, SQL Chat, PandasAI)
- Given a design or a screenshot, generating code that will render into a website that looks like the given image (screenshot-to-code, draw-a-ui)
- Translating from one programming language or framework to another (GPTMigrate, AI Code Translator)
- Writing documentation (Autodoc)
- Creating tests (PentestGPT)
- Generating commit messages (AI Commits)

Open-source AI Tools

- Image and Video Production
- Writing
- Education
- Conversational Bots
- Information Aggregation

“My hypothesis is that we’ll become so distrustful of content on the internet that we’ll only read content generated by people or brands we trust.”

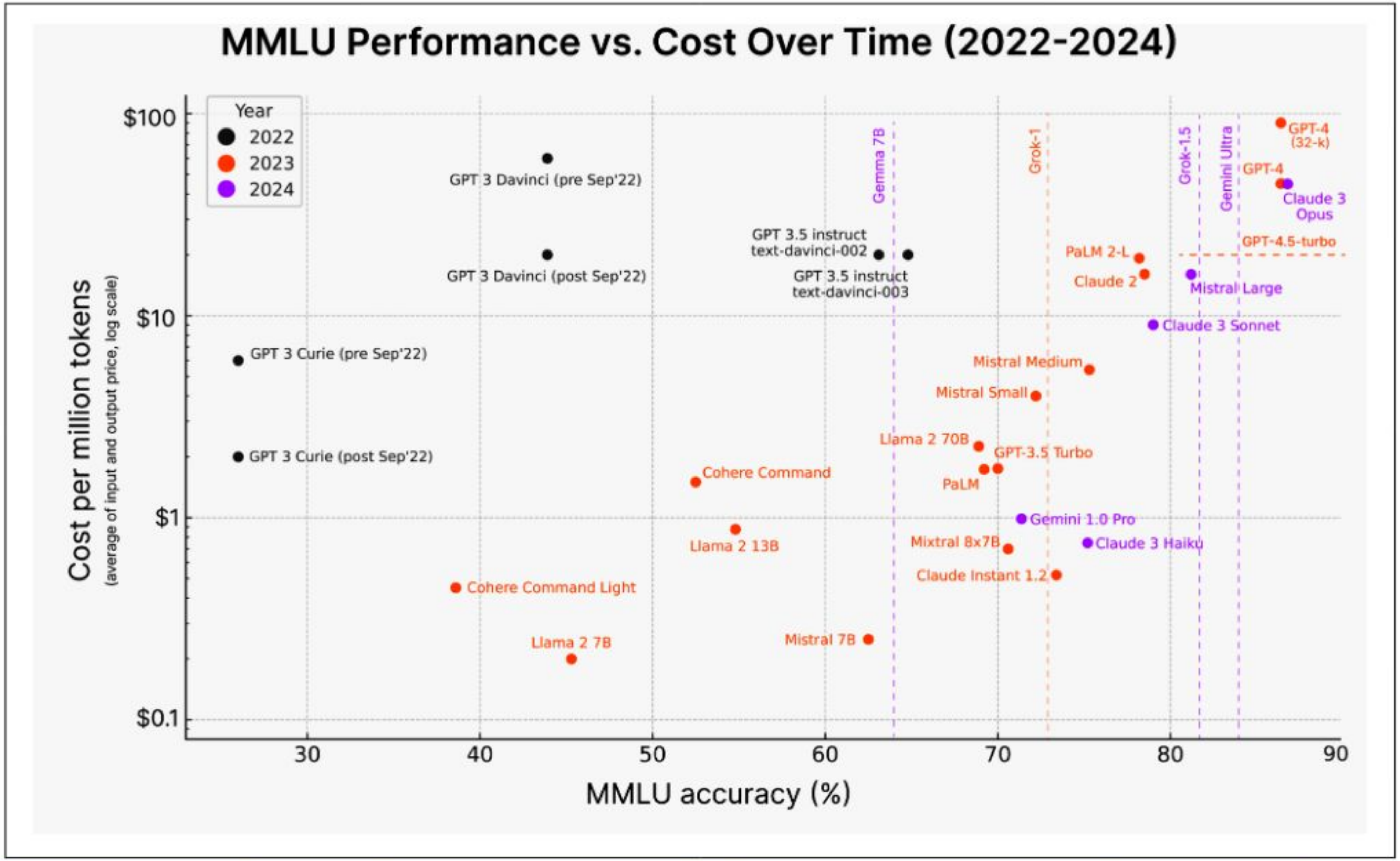
Models Evolution

The limitations of many models are being addressed.

Context lengths are getting longer.

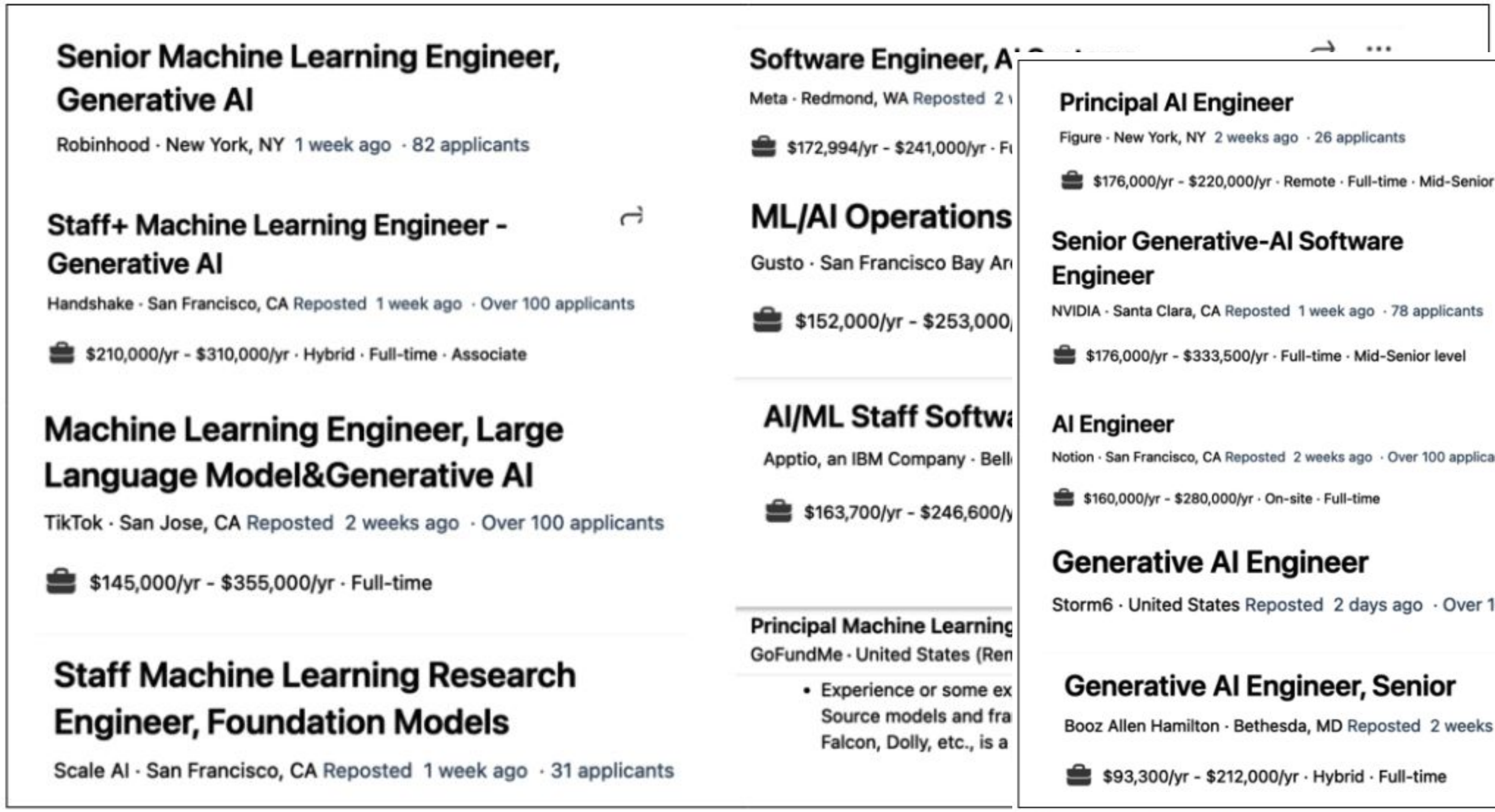
Model outputs are getting better.

Model inference, the process of computing an output given an input, is getting faster and cheaper.


MMLU Performance vs. Cost Over Time (2022-2024)
Year
2022
2023
2024
Cost per million tokens
(average of input and output price, log scale)
$100
$10
$1
$0.1
MMLU accuracy (%)
30
40
50
60
70
80
90
GPT 3 Davinci (pre Sep'22)
GPT 3 Davinci (post Sep'22)
GPT 3 Curie (pre Sep'22)
GPT 3 Curie (post Sep'22)
GPT 3.5 instruct text-davinci-002
GPT 3.5 instruct text-davinci-003
Gemma 7B
Grok-1
Grok-1.5
Gemini Ultra
GPT-4 (32-k)
GPT-4
Claude 3 Opus
GPT-4.5-turbo
PaLM 2-L
Claude 2
Mistral Large
Claude 3 Sonnet
Mistral Medium
Mistral Small
Llama 2 70B
GPT-3.5 Turbo
PaLM
Cohere Command
Gemini 1.0 Pro
Llama 2 13B
Mixtral 8x7B
Claude 3 Haiku
Claude Instant 1.2
Cohere Command Light
Llama 2 7B
Mistral 7B

AI engineering

AI engineering evolved out of ML engineering.



Senior Machine Learning Engineer, Generative AI
Robinhood · New York, NY 1 week ago · 82 applicants

Staff+ Machine Learning Engineer - Generative AI
Handshake · San Francisco, CA Reposted 1 week ago · Over 100 applicants
$210,000/yr - $310,000/yr · Hybrid · Full-time · Associate

Machine Learning Engineer, Large Language Model&Generative AI
TikTok · San Jose, CA Reposted 2 weeks ago · Over 100 applicants
$145,000/yr - $355,000/yr · Full-time

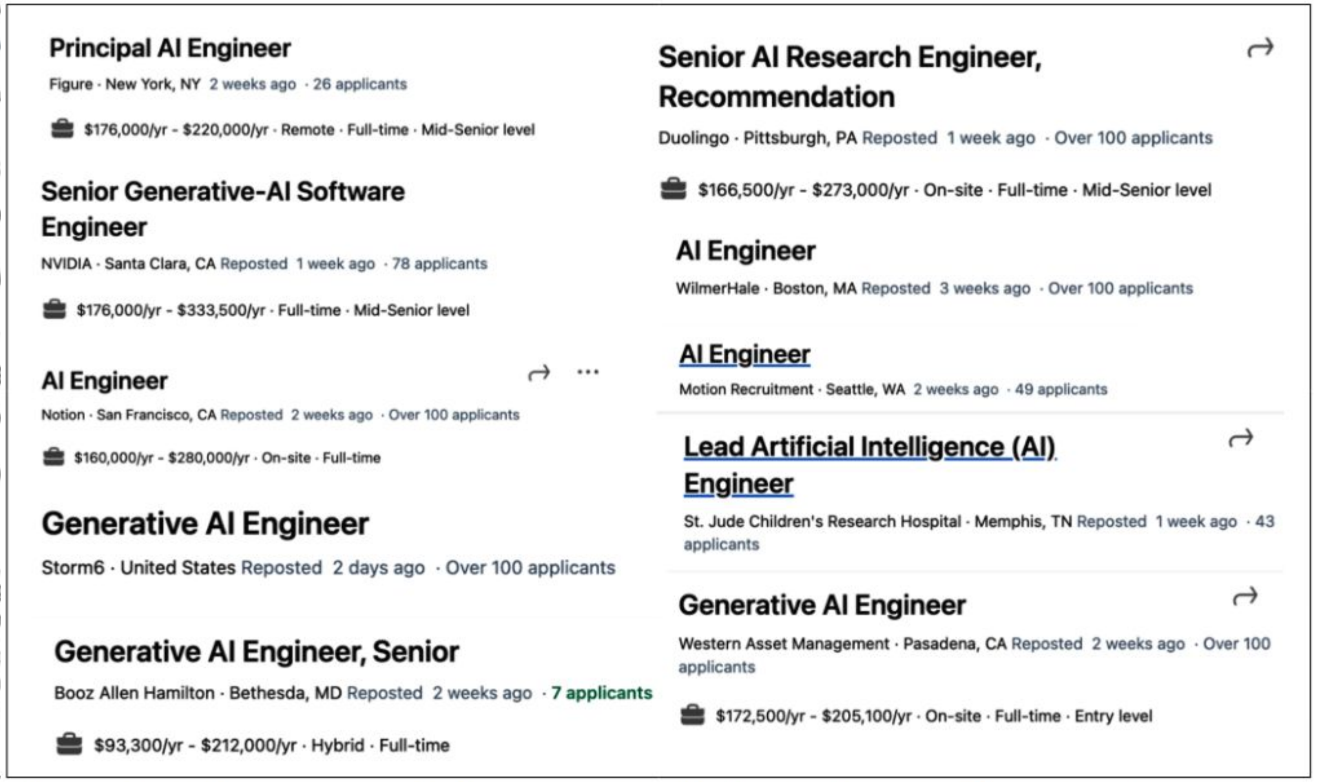
Staff Machine Learning Research Engineer, Foundation Models
Scale AI · San Francisco, CA Reposted 1 week ago · 31 applicants

Software Engineer, A
Meta · Redmond, WA Reposted 2
$172,994/yr - $241,000/yr · F

ML/AI Operations
Gusto · San Francisco Bay Ar
$152,000/yr - $253,000

AI/ML Staff Softw
Apptio, an IBM Company · Bell
$163,700/yr - $246,600/y

Principal Machine Learning
GoFundMe · United States (Ren
- Experience or some ex
Source models and fra
Falcon, Dolly, etc., is a



Principal AI Engineer
Figure · New York, NY 2 weeks ago · 26 applicants
$176,000/yr - $220,000/yr · Remote · Full-time · Mid-Senior level

Senior Generative-AI Software Engineer
NVIDIA · Santa Clara, CA Reposted 1 week ago · 78 applicants
$176,000/yr - $333,500/yr · Full-time · Mid-Senior level

AI Engineer
Notion · San Francisco, CA Reposted 2 weeks ago · Over 100 applicants
$160,000/yr - $280,000/yr · On-site · Full-time

Generative AI Engineer
Storm6 · United States Reposted 2 days ago · Over 100 applicants

Generative AI Engineer, Senior
Booz Allen Hamilton · Bethesda, MD Reposted 2 weeks ago · 7 applicants
$93,300/yr - $212,000/yr · Hybrid · Full-time

Senior AI Research Engineer, Recommendation
Duolingo · Pittsburgh, PA Reposted 1 week ago · Over 100 applicants
$166,500/yr - $273,000/yr · On-site · Full-time · Mid-Senior level

AI Engineer
WilmerHale · Boston, MA Reposted 3 weeks ago · Over 100 applicants

AI Engineer
Motion Recruitment · Seattle, WA 2 weeks ago · 49 applicants

Lead Artificial Intelligence (AI) Engineer
St. Jude Children's Research Hospital · Memphis, TN Reposted 1 week ago · 43 applicants

Generative AI Engineer
Western Asset Management · Pasadena, CA Reposted 2 weeks ago · Over 100 applicants
$172,500/yr - $205,100/yr · On-site · Full-time · Entry level

Three Layers of the AI Stack

Application development: involves providing a model with good prompts and necessary context. This layer requires rigorous evaluation.

Model development: This layer provides tooling for developing models, including frameworks for modeling, training, finetuning, and inference optimization. Because data is central to model development, this layer also contains dataset engineering.

Infrastructure: includes tooling for model serving, managing data and compute, and monitoring.

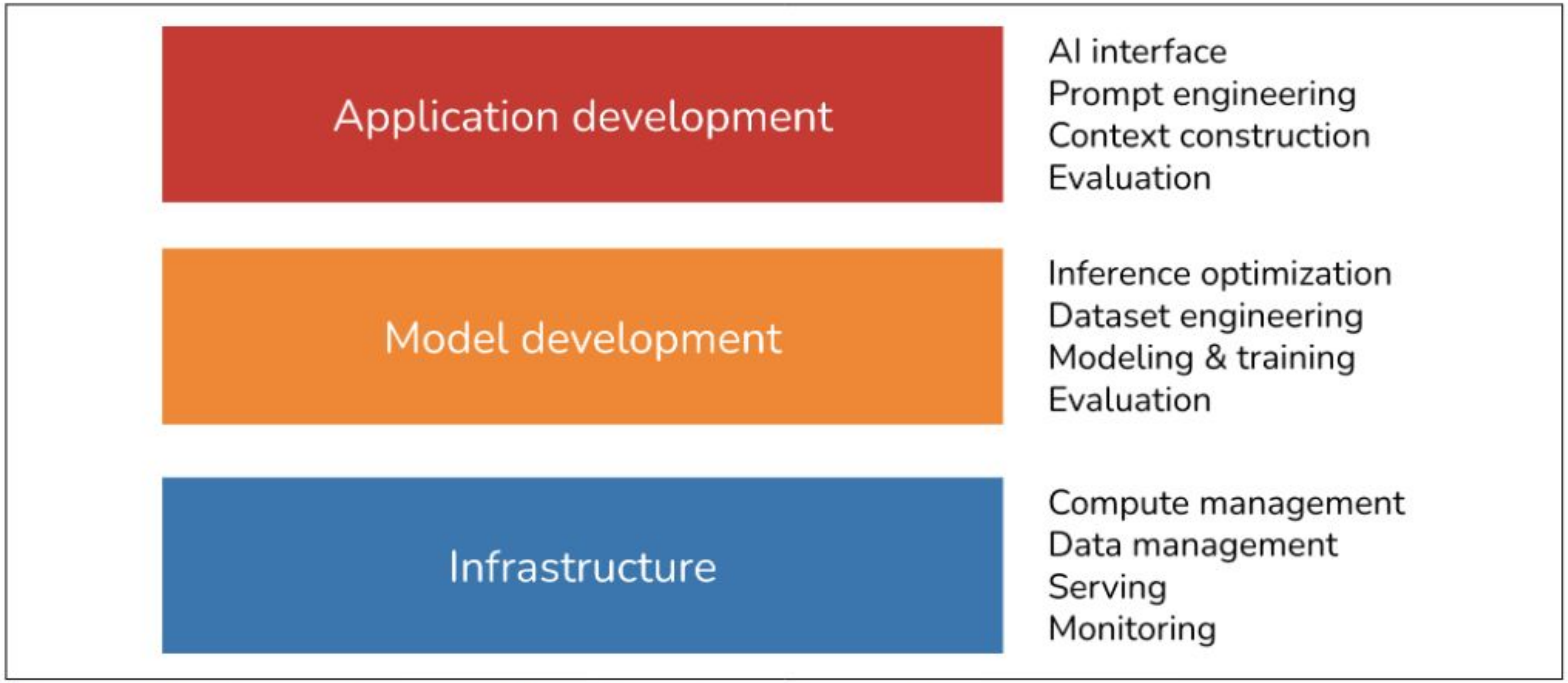


Figure 1-14. Three layers of the AI engineering stack.

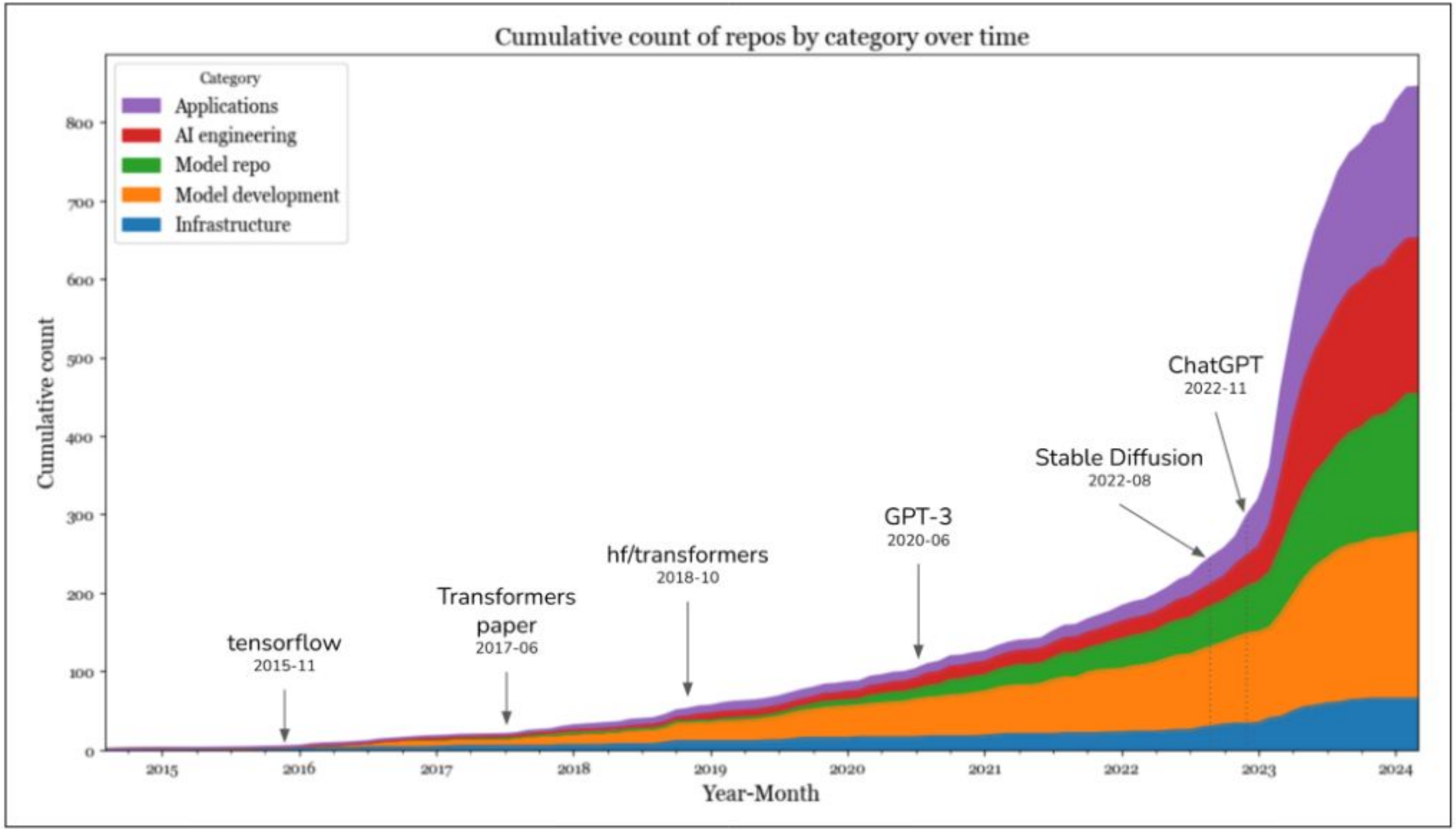


Figure 1-15. Cumulative count of repositories by category over time.

AI engineering vs. ML engineering

1. Without foundation models, you have to train your own models for your applications. With AI engineering, you use a model someone else has trained for you. This means that AI engineering focuses less on modeling and training, and more on **model adaptation**.

2. AI engineering works with models that are bigger, consume **more compute resources**, and incur higher latency than traditional ML engineering. This means that there’s more pressure for efficient training and inference optimization. Many companies now need more GPUs and work with bigger compute clusters than they previously did.

3. AI engineering works with models that can produce **open-ended outputs**. Open Ended outputs give models the flexibility to be used for more tasks, but they are also **harder to evaluate**. This makes evaluation a much bigger problem in AI engineering.

AI engineering differs from ML engineering in that it’s less about **model development** and more about **adapting and evaluating models**.

Model adaptation techniques

Prompt engineering: adapt a model without updating the model weights.

- You adapt a model by giving it instructions and context instead of changing the model itself.
- Prompt engineering is easier to get started and requires less data.
- However, prompt engineering might not be enough for complex tasks.

Finetuning: requires updating model weights.

- You adapt a model by making changes to the model itself.
- More complicated and require more data, but they can improve your model's quality, latency, and cost significantly.
- Many things aren't possible without changing model weights.

Model Development

Model development is the layer most commonly associated with traditional ML engineering.

It has three main responsibilities:

1. modeling and training,
2. dataset engineering,
3. inference optimization.

1. Modeling and training

The process of coming up with a **model architecture**, training it, and finetuning it.

- Examples of tools in this category are Google's TensorFlow, Hugging Face's Transformers, and Meta's PyTorch.

Developing ML models requires **specialized ML knowledge**.

- It requires knowing different types of ML algorithms (such as clustering, logistic regression, decision trees, and collaborative filtering) and neural network architectures (such as feedforward, recurrent, convolutional, and transformer).

It also requires understanding **how a model learns**, including concepts such as gradient descent, loss function, regularization, etc.

"With the availability of foundation models, ML knowledge is no longer a must-have for building AI applications."

2. Dataset engineering

Curating, generating, and annotating the data needed for training and adapting AI models.

In traditional **ML engineering**, most use cases are **close-ended**

- A model's output can only be among predefined values.
- For example, spam classification with only two possible outputs, "spam" and "not spam", is close-ended.

Foundation models, however, are **open-ended**

- Annotating open-ended queries is much harder than annotating close-ended queries
- It's easier to determine whether an email is spam than to write an essay.
- So data annotation is a much bigger challenge for AI engineering.

Models are now commodities: data will be the main differentiator

Training a model from scratch generally requires more data than finetuning, which, in turn, requires more data than prompt engineering.

3. Inference optimization

Making models faster and cheaper.

One challenge with foundation models is that they are often autoregressive:

- Tokens are generated sequentially.
- If it takes 10 ms for a model to generate a token, it'll take a second to generate an output of 100 tokens, and even more for longer outputs.

100 ms latency expected for a typical internet application is a huge challenge.

Inference optimization has become an active subfield in both industry and academia.

Table 1-4. How different responsibilities of model development have changed with foundation models.

Category	Building with traditional ML	Building with foundation models
Modeling and training	ML knowledge is required for training a model from scratch	ML knowledge is a nice-to-have, not a must-have[a]
Dataset engineering	More about feature engineering, especially with tabular data	Less about feature engineering and more about data deduplication, tokenization, context retrieval, and quality control
Inference optimization	Important	Even more important

[a] Many people would dispute this claim, saying that ML knowledge is a must-have.

Training, Pre-Training, Finetuning, and Post-Training

Training always involves changing model weights, but not all changes to model weights constitute training.

Pre-training refers to training a model from scratch: the model weights are randomly initialized.

- For LLMs, pre-training often involves training a model for text completion.
- Out of all training steps, pre-training is often the most resource intensive by a long shot.
- Pre-training also takes a long time to do.

Finetuning means continuing to train a previously trained model: the model weights are obtained from the previous training process.

- Because the model already has certain knowledge from pre-training, finetuning typically requires fewer resources (e.g., data and compute) than pre-training.

Post-training It’s finetuning when it’s done by application developers.

Application development

With traditional ML engineering, where teams build applications using their proprietary models, the model quality is a differentiation. With foundation models, where many teams use the same model, differentiation must be gained through the application development process. The application development layer consists of these responsibilities: evaluation, prompt engineering, and AI interface.

Evaluation

Evaluation is about **mitigating risks** and uncovering opportunities.

Evaluation is needed to select models, to benchmark progress, to determine whether an application is ready for deployment.

The challenges of evaluating foundation models chiefly arise from foundation models' open-ended nature and expanded capabilities.

Prompt engineering

Prompt engineering is about getting AI models to **express the desirable behaviors from the input alone**, without changing the model weights.

Prompt engineering is not just about telling a model what to do. It’s also about giving the model the **necessary context** and tools to do a given task.

For complex tasks with long context, you might also need to provide the model with a memory management system so that the model can keep track of its history.

Even with long context windows, **LLMs don’t reliably “remember” long histories** and it’s expensive to keep everything in the prompt. A **memory management system decides** what to keep, compress, retrieve, and forget across turns.

LLMs are **stateless per call**. They don’t persist history across requests. The only “memory” they have is the **current prompt** (ephemeral)

AI interface

AI interface means creating an interface for end users to interact with your AI applications.

- Copilot is commonly used as a plug-in in VSCode
- Grammarly is commonly used as a browser extension for Google Docs.
- Midjourney can either be used via its standalone web app or via its integration in Discord.