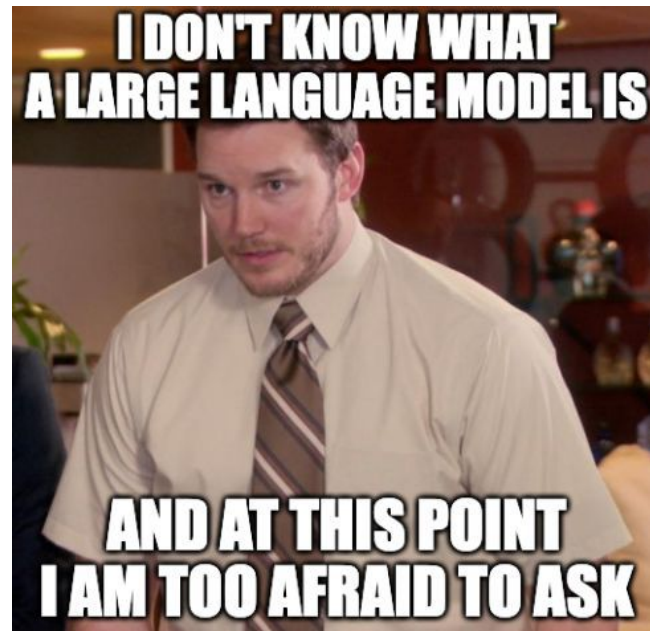


# Building LLM Powered Solutions

## Module 2:

## A very gentle intro to Large Language Models

Hamza Farooq



# Learning outcomes

- Context, as we know
- Intro to LLMs
- Transformer & General Architecture
- Introduction to ML System Design
- Future ML Architecture

## The value of context in NLP Models

The goal of natural language processing (NLP) is to find answers to four questions:

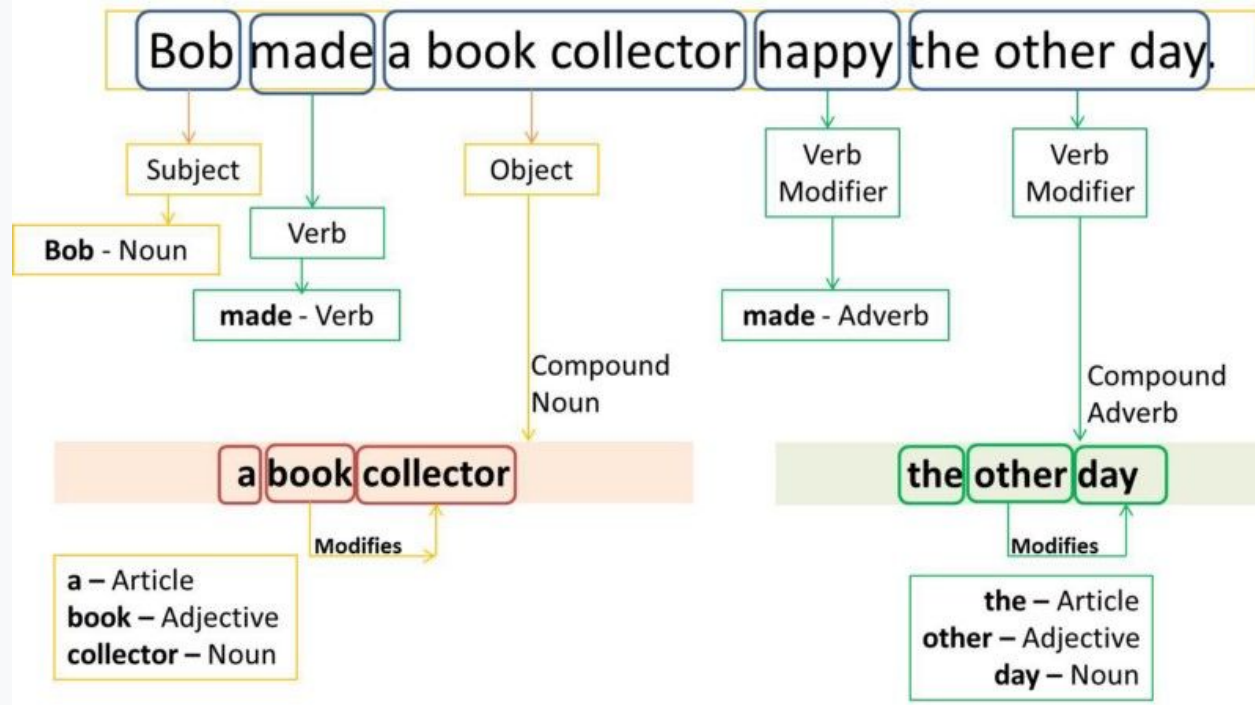
- Who is talking?
- What are they talking about?
- How do they feel?
- Why do they feel that way?

This last question is a question of context

## Content vs Context

Content is the material/matter/medium contained within the work that's available for audience.

Context is the positioning of the content, storyline or purpose that provides value to the audience.



**You shall know a word  
by the company it keeps.**

John Rupert Firth

British linguist specializing in contextual theories of  
meaning and prosodic analysis.

# What is a language model?

a language model refers to a type of model specifically designed to generate human-like text or predict the probability of a sequence of words. Language models learn patterns and statistics from large amounts of text data, enabling them to generate sensible and contextually appropriate sentences.

**In short...**

**The cat**

# Enter, Large Language Models

**Extensive Training Data:** Large language models are trained on vast amounts of text data, often comprising billions or even trillions of words. This extensive training data helps the models learn patterns, grammar, context, and a wide range of language nuances, enabling them to generate coherent and contextually appropriate responses.

**Complex Architectures:** Large language models employ complex architectures, such as transformer networks, that contain numerous layers and millions or even billions of parameters. These architectures enable the models to capture intricate language structures, understand semantics, and generate high-quality text by leveraging the vast amount of training data they have been exposed to. The large number of parameters allows the models to learn fine-grained details and provide nuanced responses.

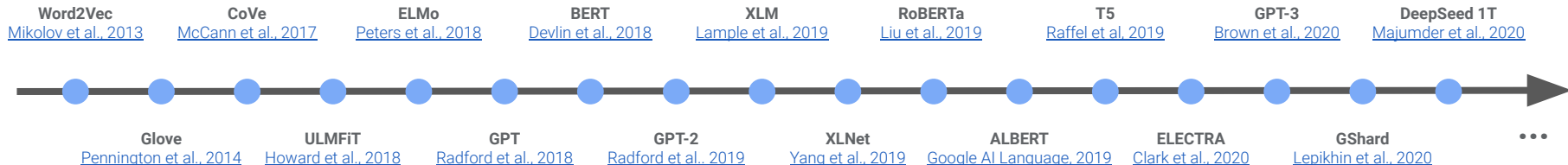
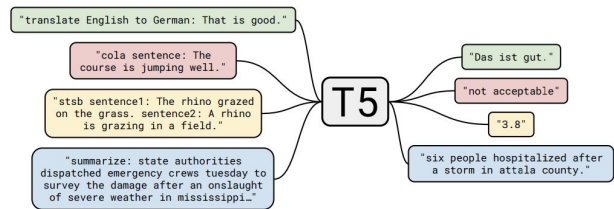
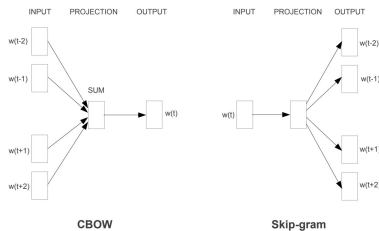
.

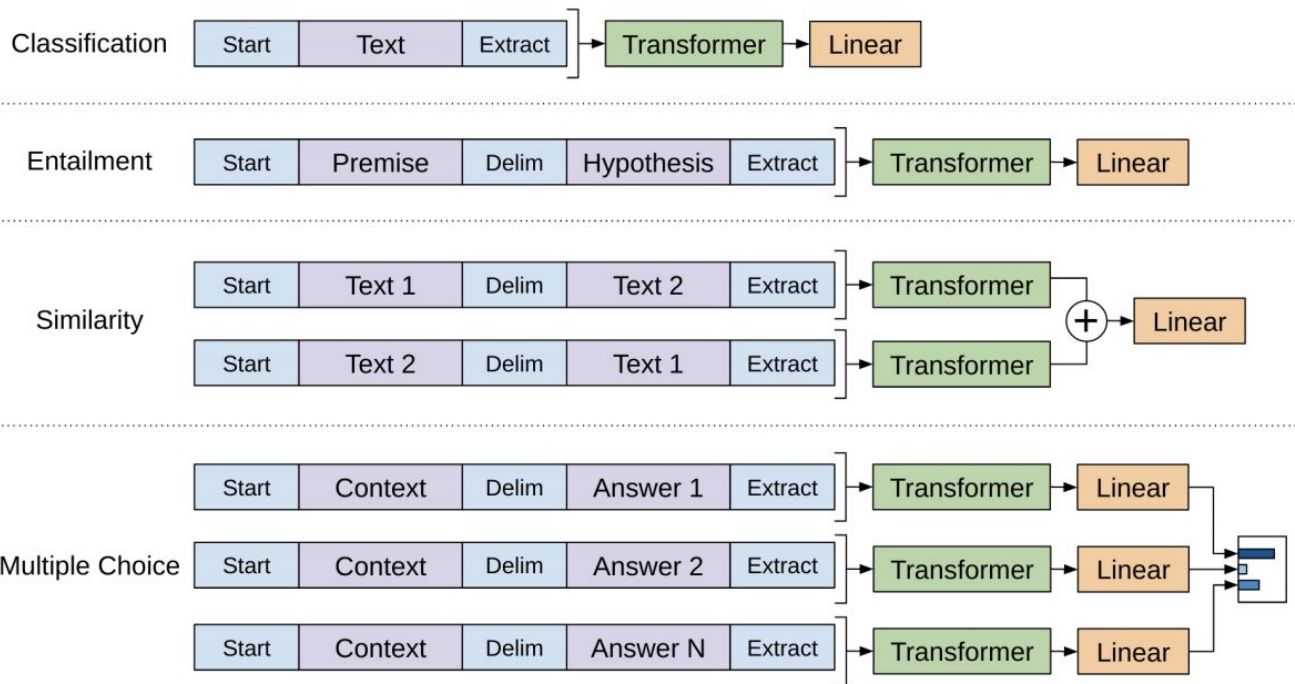
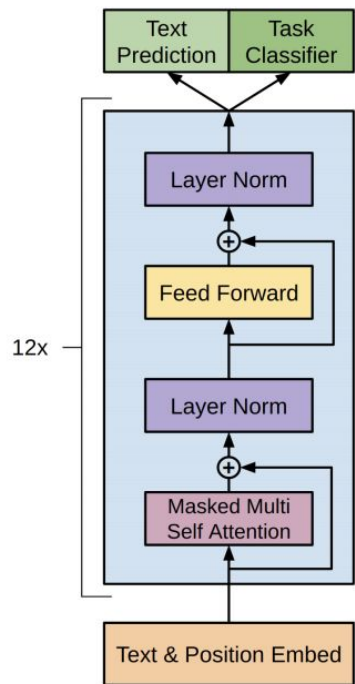


# What is a transformer?

a transformer is a type of deep learning model architecture that employs **self-attention** to capture relationships between words in a sequence, while a language model is a broader concept that encompasses models designed to generate text or predict the likelihood of word sequences. Transformers can be used as the underlying architecture for language models to enhance their performance and generate more accurate and coherent text.

# A brief history of pre-training in NLP





# (Statistical) Language Models

Try to predict tokens given a context.

- Masked LM:

We will image the upper and \_\_\_\_ eyelids to evaluate the structures.

- Left-to-right / causal LM

This is typically covered by most medical \_\_\_\_

- If we continue the left-to-right LM task indefinitely (autoregressive), we actually have a storyteller.

## Encoder-Only Models

- BERT
- Masked LM only (simpler task)
- Good at understanding, no generation capability

## Encoder-Decoder Models

- T5, LM-finetuned T5
- Span corruption (a variant of masked LM for decoder), and left-to-right LM
- Good at understanding, ok generation capability

## Decoder-Only Models

- GPT series, LaMDA, PaLM, ULM, Bard
- Left-to-right LM (mostly), combined with others
- Good at generation

# General Architecture

The architecture of Large Language Models primarily consists of multiple layers of neural networks, like

recurrent layers,

feedforward layers,

embedding layers, and

self attention layers.

These layers work together to process the input text and generate output predictions.

# Recurrent Layer

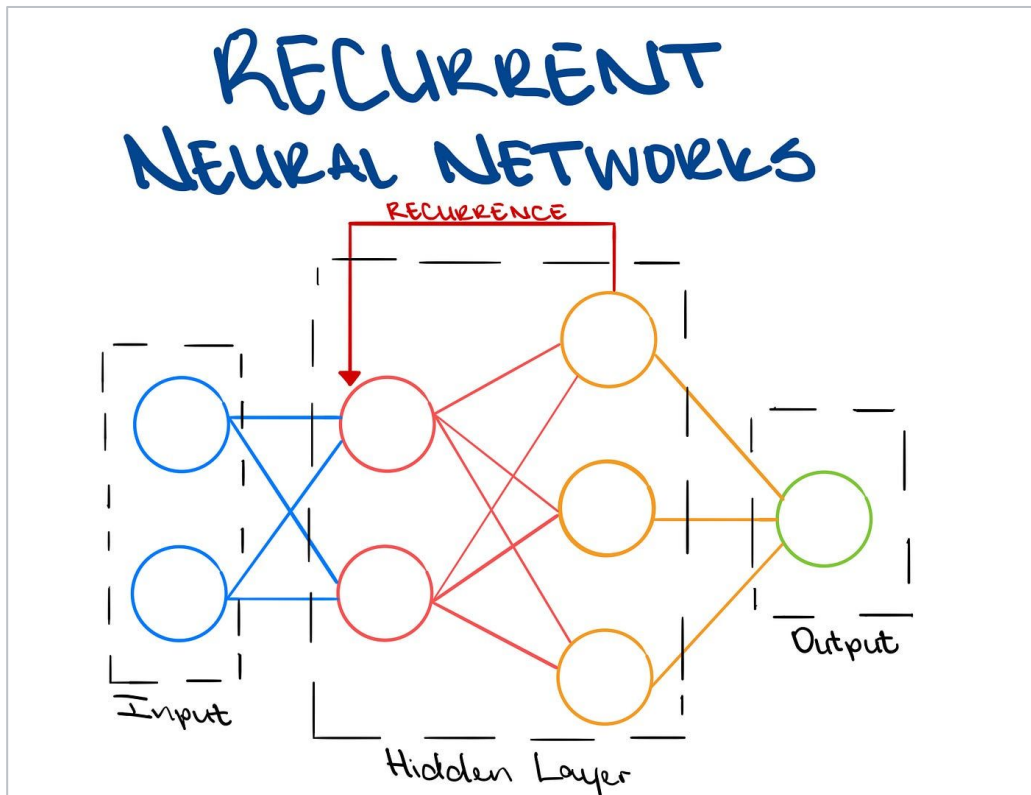
The recurrent layers of LLMs are designed to interpret information from the input text in sequence. These layers maintain a hidden state that is updated at each time step, allowing the model to capture the dependencies between words in a sentence.

# But what does it really mean??

- Recurrent layers are designed to capture **temporal dependencies** in sequential data.
- They maintain an internal state or memory that can retain information from previous time steps and pass it along to subsequent time steps.
- This memory allows the layer to process sequences with variable lengths and model the context and dependencies between elements in the sequence.



# Recurrent Layer

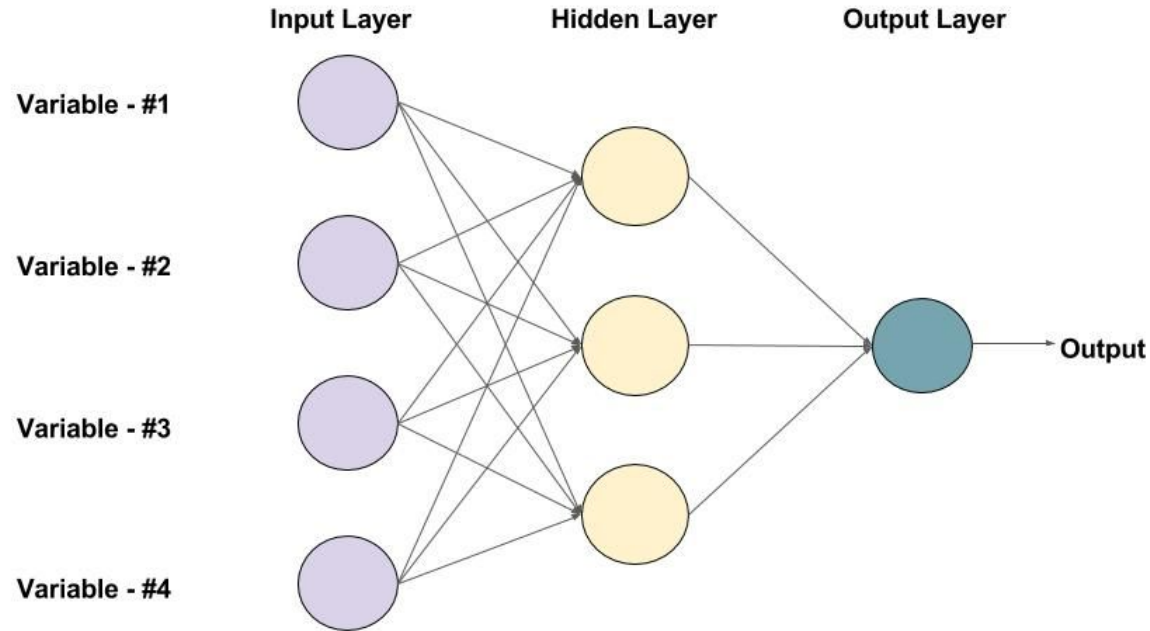


# Feedforward Layer

The feedforward layers of Large Language Models have multiple fully connected layers that apply nonlinear transformations to the input embeddings. These layers help the model learn higher-level abstractions from the input text.

Feedforward is a basic concept used in many areas, including artificial intelligence and neural networks. In simple words, feedforward refers to a process where information flows in a single direction, from the input to the output, without any feedback loops.

# Feedforward Layer

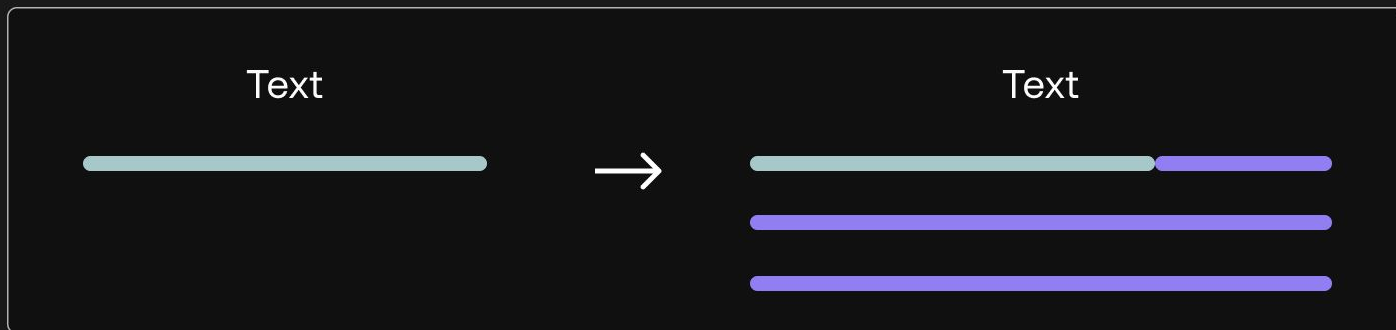


An example of a Feed-forward Neural Network with one hidden layer ( with 3 neurons )

# Embedding Layer

The embedding layer converts each word in the input text into a high-dimensional vector representation. These embeddings capture semantic and syntactic information about the words and help the model to understand the context.

## Text Generation



## Text Representation



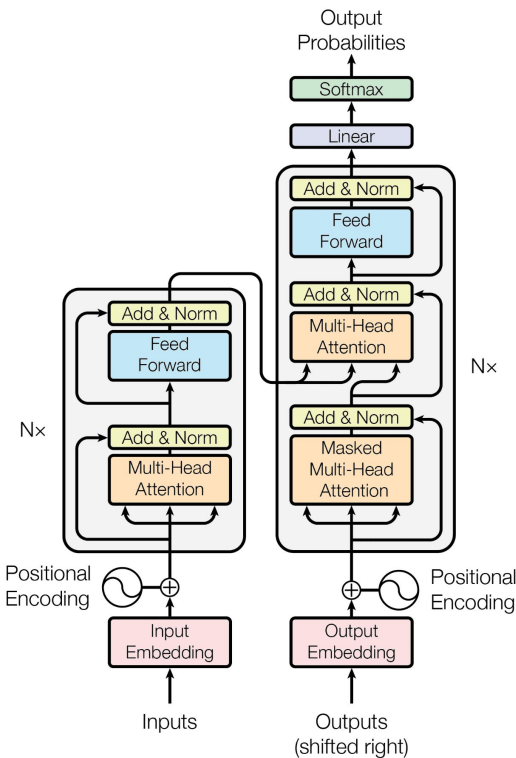
# Self Attention

Self-attention is a mechanism used to help understand and weigh the importance of different parts within a sequence of data.

It allows the model to focus on different elements and determine their relevance to each other without relying on fixed patterns or positions.

## Encoder

- $\Leftrightarrow$  Understanding / Classification / Regression / Sequence Labeling
- Given a textual inputs, produce hidden representations
  - Eg. embedding, topic distribution, class likelihood, etc.

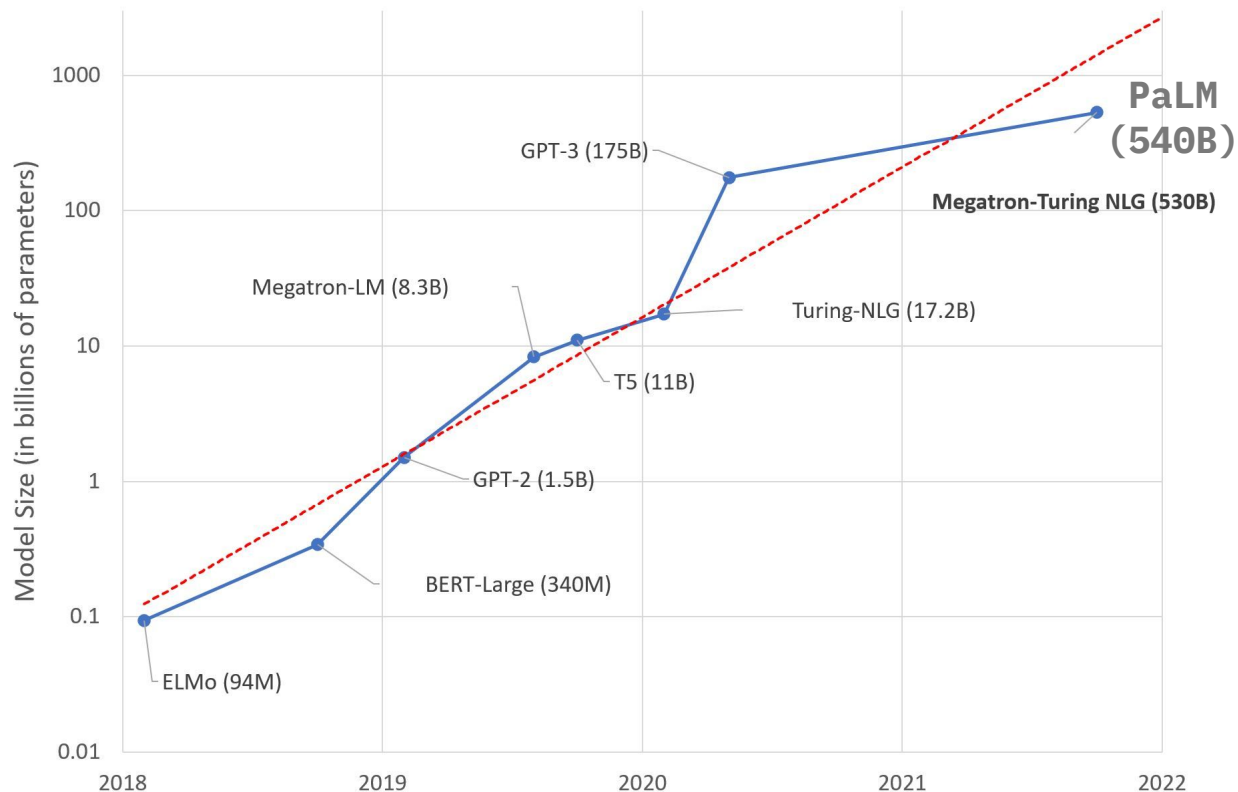


## Decoder

- $\Leftrightarrow$  Generation / Summarization
- Given a hidden representation, produce the textual outputs.
- Note: Understanding tasks can also be modeled as generation task

LLMs continue to grow  
at an unprecedented  
rate!

From 2018 to 2022,  
Language models have  
grown by **600,000%**



Source: <https://huggingface.co/blog/large-language-models>



# Parameters

Parameters refer to the learnable elements or variables within the model.

These parameters capture the knowledge and patterns learned during training and are used to make predictions or generate text.

They represent the internal representations and connections between the different components of the model, such as the hidden layers and attention mechanisms.

# Parameters: Explain like I'm 5

Imagine you have a very smart robot that can understand and speak in different languages.

To make this robot understand and respond correctly, we need to give it some instructions and knowledge.

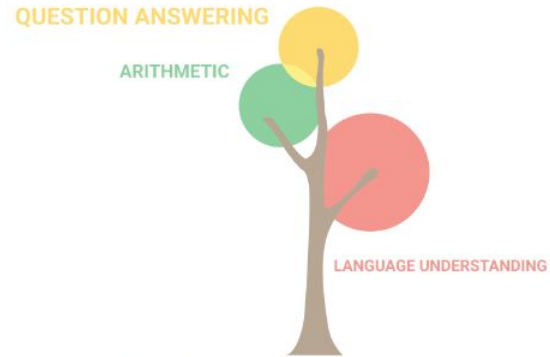
Parameters in large language models are like little pieces of information that the robot learns and remembers during its training.

# Parameters: Explain like I'm 5

The more parameters the robot has, the more information it can remember and use to understand and speak languages.

It's like having a bigger memory or a bigger brain!

With more parameters, the robot can become smarter and perform better at tasks like translating languages, answering questions, or even having a conversation with you.



**8 billion parameters**

Use Cases continue to grow too

# The rise of startups offering LLM services

Numerous players have come up in from open-source to closed source

Generally speaking, we have 5 classes of application:

Classification, Response Generation, Text Generation, Translation and Knowledge Answering

	Publicly Available	Commercial License	Open	Embeddings Available	Fine-Tuning Available	Text Generation	Classification	Response Generation	Knowledge Answering (KI-NLP)	Translation	Multi-Lingual
OpenAI (GPT3)	●	●	●	●	●	●	●	●	●	●	●
Cohere	●	●	●	●	●	●	●	●	●	●	●
GooseAI	●	●	●	●	●	●	●	●	●	●	●
EleutherAI	●	●	●	●	●	●	●	●	●	●	●
AI21labs	●	●	●	●	●	●	●	●	●	●	●
Bloom	●	●	●	●	●	●	●	●	●	●	●
LaMDA	●	●	●	●	●	●	●	●	●	●	●
BlenderBot	●	●	●	●	●	●	●	●	●	●	●
DialogPT	●	●	●	●	●	●	●	●	●	●	●
GODEL	●	●	●	●	●	●	●	●	●	●	●
NLLB	●	●	●	●	●	●	●	●	●	●	●

# The rise of Vector Databases

A collection of vector database startups have raised close to \$200M in funding, resulting in new enterprise solution providers and proponents.

## Popular Vector Databases

 Meta Faiss

"library for efficient similarity search and clustering of dense vectors"




"Enabling High Performance Billion-Scale Similarity Search"

 milvus

"Vector database built for scalable similarity search"



 Pinecone

"makes it easy to build high-performance vector search applications"



Vald

"highly scalable distributed fast approximate nearest neighbor dense vector search engine"



 vespa

"full-featured text search engine and supports both regular text search and fast approximate vector search"



Weaviate

"an open source vector search engine that stores both objects and vectors"



 elastic

"Meet enterprise infrastructure needs with standalone or embeddable search, regardless of data type, to power critical user experiences"



open source

# How it all started...

The Transformer is an architecture that uses Attention to significantly improve the performance of deep learning NLP translation models.

It was first introduced in the paper [Attention is all you need](#) by Google, in 2017

# Resources

[Wordy](#), but awesome

[Awesome](#) and easy to watch

Should have been the [first one](#)

[Foundational Models](#)



# What is life without code?

[Colab](#)

When it's been 7 hours and you still  
can't understand your own code



# Let's shift gears: Introduction to ML System Design

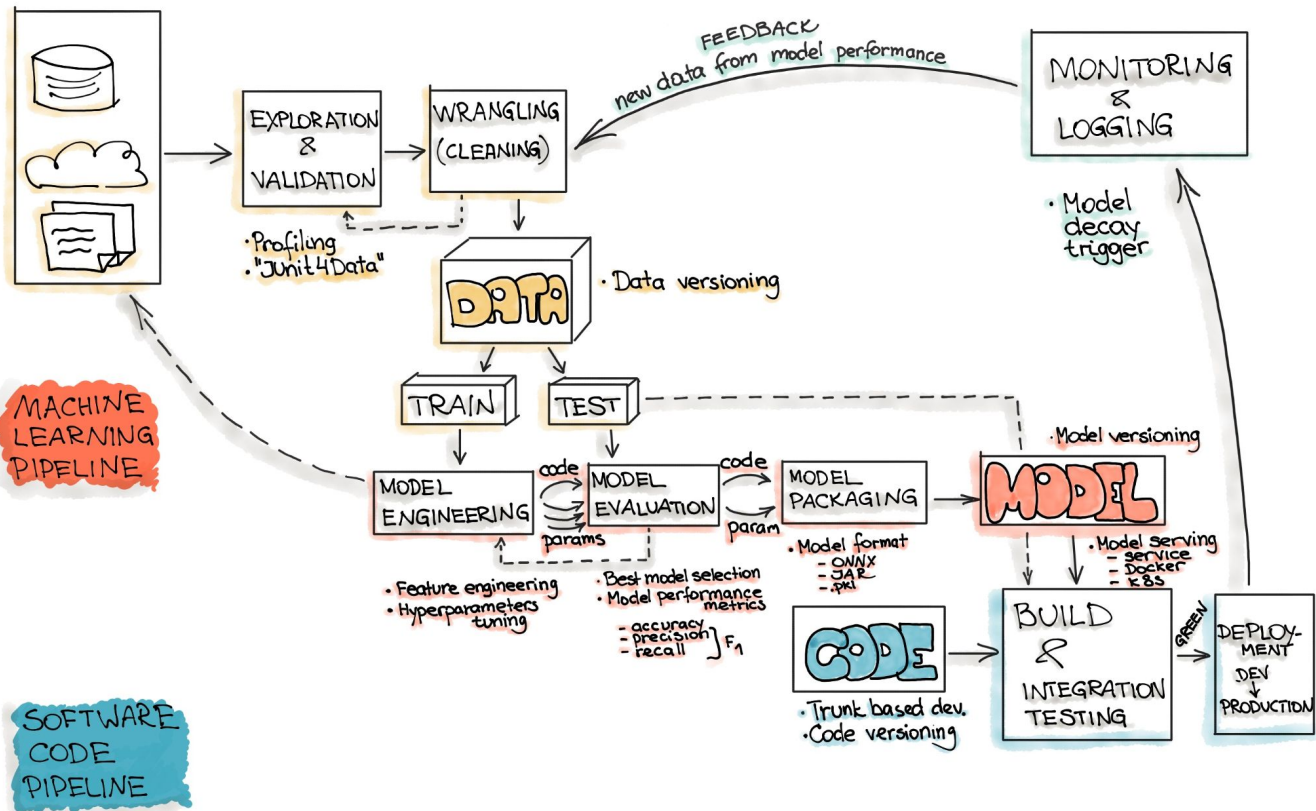


# What does it really mean?

Machine learning system design refers to the process of creating and developing a system that utilizes machine learning algorithms to solve a specific problem or task.

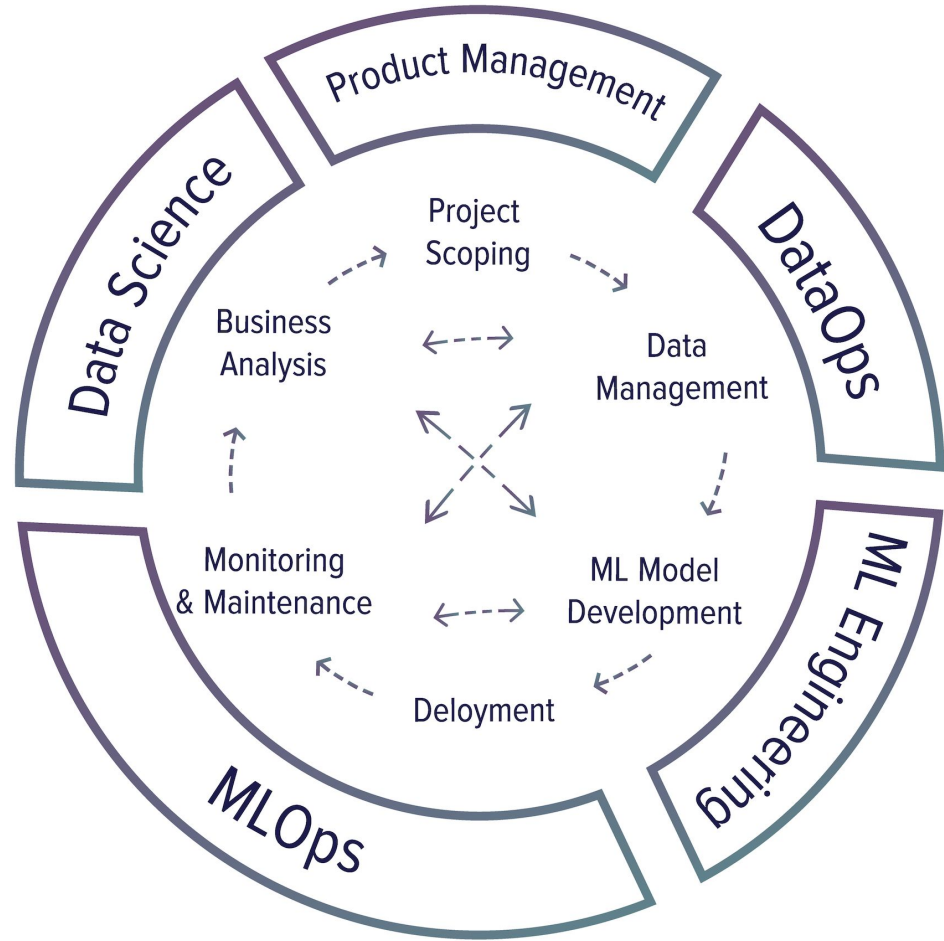
# MACHINE LEARNING ENGINEERING

## DATA PIPELINE



The horror

## Simplified Version



# In summary...

machine learning system design involves

- problem identification,
- data preparation,
- model selection and
- training, evaluation and optimization,
- deployment, and
- ongoing monitoring.

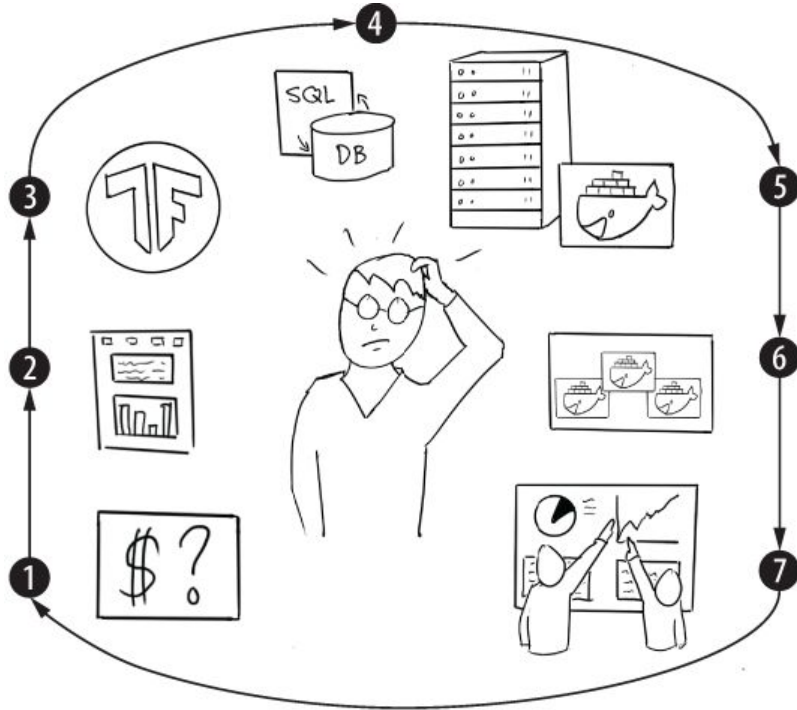
It is a cyclical process aimed at creating effective and efficient systems that leverage machine learning techniques to solve real-world problems.

# Word of caution

We want to stay away from software system design because that is a whole new paradigm

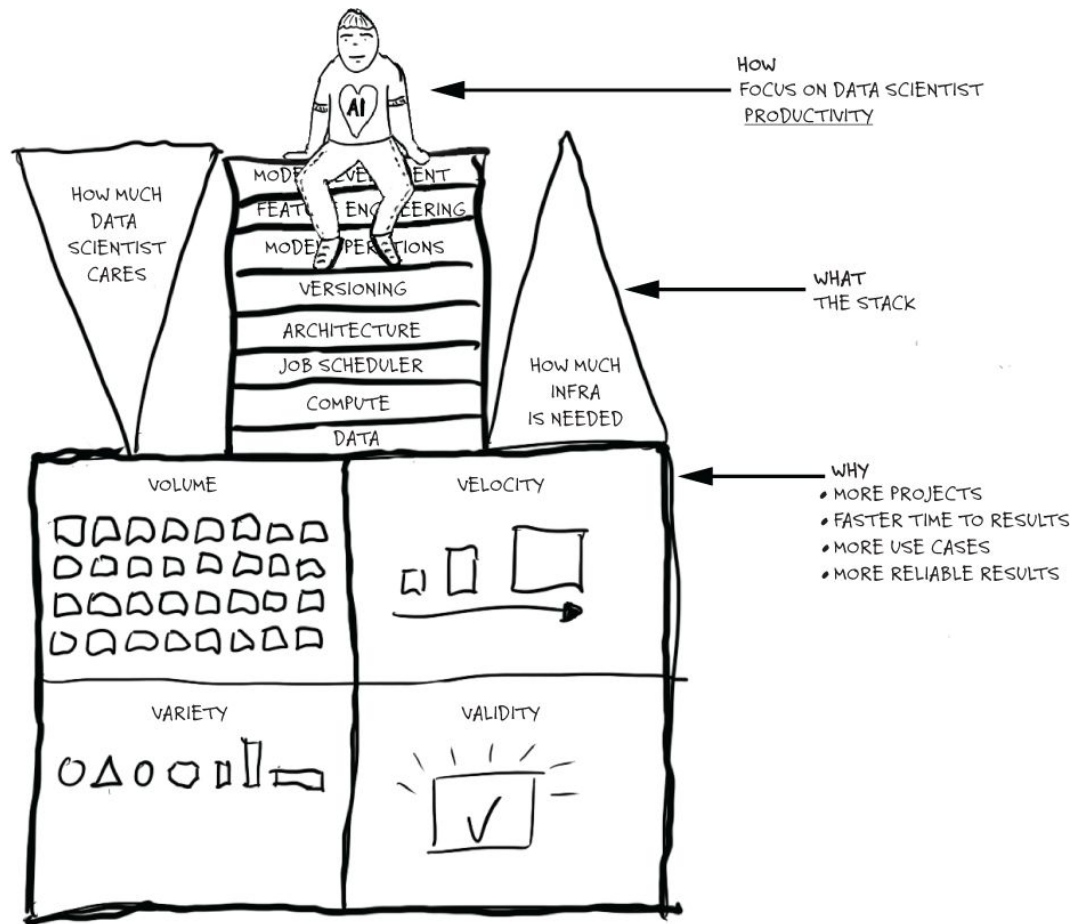
# Why we it all in the first place?

Life cycle of a Data Science Project

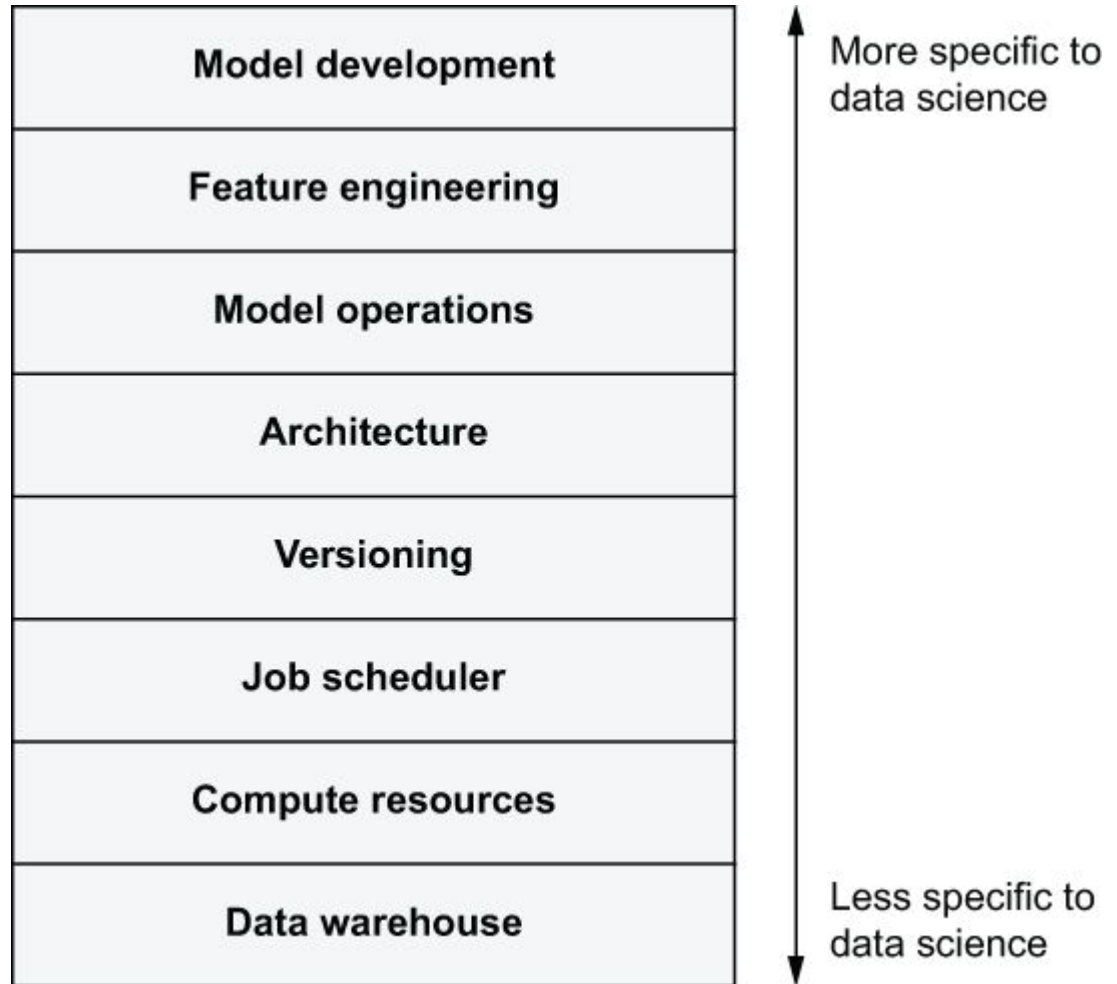


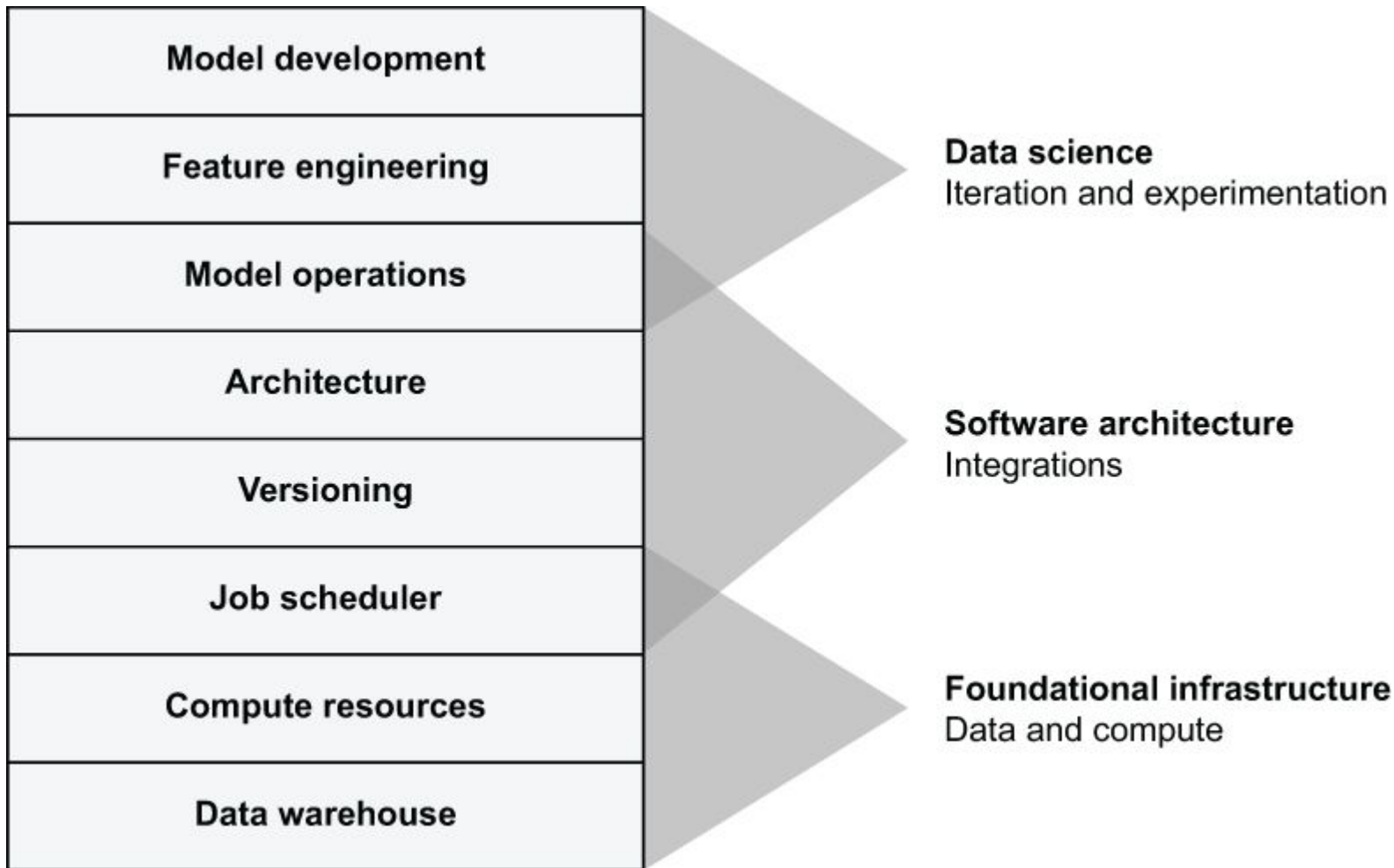


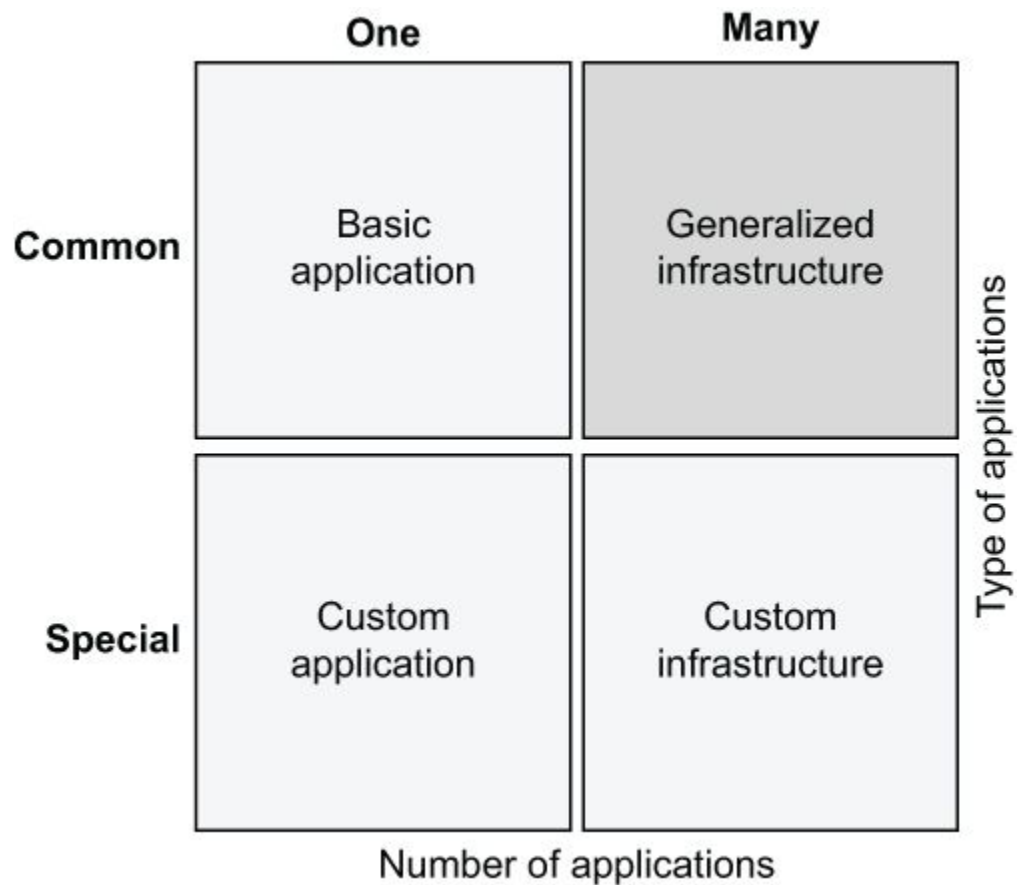
# Data Scientists care less about infra and everything else



**Because  
sometimes it is  
not their job**





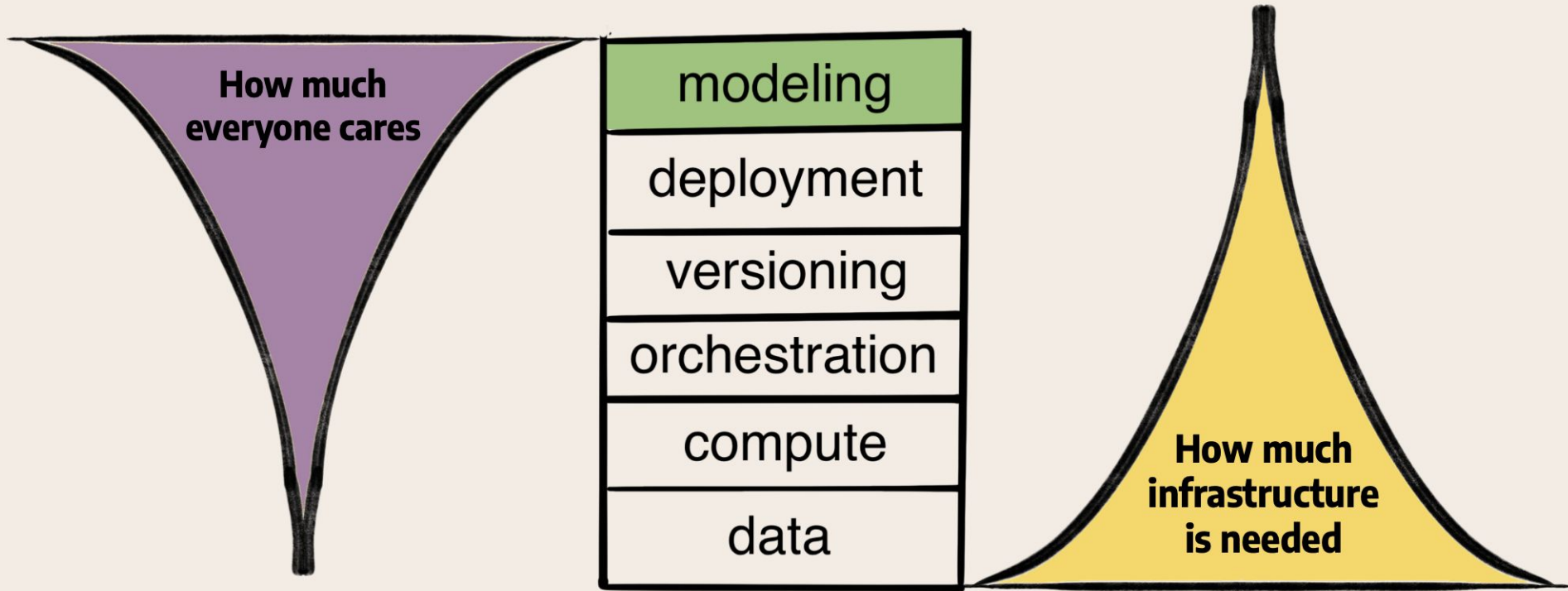


# How do LLMs change all this?

*Adoption of LLMs and Foundation Models:* LLMs and other foundation models are expected to be embraced as a **potent addition to the machine learning (ML) stack**, complementing existing models rather than replacing them.

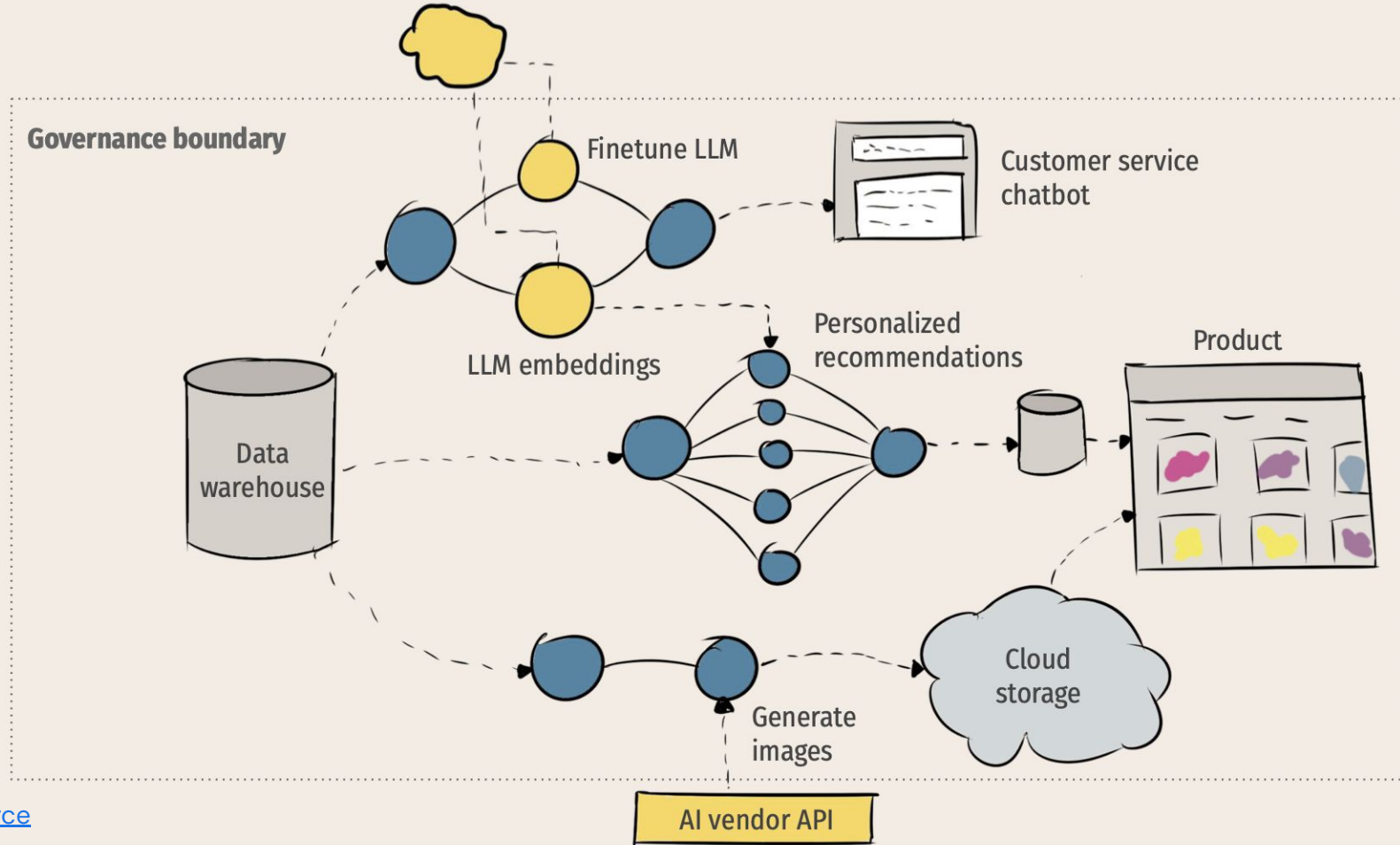
*Consistency in Technology Adoption:* Companies are likely to maintain their existing approaches to technology adoption when incorporating LLMs and foundation models, as there is no compelling reason to suggest otherwise.

# LLMs are just one part of the larger system



**What should LLM Powered ML  
Systems look like...**

## Open-source foundation model





**Thank you.**

# Appendix