Machine Learning for Advanced AI Search

Sowjanya Pandruju

Cloud Application Architect
Amazon Web Services

What is generative artificial intelligence (AI)?

Creates new content and ideas, including conversations, stories, images, videos, and music

Powered by large models that are pre-trained on vast corpora of data and commonly referred to as foundation models (FMs)

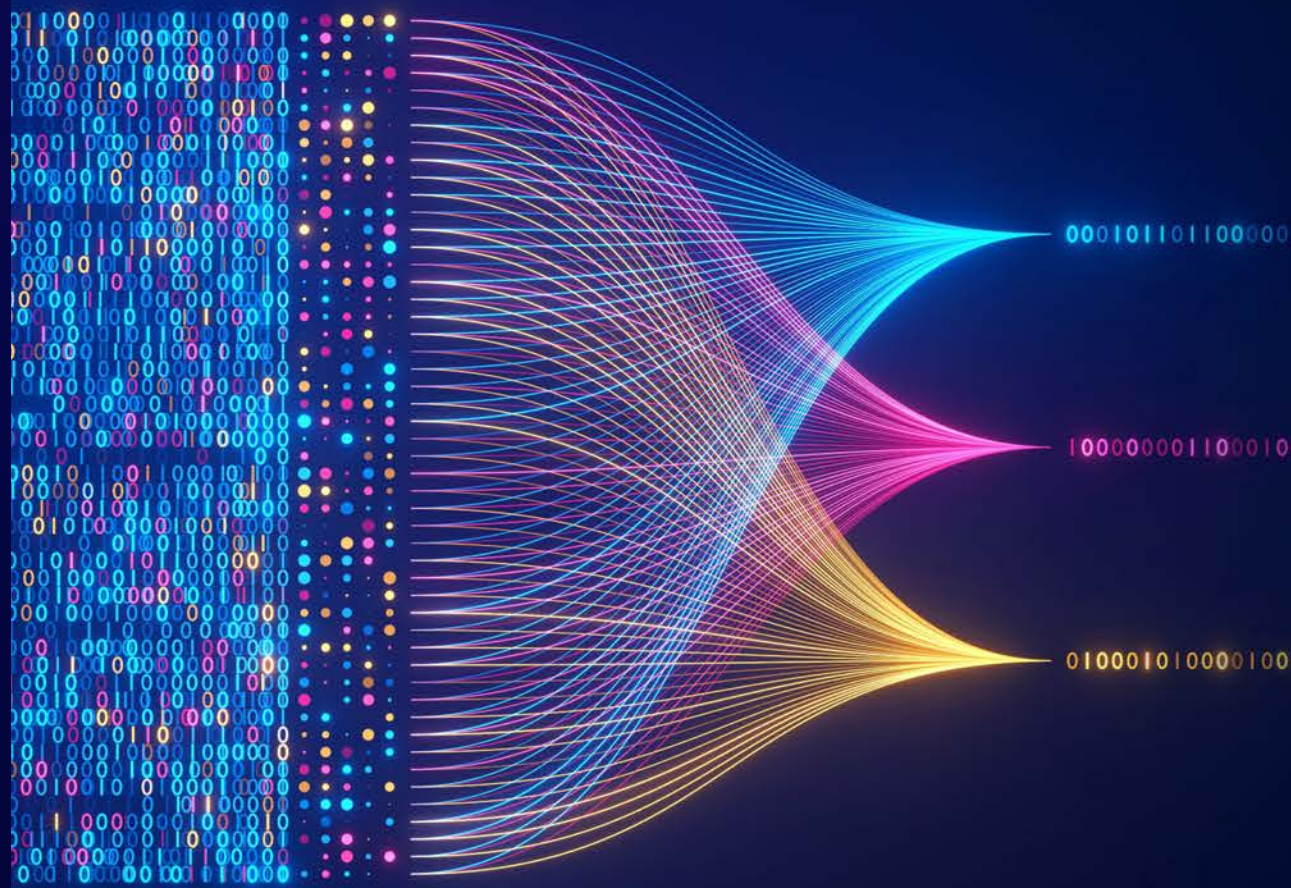
Generative AI is powered by foundation models

Pre-trained on vast amounts of unstructured data

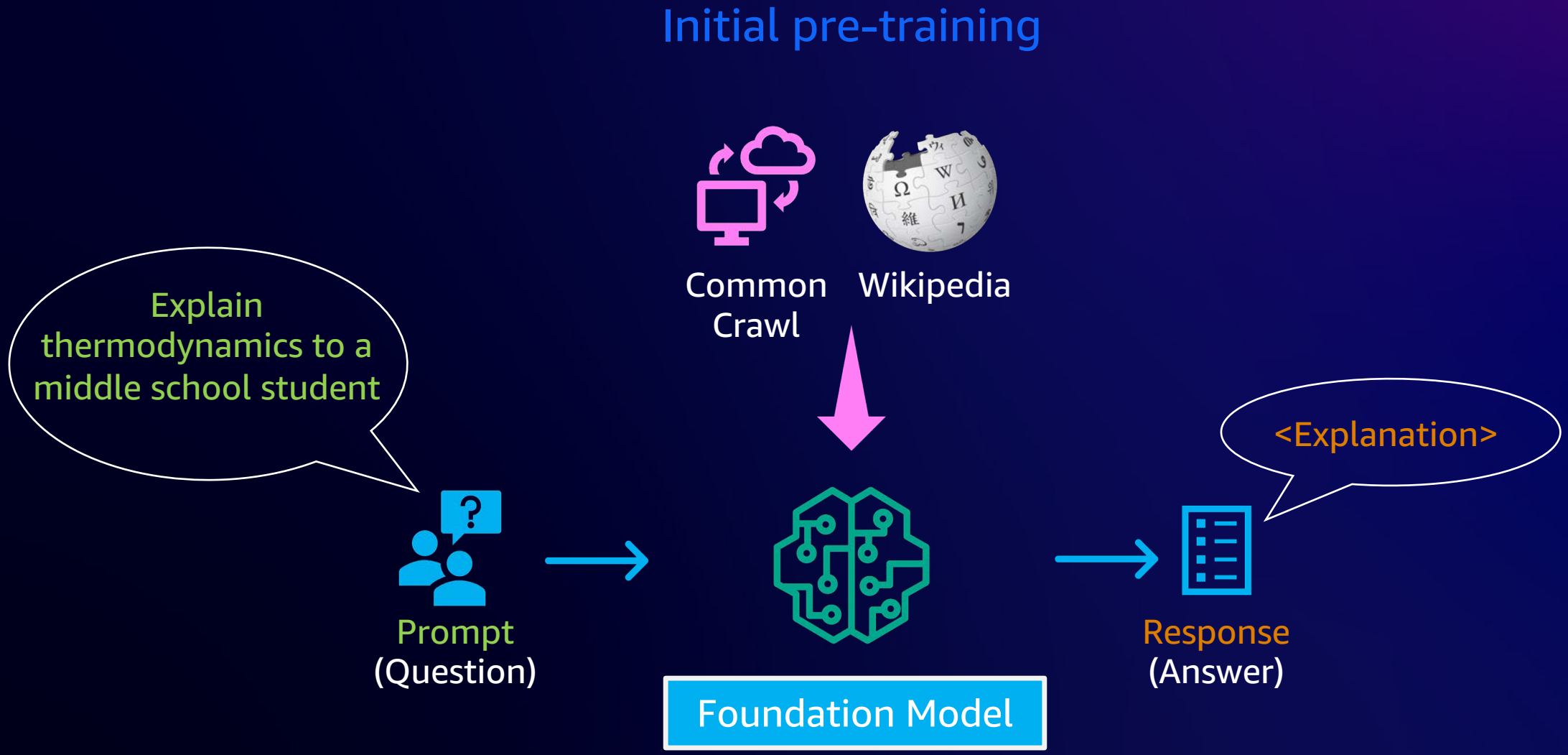
Contains large number of parameters which makes them capable of learning complex concepts

Can be applied in a wide range of contexts

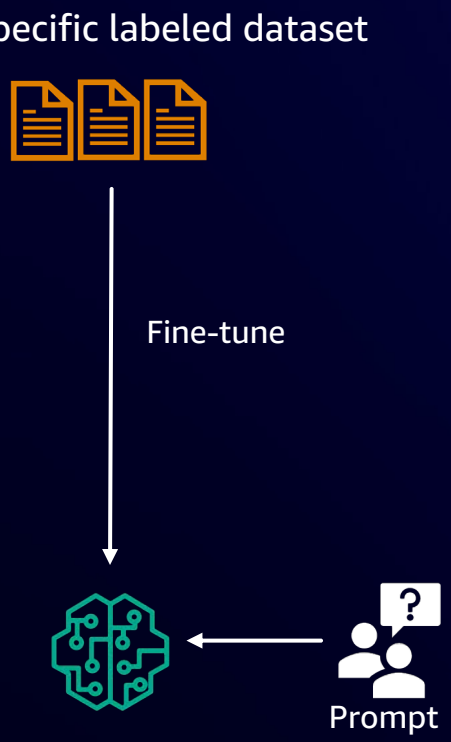
Customize FMs using your data for domain specific tasks



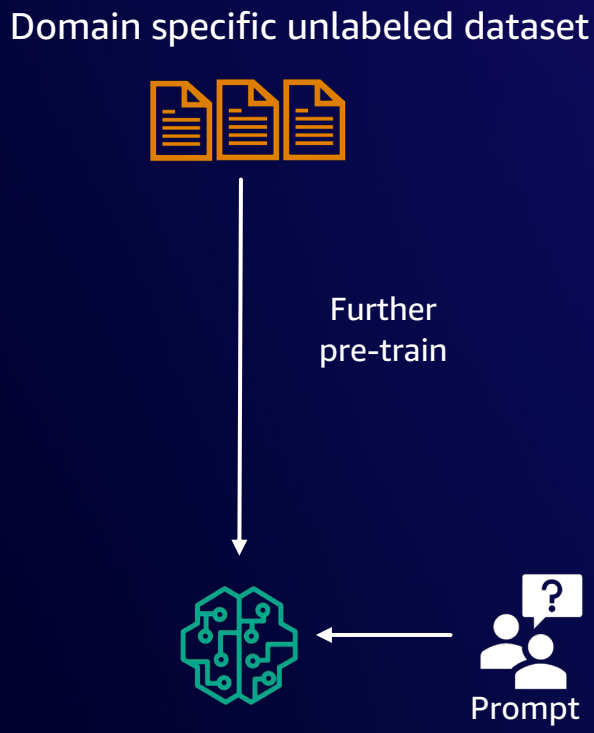
What are inputs & outputs of foundation models?



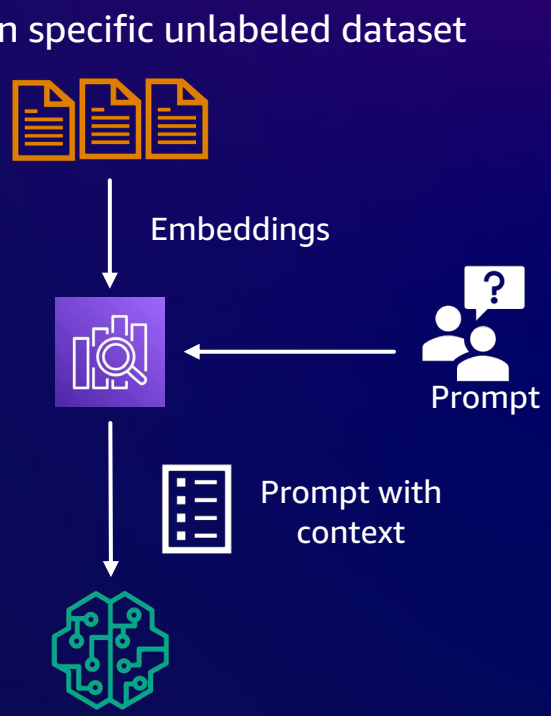
How can we customize a foundation model?



Instruction-based fine-tuning



Domain adaptation



Information Retrieval

Vectors

- A **Vector** is simply an **ordered list of numbers**.
- Vectors are often used to **represent features** of data points.
- Vectors turn real-world data into a format (mathematical objects) that machine learning models can understand, compare, and manipulate efficiently.

What is a vector embedding?

- A numerical representation of words or sentences, used in NLP
- NLP models can easily perform tasks such as querying, classification, and applying machine learning algorithms on textual data.
- They preserve important properties like similarity and structure.

Vector store

RAW Data

Documents

Images

Audio

Machine Learning Model (Embedding)

Amazon Bedrock

Dense Vector Encodings

0.3, 2.1, 0, 0.9, 1.0,...

1.3, 8.1, 0, 4.6, 3.6,...

7.3, 1.1, 0, 2.9, 1.0,...

Vector Embedding Space

Vector Database

Vector datastore in AWS?

Looking for low-code no-code solution and rapid deployment

Use out-of-the-box 40+ connectors to ingest data from many data sources

Do not want to deal with management of data chunking, embeddings, and indexing algorithm choices

Already using Amazon OpenSearch Service and comfortable with NoSQL

Need low latency with HNSW algorithm, and lower memory with product quantization

Need higher search accuracy using larger vector dimensions; flexibility to pick any vector embedding model

Already using Amazon RDS/Aurora PostgreSQL and prefer using SQL

Keep application and AI/ML data and vectors in the same DB for better governance, faster clones

Need transactional and immediate consistency

Role Based Access Controls, Encryption, Authentication, Authorization, Auditing, and Fully Managed Services with Serverless Option

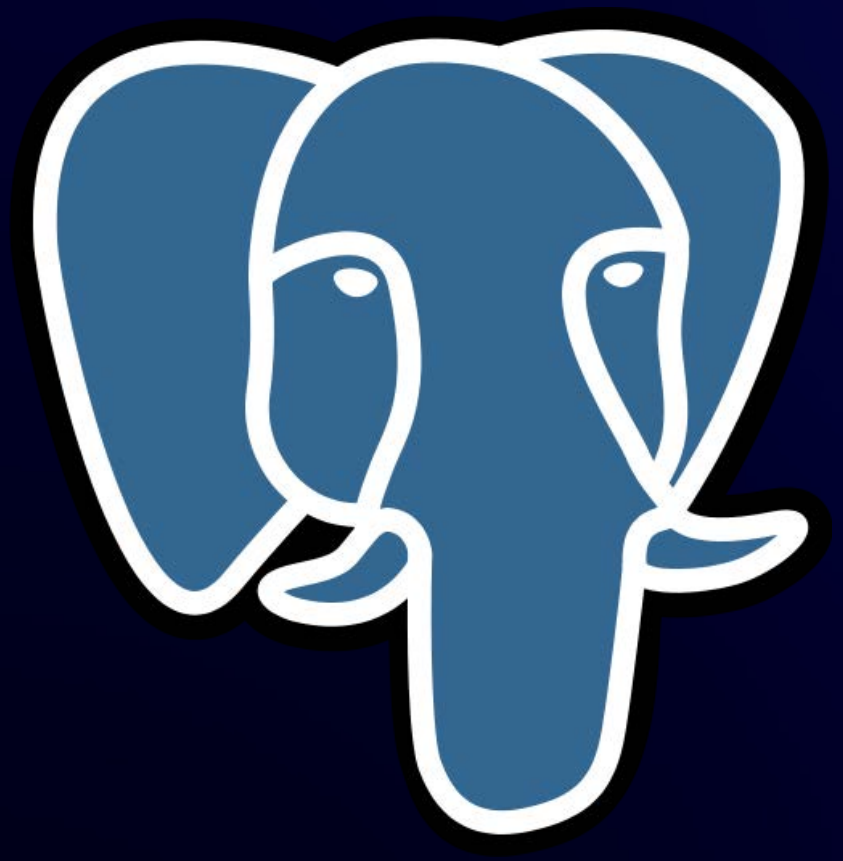
Amazon Kendra

Amazon OpenSearch Service (Serverless)

Amazon Aurora

Amazon RDS

Why PostgreSQL?



Open source

- Active development for more than 35 years
- Controlled by a community, not a single company

Performance and scale

- Robust data type implementations
- Extensive indexing support
- Parallel processing for complex queries
- Native partitioning for large tables

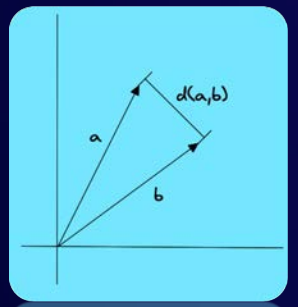
pgvector: An open-source library for vector search

- A Vector data type
- Similarity search
- Seamless SQL integration

pgvector example: Querying nearest neighbor

Euclidean (L2)

Useful for counts/measurements recommendation systems

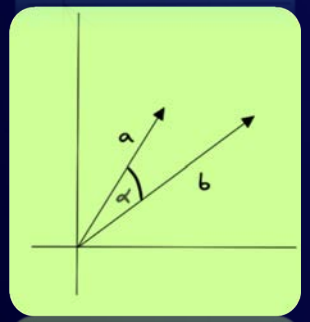


$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

<->

Cosine similarity

Useful for semantic search and document classification

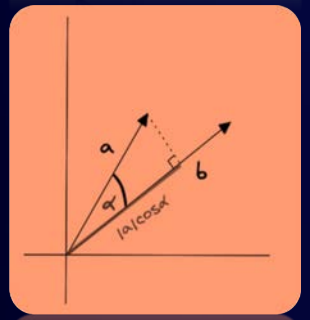


$$sim(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||}$$

<#>

Dot product

Useful for collaborative filtering



$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|cos\alpha$$

<=>

pgvector example: Querying nearest neighbor

```
CREATE TABLE test_embeddings(product_id bigint, embeddings vector(3));

INSERT INTO test_embeddings VALUES
(1, '[1, 2, 3]'), (2, '[2, 3, 4]'), (3, '[7, 6, 8]'), (4, '[8, 6, 9]’);

SELECT product_id, embeddings, embeddings <-> '[3,1,2]' AS distance
     FROM test_embeddings
     ORDER BY embeddings <-> '[3,1,2]’ limit 2;

------------+------------+--------------------
product_id  | embeddings |     distance
------------+------------+--------------------
          1 | [1,2,3]    | 2.449489742783178
          2 | [2,3,4]    |                 3
(2 rows)
```

Thank you!