# Foundation Models and Large Language Models

*Generative AI*

Module 1 – Lesson 2

# Today’s activities

- Traditional ML
- Foundation models and LLMs
- Transformers Architecture
- Challenges and Limitations of LLMs

# Review: Traditional ML (1/2)

- Trained on task-specific data
  - Models specialized for the task
- Training typically starts from scratch
- Several model choices
  - Tree-based models, linear models, neural networks, etc.

# Review: Traditional ML (2/2)

- Optimized for one task
  - Classification, regression, clustering
- Challenging to adapt to another similar task

# Review: Machine learning terminology

**ML: Train a computer to recognize patterns in historical data to make predictions on new data**

**The ML algorithm takes a random function and refines it until the features predict the correct label.**

**model**

**f**

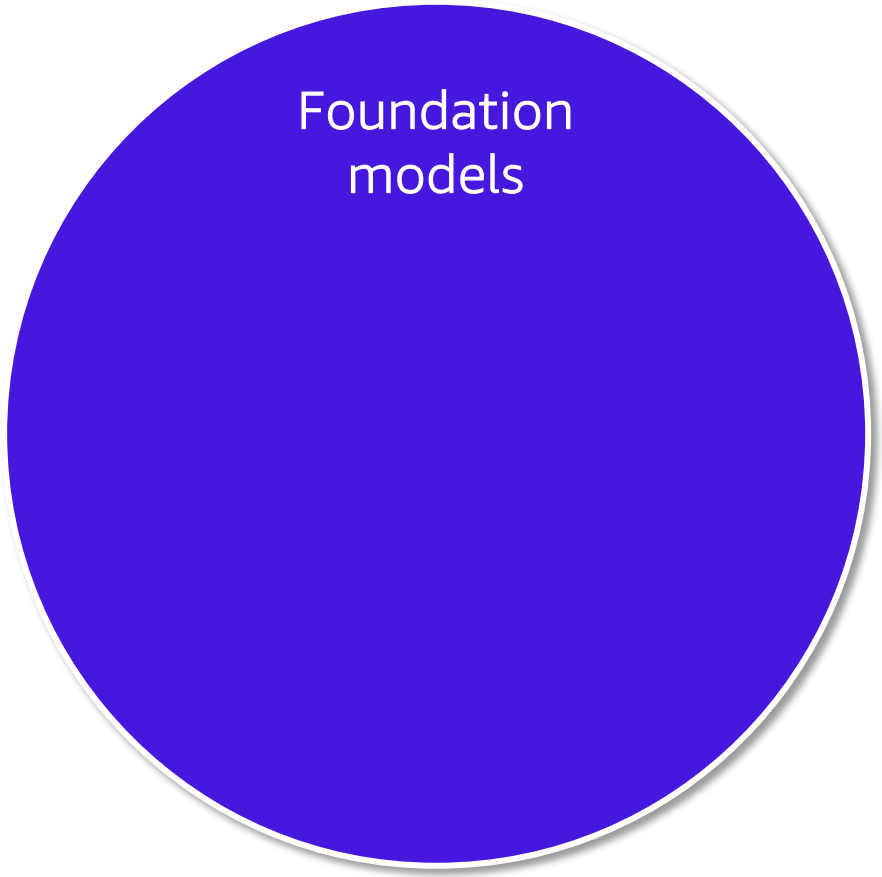
**features**

| Number of logins | Watched Prime Video | … | Number of purchases |
|---|---|---|---|
| 120 | Yes | … | 4 |
| 1 | No | … | 0 |
| 219 | No | … | 12 |
| 57 | Yes | … | 2 |
| 0 | No | … | 0 |
| 23 | Yes | … | 1 |

**label**

| Prime signup after free trial |
|---|
| Yes |
| No |
| Yes |
| Yes |
| No |
| No |

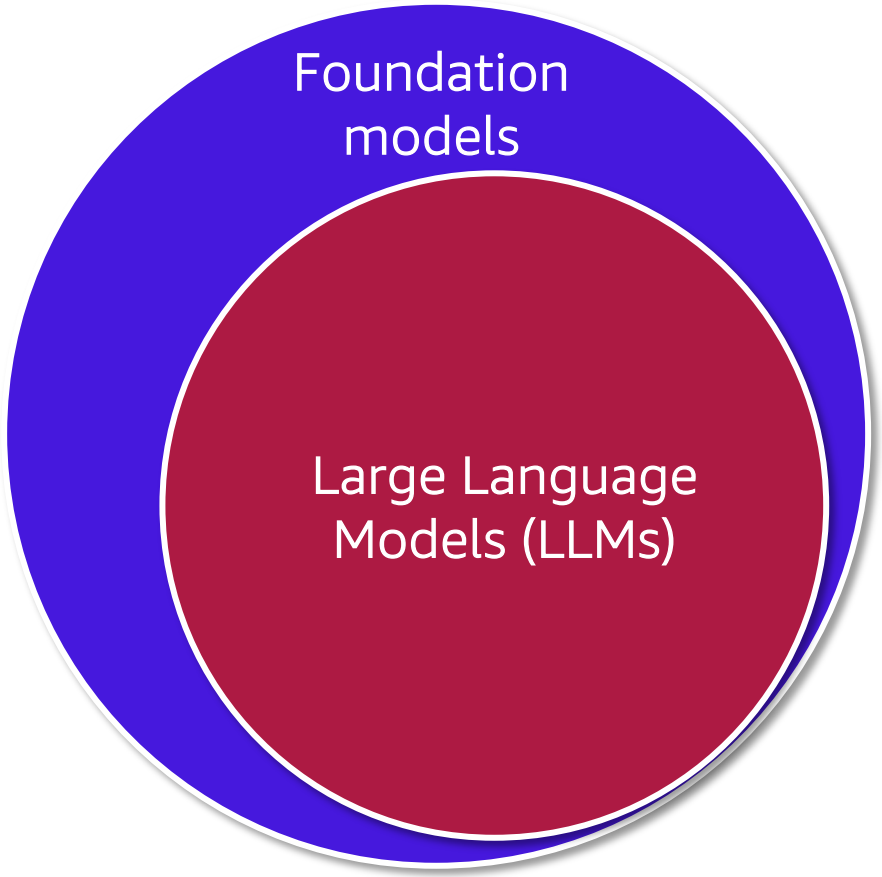
# Review: Foundation models



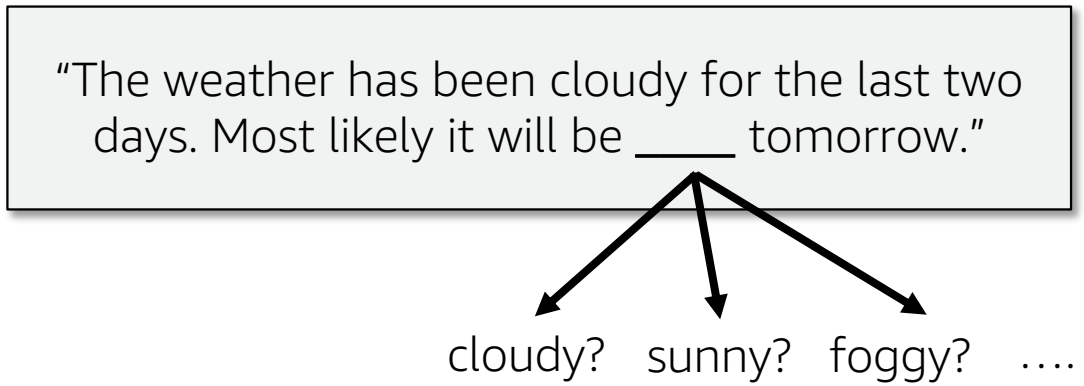

- Large ML models that are **pre-trained** with **vast amounts of data**. These can be **adapted** to more specialized tasks.

- Can be trained on any kind of data
  - Text
  - Images
  - Video
  - Audio
  - Etc…

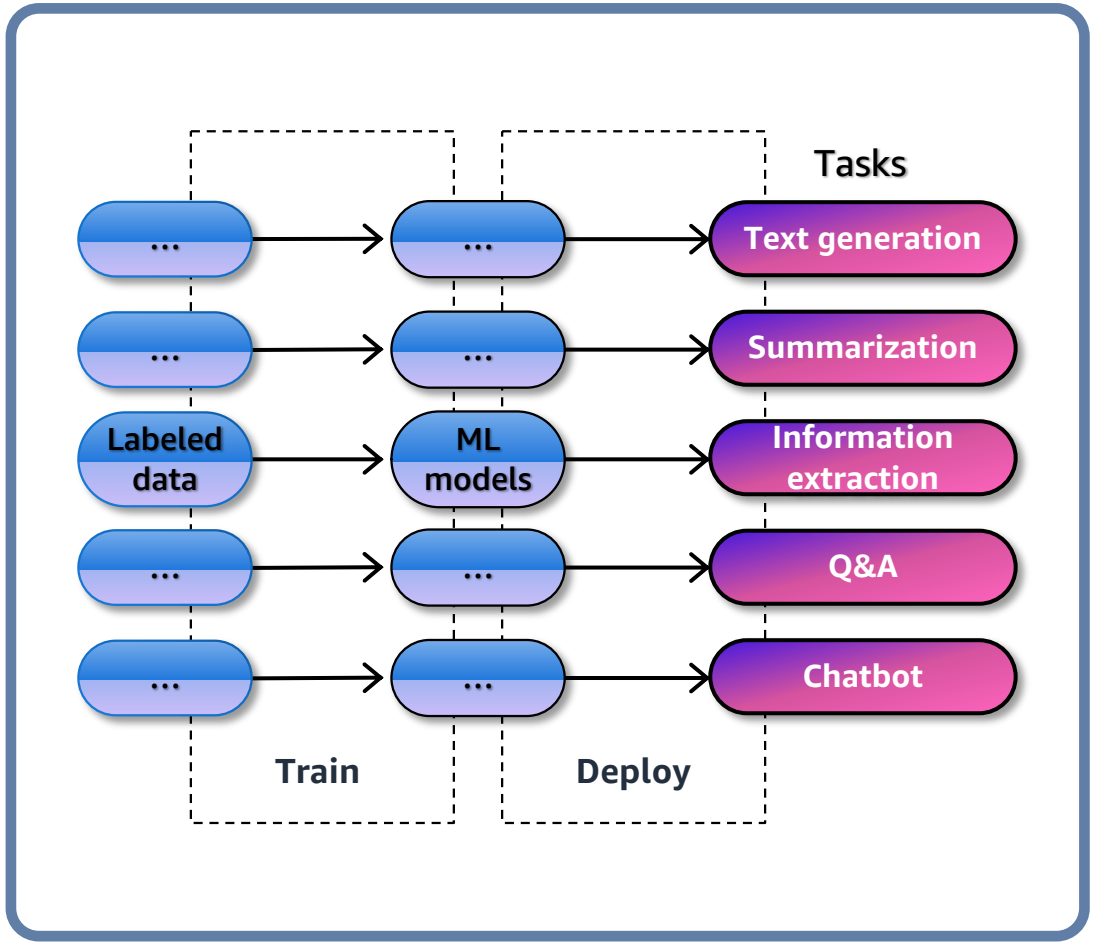
# Review: Large Language Models (LLMs)



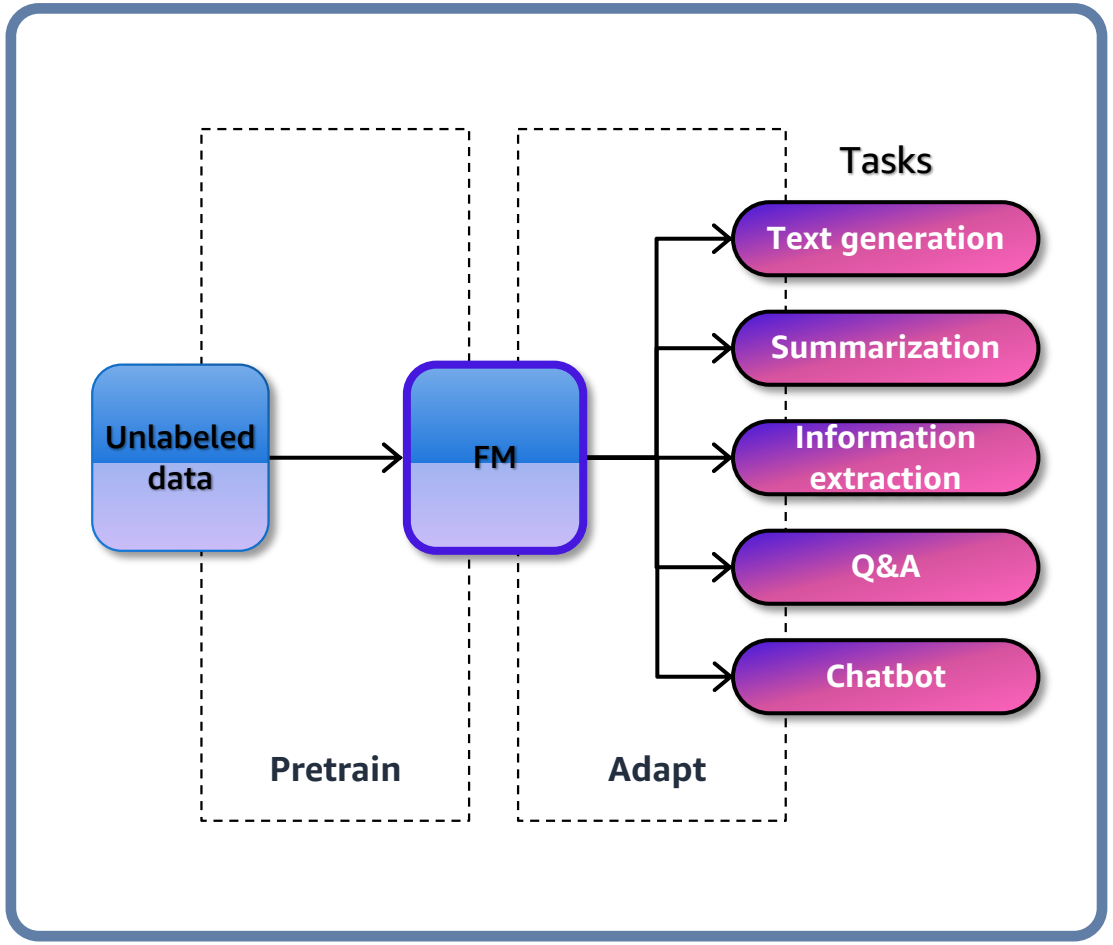

- Foundation models trained on **text.**
- Large ML models that learn the **probabilities of words** being used in certain contexts.
- **Training task:** Learn to predict the missing word in a text sequence.



# Foundation models (FMs)

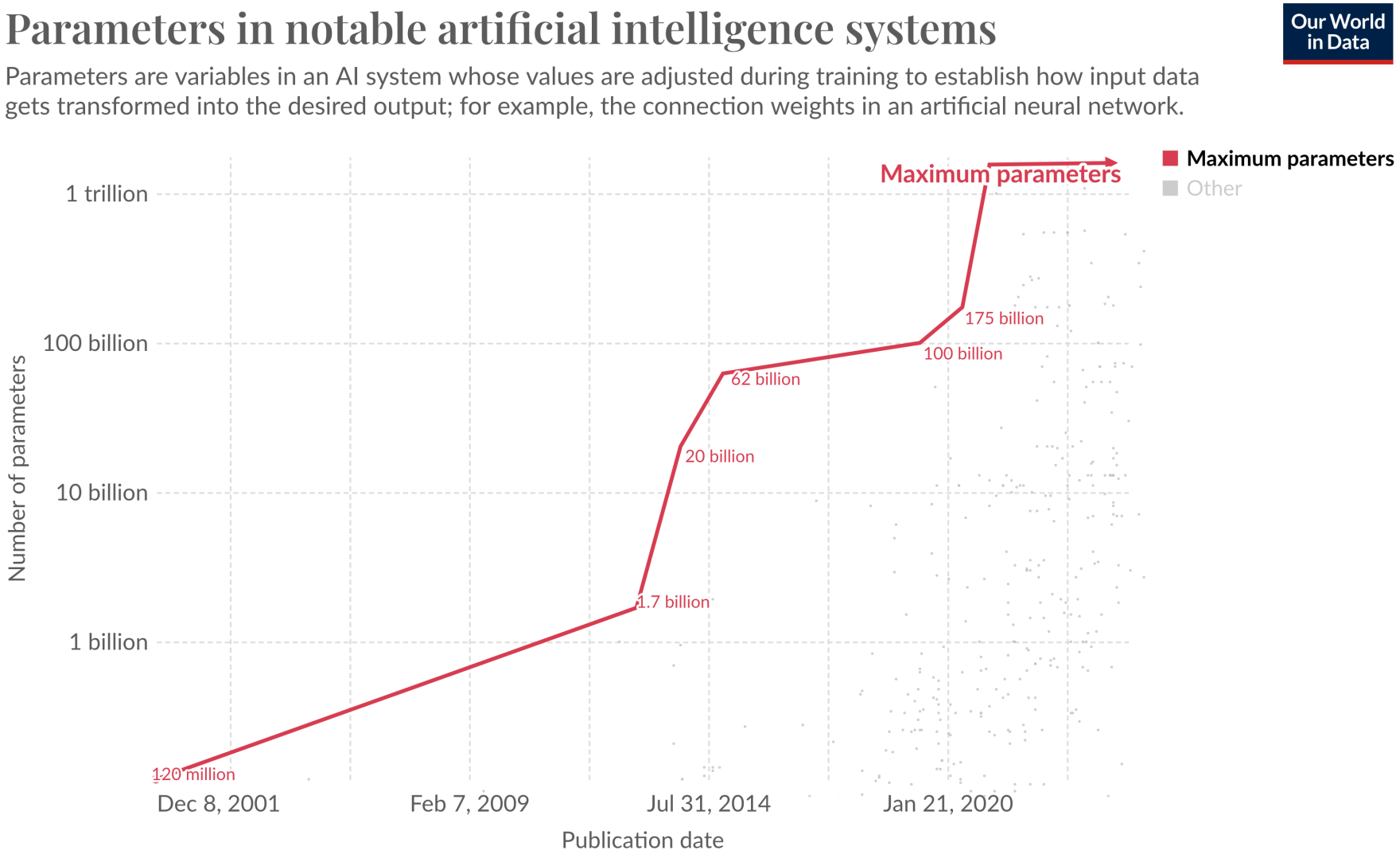



Traditional ML models




Foundation models

# The growth of LLMs




Source: [outworldindata]

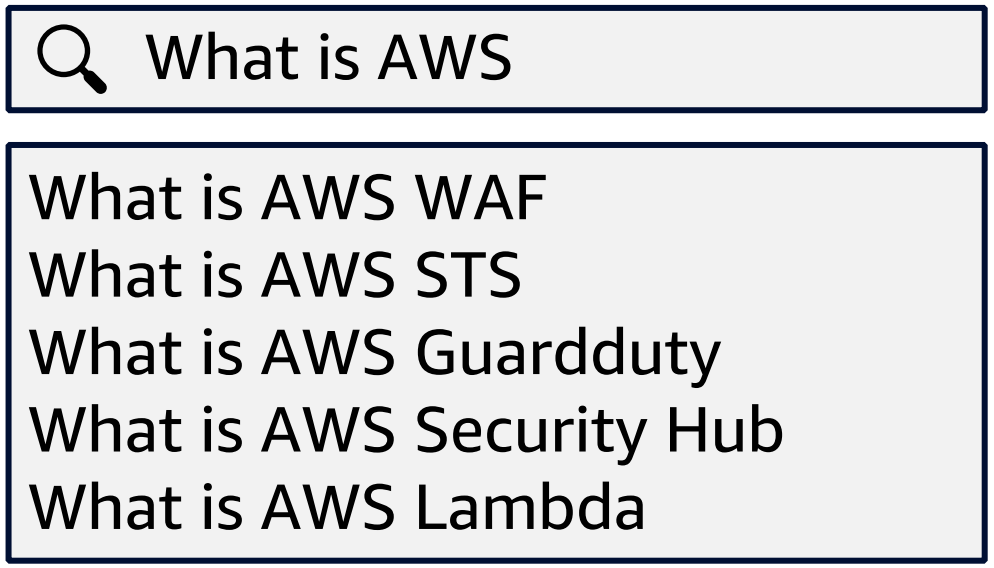
# Transformer Networks

# Sequence-to-sequence models

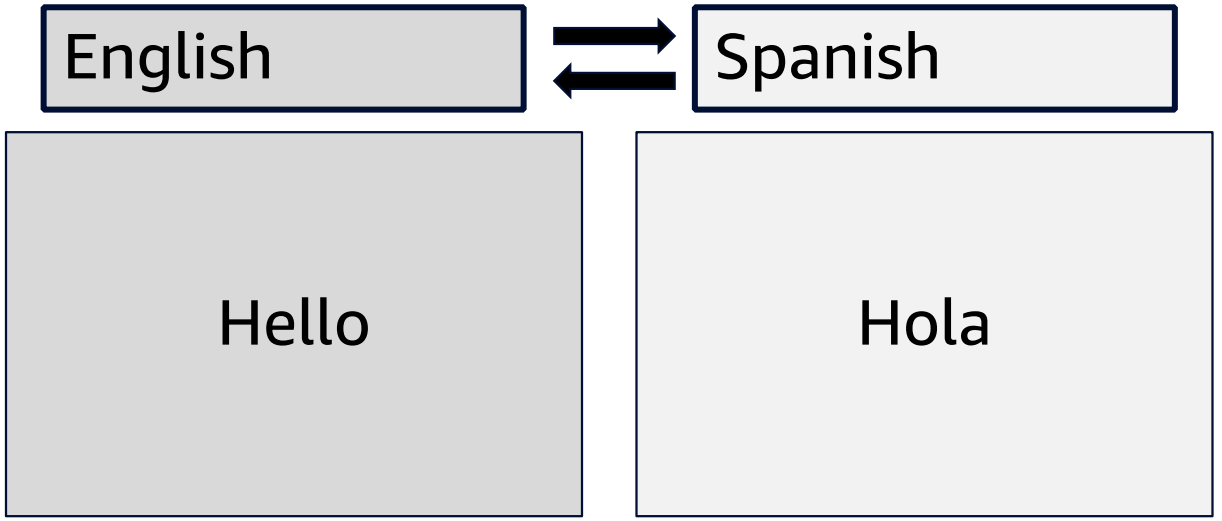
Use input sequences to produce output sequences.

- Input and output are **both variable-length** sequences.
- Examples:

Auto-completion

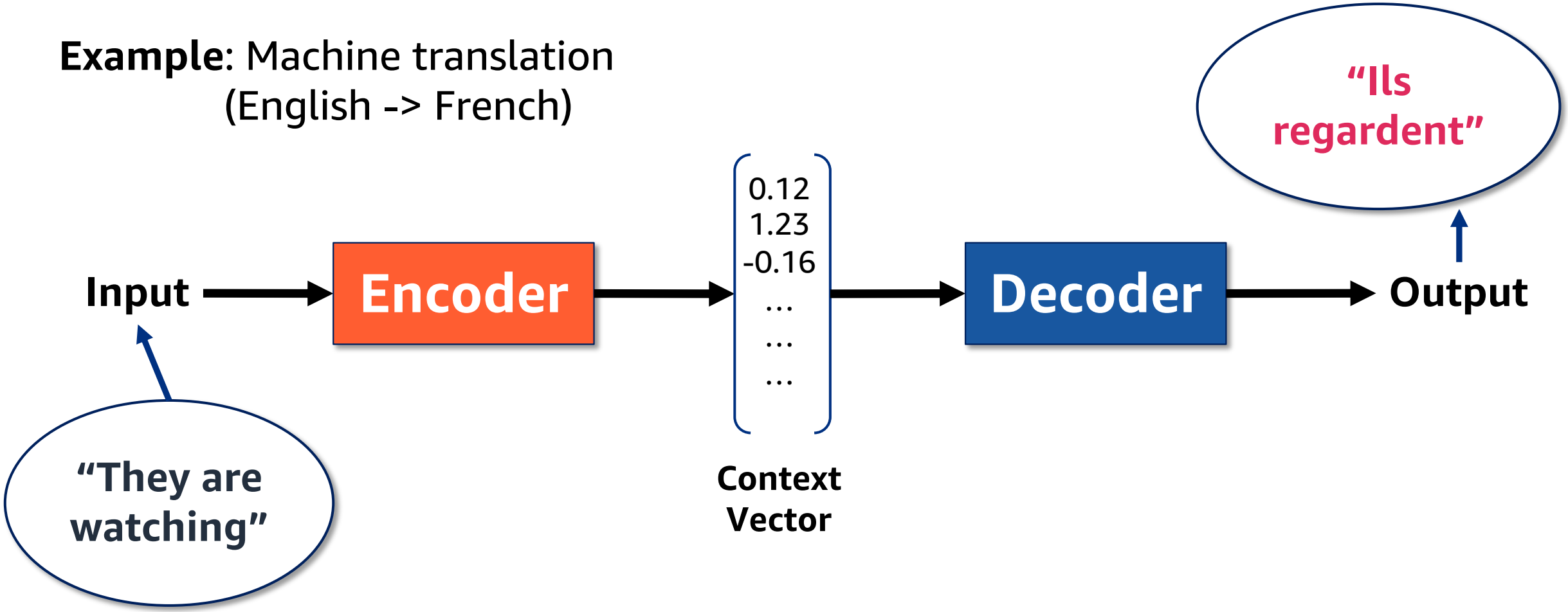



Machine translation



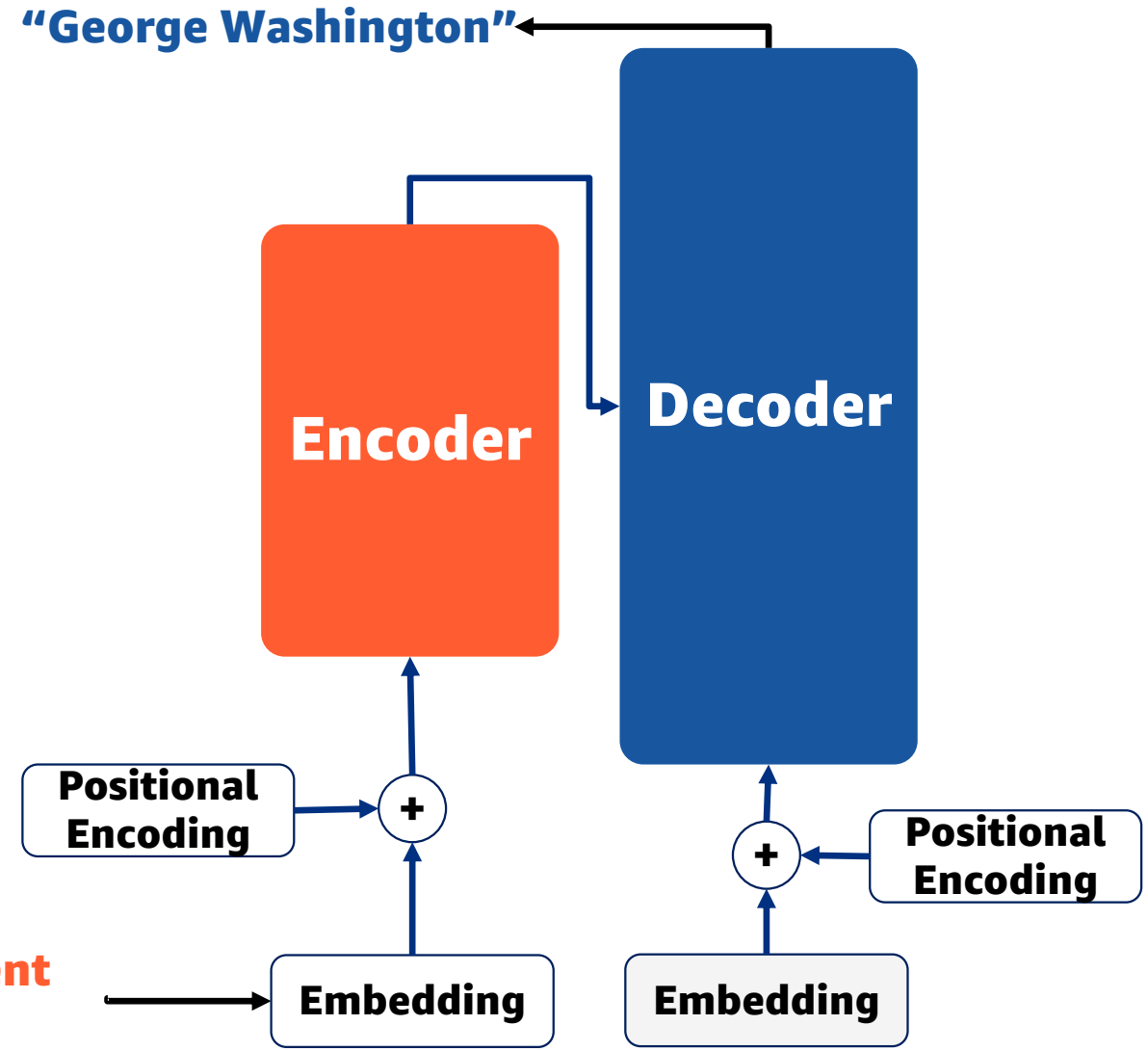
# Encoder-Decoder Architecture

**Example**: Machine translation
(English -> French)



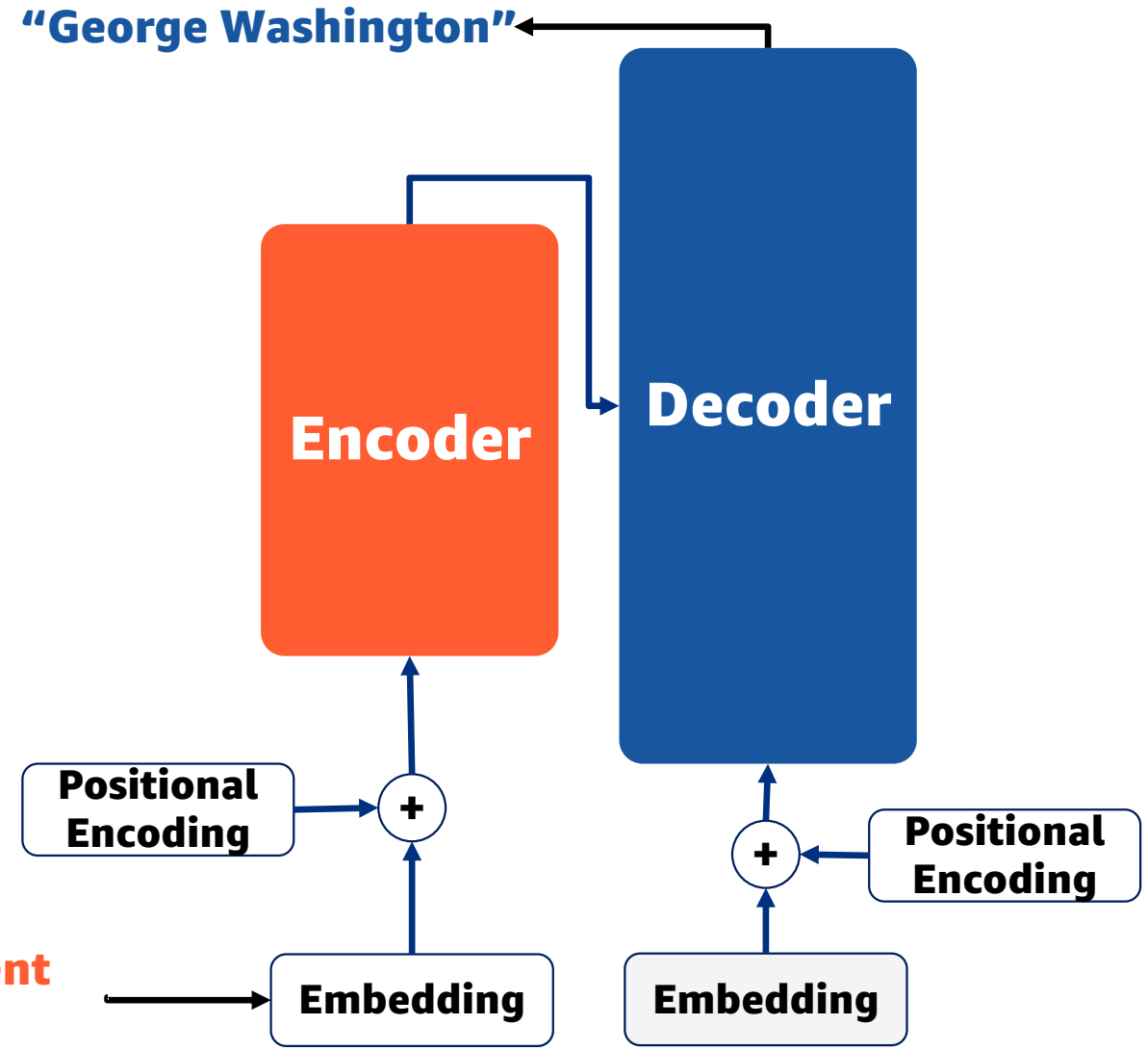
# Transformers

- Developed as an encoder-decoder model
- Uses the **attention mechanism**
  - Rich semantic and syntactic representation
  - Long context memory
- Parallel processing of inputs
- Distributed inference



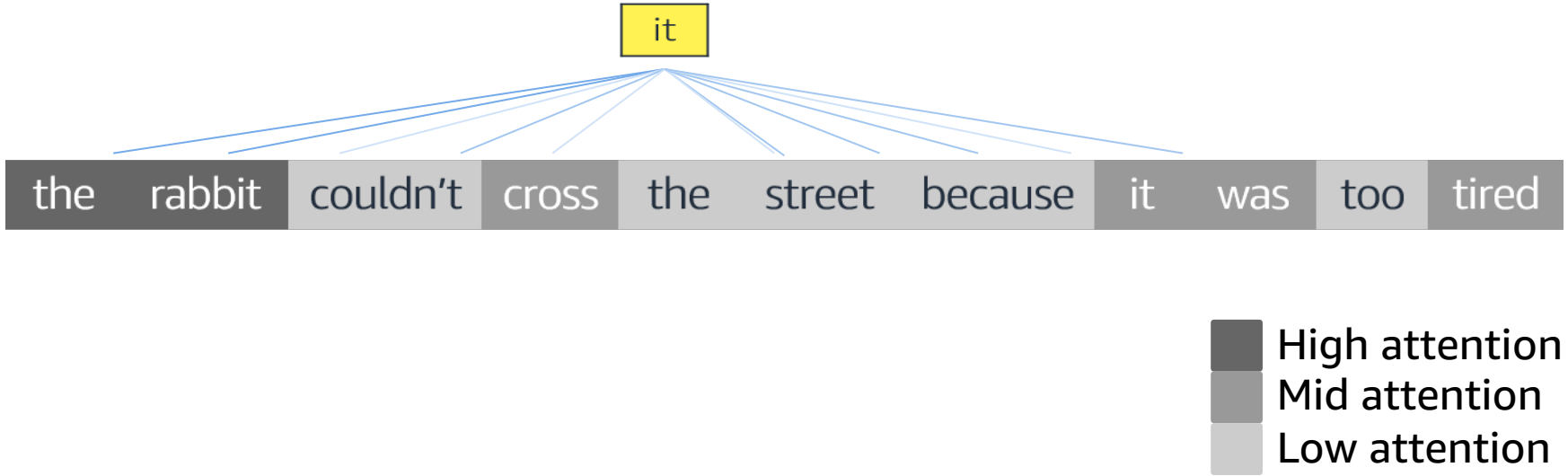
# Transformers

- **Encoder** generates rich representation of input
- **Decoder** generates relevant content
  - Output is again passed as input to the decoder
- Positional information added to input



# Attention Mechanism

- Attention helps models **associate each word** with **other words** in the sentence



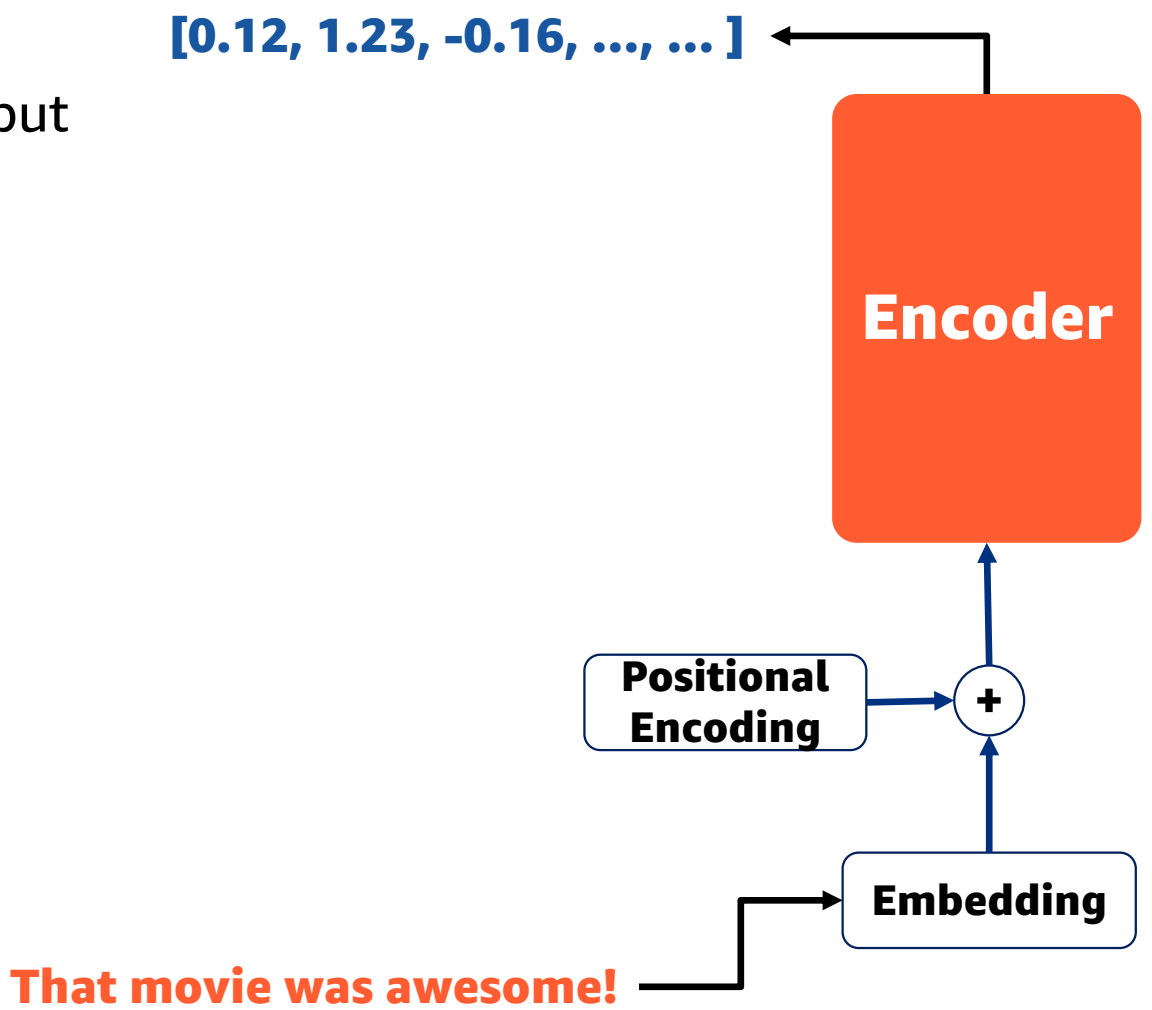
# Attention Mechanism

- Allows the model to access all the words/tokens in the observable input
- Indicates which tokens are most relevant for the next prediction

- Both the encoder and decoder use the attention mechanism
  - **Multi-headed attention module**

# Encoder-only Architecture

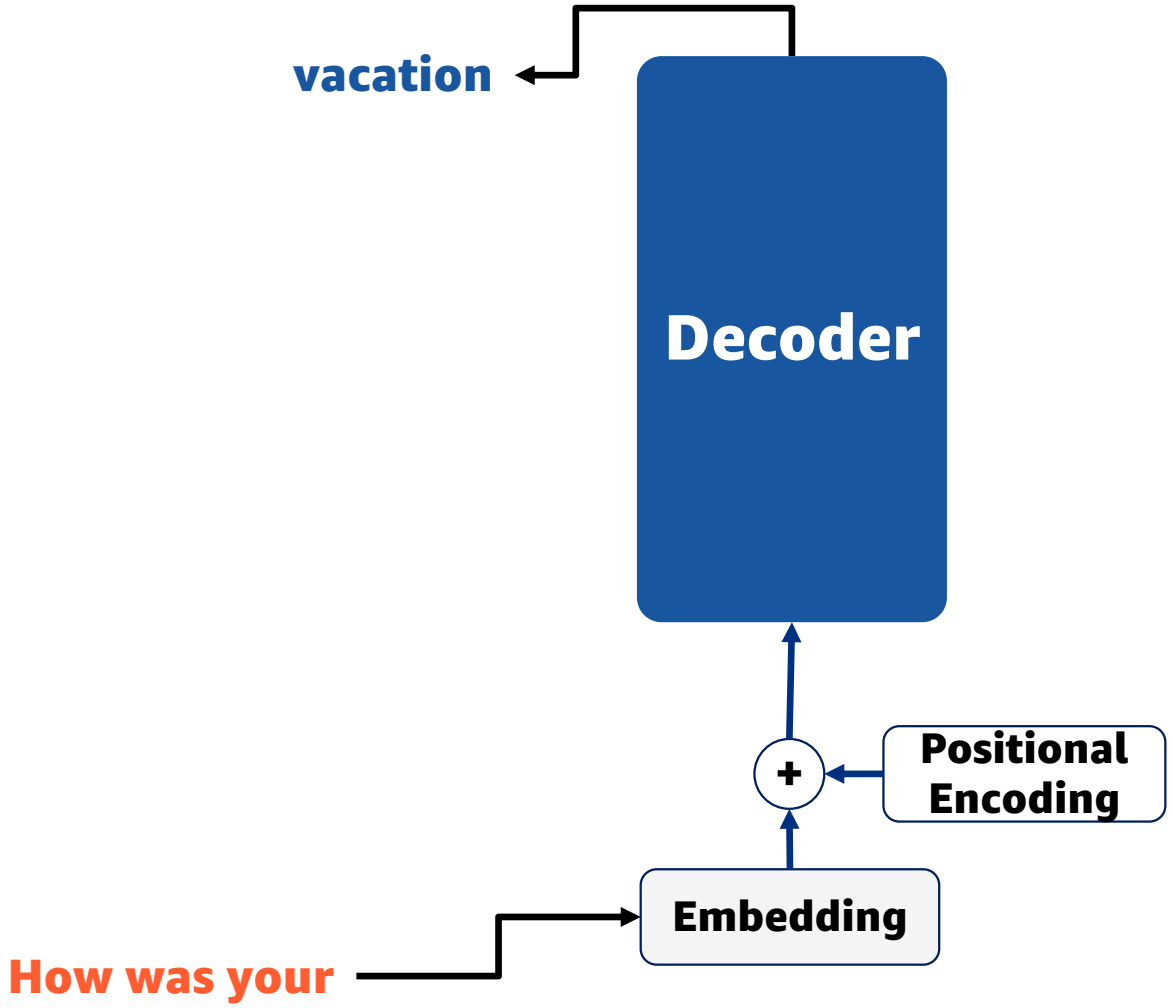
- Only uses the encoder block
- Attention layers access all the words in the input sentence.
  - **Bi-directional attention**
- Suitable for Natural Language Understanding (NLU) tasks
  - Sentence classification
  - Named entity recognition (NER)
  - Text extraction
- Examples:
  - BERT Models
  - ELECTRA



# Decoder-only Architecture

- Only uses the decoder block
- Only access words that came before
  - **Auto-regressive models**
- Generate next token based on input
- Suitable for text generation tasks

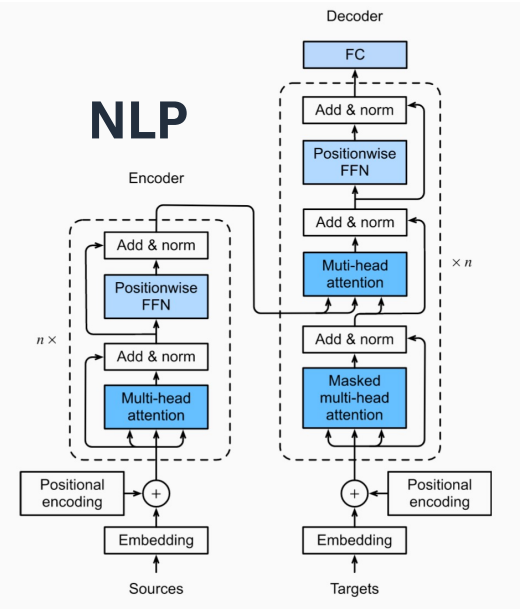
- Examples:
  - GPT Models by OpenAI
  - Llama models by Meta
  - Claude models by Anthropic

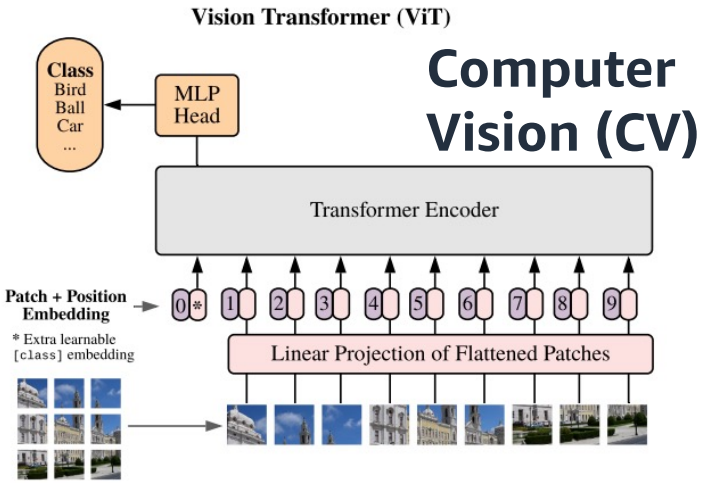


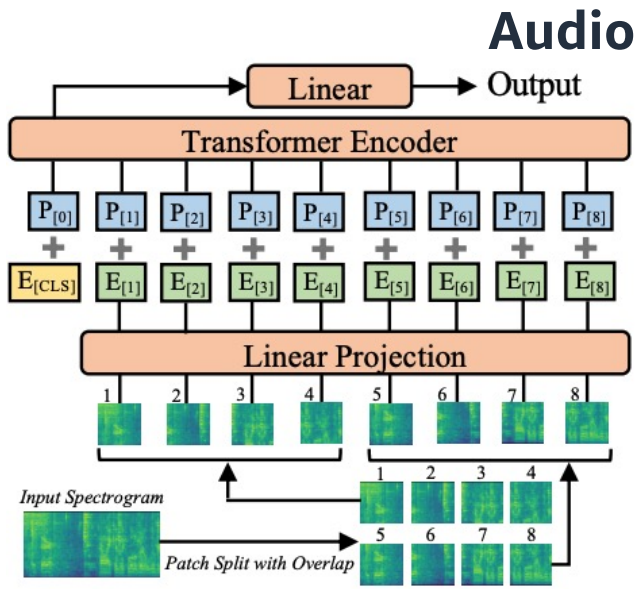
# Transformers: Summary

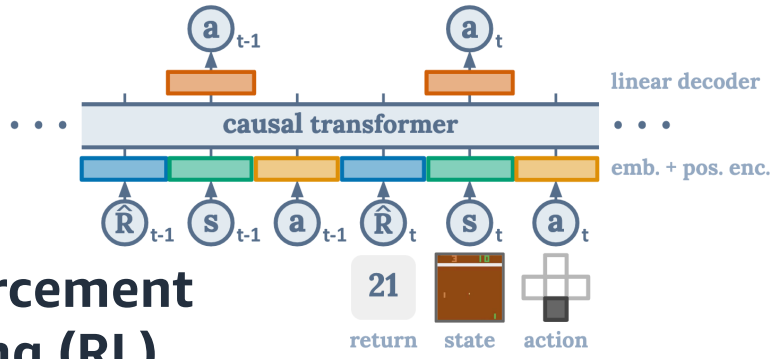
- **State-of-the art** deep learning architecture
- Propelled the growth and adoption of **Generative AI**
- Transformers can process the input data in **parallel**
- Self-attention mechanism capture interdependencies between all words, regardless of position
- Typically undergo **self-supervised learning**
  - Labels generated automatically from unlabeled data
  - No need to curate labelled data
- Increasingly popular with Computer Vision (CV), Audio, Reinforcement Learning (RL) tasks, and multi-modal applications

# Transformers for all!




Reinforcement Learning (RL)

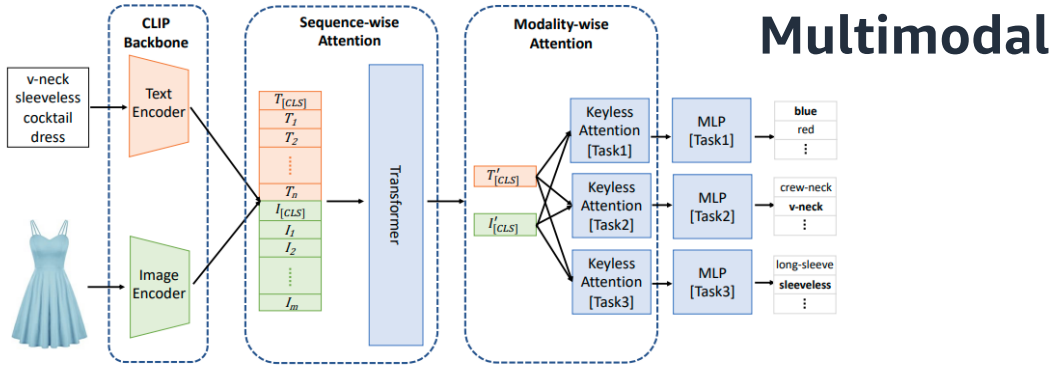



Figure 1: The pipeline of our proposed CMA-CLIP.

# Challenges and Limitations of LLMs

# Challenges and limitations of LLMs (1/3)

- **Reliability and bias**
  - Knowledge limited to the training data
  - Inability to discern false information or bias
- **Context window**
  - Model’s attention limited to the context window
  - Inputs exceeding the context window length are invisible to the model
  - For instance, Titan Premier had a limit of 30,000 tokens at the time of release

# Challenges and limitations of LLMs (2/3)

- **Potential copyright infringement issue**
  - Training data might contain sensitive or copywrite data
  - May generate content similar to someone’s intellectual or creative property
  - Have ethical and legal implications

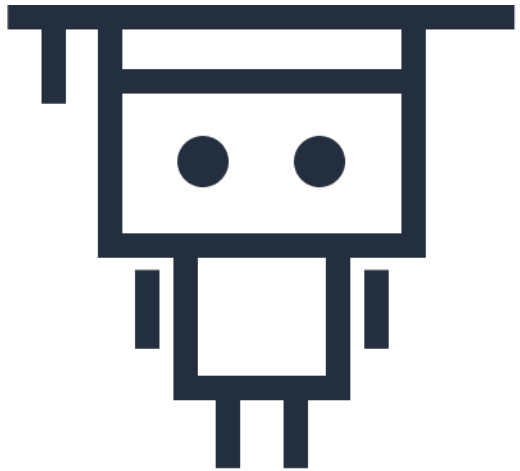
- **Create and propagate misinformation**
  - May generate personal or sensitive data that could be used to identify and harm others
  - Create and propagate misinformation about individuals,  groups, organizations, etc

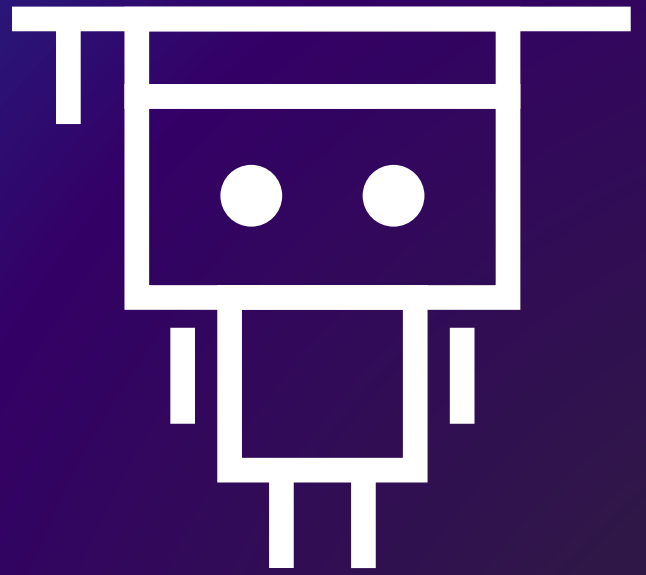
# Challenges and limitations of LLMs (3/3)

- **System costs**
  - Requires significant investment in the form of computer systems, human capital (engineers, researchers, scientists, etc.), and power.
  - Models with >100 billion parameters can have a total project cost of over $100 million
- **Environmental Impact**
  - LLMs need a lot of power and leave behind large carbon footprints
  - According to a study, $CO_2$ emissions from training 5B models on GPU is roughly equivalent to a trans-American flight.

# Next lesson

- This lesson covered foundation modes and LLMs.
- In the next lesson, you will learn about prompt engineering and a few techniques to improve the quality of the model’s response through prompting.





# Thank you!