

*September 6, 2023*

---

# AI Tools for Education and Research

Gordon Erlebacher  
Department of Scientific Computing  
Florida State University

---



With great  
power comes  
great  
responsibility

---

# Outline

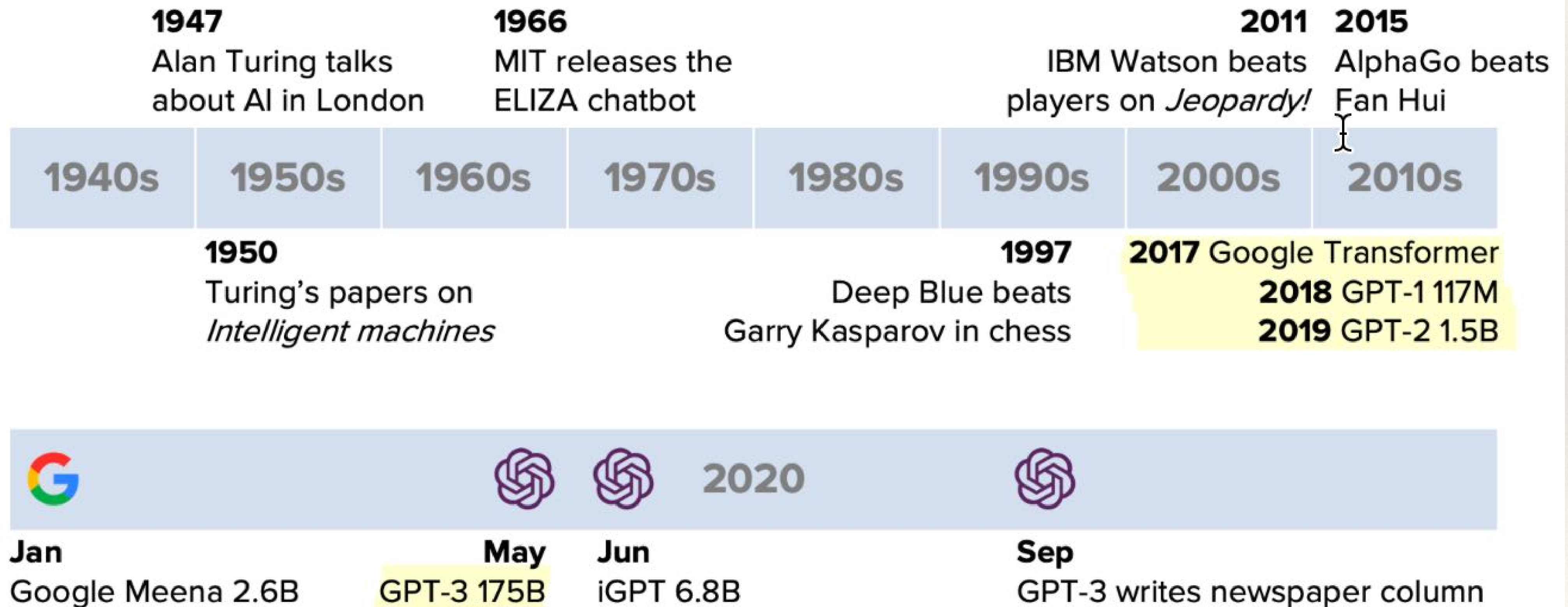
---

- ❖ Introduction
- ❖ Transformer Architecture
- ❖ Prompting
- ❖ AI model comparison
- ❖ AI for Education
- ❖ AI helps Coding
- ❖ Conclusions and Predictions

# Introduction

# Rise of AI

## AI TIMELINE: 1947-2020



1997

# IBM DeepBlue



Generated by Leonard.ai

2016

AlphaGo: 5  
Lee Sedol: 1

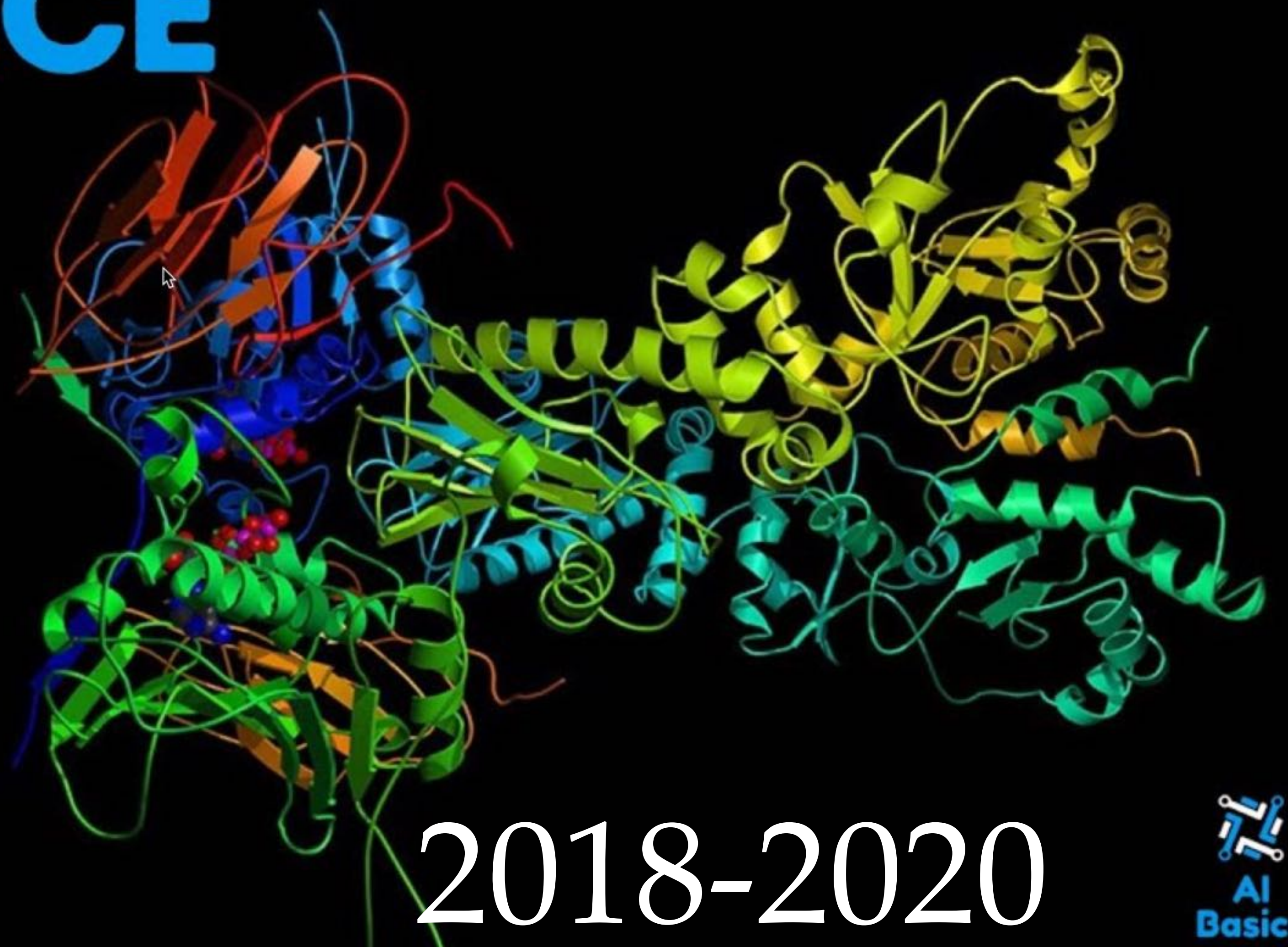


# ARTIFICIAL INTELLIGENCE



SOLVES 50  
YEAR OLD  
SCIENCE  
PROBLEM

(ALPHAFOLD)



2018-2020

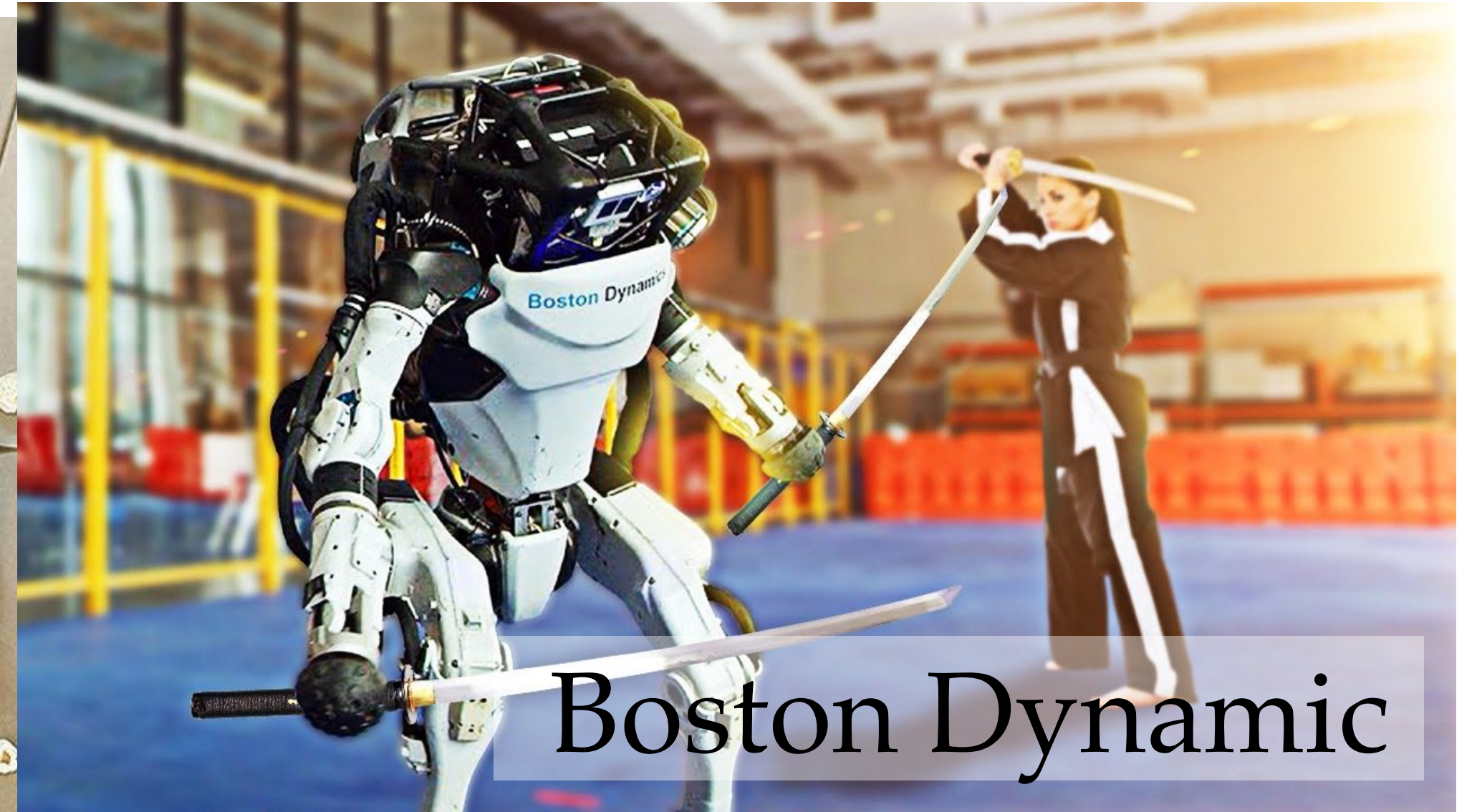




Google Home



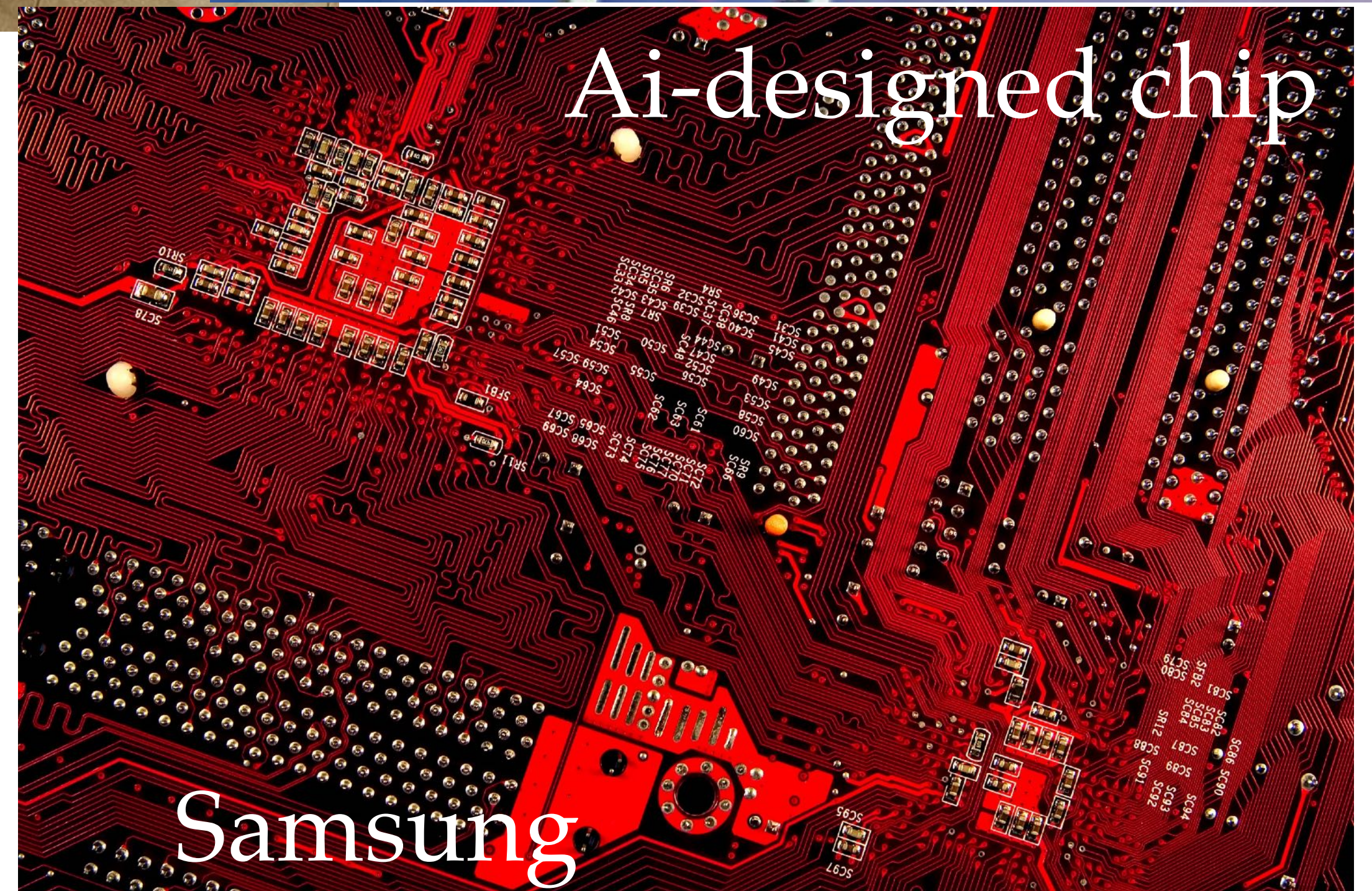
Roomba



Boston Dynamic



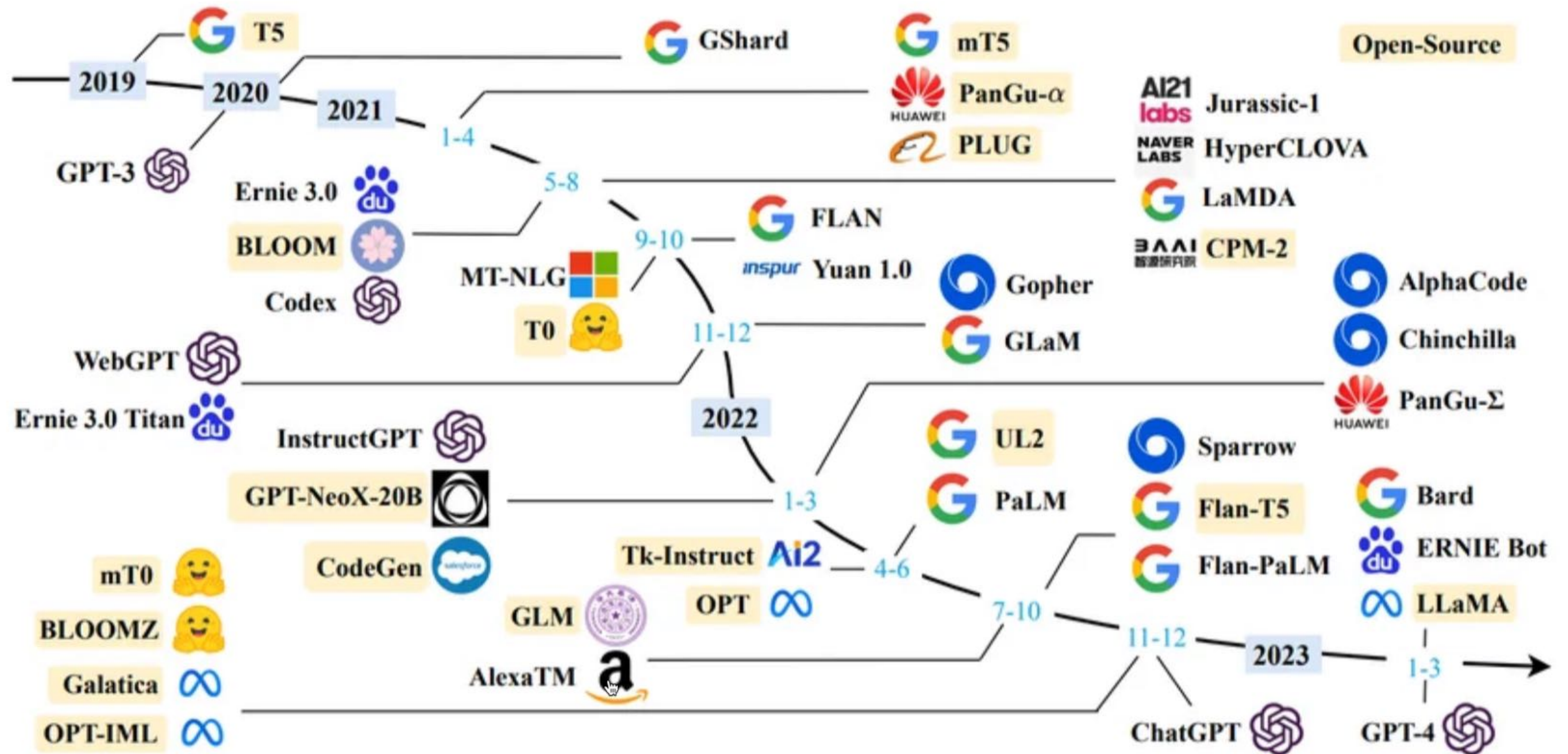
AI-enhanced cameras



Ai-designed chip

Samsung

# AI's Evolutionary Journey: 2019-mid2023



---

# What is a Large-Language Model?

---

a next word predictor

given all the previous words

in a context

---

# How to use LLMs

---

- ❖ LLMs are extremely powerful
- ❖ LLMs are fallible
- ❖ Do not implicitly trust a LLM
- ❖ Consider polling multiple LLMs
- ❖ Ask a question several times in separate conversations
- ❖ Be aware of the context length of each LLM (for now)

---

# Main parameters in a LLM

---

- ❖ **Context size:** the maximum number of input characters
- ❖ **Model size:** the number of unknown parameters in the neural network
- ❖ **Time to train:** how long the network is trained for
- ❖ If some function of the three is large enough, *emergence* can take place.

---

# The Data

---

- ❖ **Input data**  $(x_1, x_2, \dots, x_n)$  where  $x_i$  is one or more sentences where total number of words  $\leq N$ , where  $N$  is the **context length**
- ❖ **Example: GPT-4** has a context length of 16,000 tokens.
- ❖ The maximum length of  $x_i$  is 16,000 tokens (approx. 12,000 words).
- ❖ A 15 page NSF grant in 11 pt font will have  $O(7500)$  words.
- ❖ An NSF grant  $\leq x_i$  for GPT-3.5 (ChatGPT)

---

# Tokenization

---

- ❖ **Input sentence:** “What is the meaning of this sentence? The meaning is clear to me.”
- ❖ Each character is a token:
  - ❖ ['W', 'h', 'a', 't', ' ', 'i', 's', ' ', 'e', 'n', '...']
  - ❖ Hard to work with
- ❖ Each space-delimited word is a token:
  - ❖ ['What', 'is', 'the', 'meaning', 'of', 'this', 'sentence', '?', 'clear', 'to', 'me']
  - ❖ Very large vocabularies. Some languages do not have spaces.
- ❖ Sub-word tokenizer: identify most common `n-grams`: (depends on the vocabulary size)
  - ❖ ['\_\_What', '\_\_is', '\_\_the', '\_\_mean', 'ing', '\_\_of', '\_\_this', '\_\_s', 'ent', 'ence', '?', '\_\_The', '\_\_mean', 'ing', '\_\_is', '\_\_cle', 'ar', '\_\_to', '\_\_me.']
  - ❖ Works across all languages.
  - ❖ Most common algorithm in Large-language models: Sentencepiece
  - ❖ Combines the advantages of character-based and word-based tokenization

---

# Tokenization

---

- ❖ **Input sentence:** “What is the meaning of this sentence? The meaning is clear to me.”
- ❖ Each character is a token:
  - ❖ ['W', 'h', 'a', 't', ' ', 'i', 's', ' ', 'e', 'n', 'g', ' ', 'o', 'f', ' ', 't', 'h', 'i', 's', ' ', 's', 'e', 'n', 't', 'e', 'n', 'c', 'e', '?', ' ', 'T', 'h', 'e', ' ', 'm', 'e', 'a', 'n', 'i', 'n', 'g', ' ', 'i', 's', ' ', 'c', 'l', 'e', 'a', 'r', ' ', 't', 'o', ' ', 'm', 'e', '.']
  - ❖ Hard to work with
- ❖ Each space-delimited word is a token:
  - ❖ ['What', 'is', 'the', 'meaning', 'of', 'this', 'sentence', '?', 'clear', 'to', 'me']
  - ❖ Very large vocabularies. Some languages do not have spaces.
- ❖ Sub-word tokenizer: identify most common `n-grams`: (depends on the vocabulary size)
  - ❖ ['\_\_What', '\_\_is', '\_\_the', '\_\_mean', 'ing', '\_\_of', '\_\_this', '\_\_s', 'ent', 'ence', '?', '\_\_The', '\_\_mean', 'ing', '\_\_is', '\_\_cle', 'ar', '\_\_to', '\_\_me.']
  - ❖ Works across all languages.
  - ❖ Most common algorithm in Large-language models: Sentencepiece
  - ❖ Combines the advantages of character-based and word-based tokenization



---

# Tokenization

---

- ❖ **Input sentence:** “What is the meaning of this sentence? The meaning is clear to me.”
- ❖ Each character is a token:
  - ❖ ['W', 'h', 'a', 't', ' ', 'i', 's', ' ', 'e', 'n', '...', ...]
  - ❖ Hard to work with
- ❖ Each space-delimited word is a token:
  - ❖ ['What', 'is', 'the', 'meaning', 'of', 'this', 'sentence', '?', 'clear', 'to', 'me']
  - ❖ Very large vocabularies. Some languages do not have spaces.
- ❖ Sub-word tokenizer: identify most common `n-grams`: (depends on the vocabulary size)
  - ❖ ['\_\_What', '\_\_is', '\_\_the', '\_\_mean', 'ing', '\_\_of', '\_\_this', '\_\_s', 'ent', 'ence', '?', '\_\_The', '\_\_mean', 'ing', '\_\_is', '\_\_cle', 'ar', '\_\_to', '\_\_me.']
  - ❖ Works across all languages.
  - ❖ Most common algorithm in Large-language models: Sentencepiece
  - ❖ Combines the advantages of character-based and word-based tokenization

---

# Tokenization

---

- ❖ **Input sentence:** “What is the meaning of this sentence? The meaning is clear to me.”
- ❖ Each character is a token:
  - ❖ ['W', 'h', 'a', 't', ' ', 'i', 's', ' ', 't', 'h', 'e', ' ', 'm', 'e', 'a', 'n', 'i', 'n', 'g', ' ', 'o', 'f', ' ', 't', 'h', 'i', 's', ' ', 's', 'e', 'n', 't', 'e', 'n', 'c', 'e', '?', ' ', 't', 'h', 'e', ' ', 'm', 'e', 'a', 'n', 'i', 'n', 'g', ' ', 'i', 's', ' ', 'c', 'l', 'e', 'a', 'r', ' ', 't', 'o', ' ', 'm', 'e', '.']
  - ❖ Hard to work with
- ❖ Each space-delimited word is a token:
  - ❖ ['What', 'is', 'the', 'meaning', 'of', 'this', 'sentence', '?', 'clear', 'to', 'me']
  - ❖ Very large vocabularies. Some languages do not have spaces.
- ❖ Sub-word tokenizer: identify most common `n-grams`: (depends on the vocabulary size)
  - ❖ ['\_\_What', '\_\_is', '\_\_the', '\_\_mean', 'ing', '\_\_of', '\_\_this', '\_\_s', 'ent', 'ence', '?', '\_\_The', '\_\_mean', 'ing', '\_\_is', '\_\_cle', 'ar', '\_\_to', '\_\_me.']
  - ❖ Works across all languages.
  - ❖ Most common algorithm in Large-language models: Sentencepiece
  - ❖ Combines the advantages of character-based and word-based tokenization

---

# Context of 4000 Tokens

---

- ❖ “Please tell me about the history of mathematics”
  - ❖ The max size is 4000 words
  - ❖ The used size is 8 words
- ❖ In chatGPT, all questions and answers up to the current questions *must* fit within the context
- ❖ Therefore, a larger context is often better!

---

# Context Length

---

- ❖ GPT3: 2k
- ❖ ChatGPT: Nov. 2021: 4k, Aug. 2023: 32k
- ❖ GPT-4: 32k
- ❖ Claude-2: 100k
- ❖ Llama-Code: 100k
  
- ❖ For reference: The Great Gatsby (book) has 72k tokens (210 pages)

# Transformer Architecture

---

# Unpacking the Transformer

---

- ❖ Input is a context:
  - ❖ one or more words, sentences, paragraphs
- ❖ Output is another context
  
- ❖ Examples:
  - ❖ English paragraph  $\rightarrow$  French paragraph
  - ❖ Orthography  $\rightarrow$  Phonology
  - ❖ Images  $\rightarrow$  Captions
  - ❖ Text  $\rightarrow$  Images, Video
  - ❖ More generally: `input_sequence`  $\rightarrow$  `output_sequence`

---

# Transformer Elements

---

- ❖ **Input Sequence**

  - Text tokens (or other data types) are fed into the model as a sequence

- ❖ **Encoder Layers**

  - Process the input sequence and produce an intermediate representation input into the decoder

- ❖ **Decoder Layers**

  - Generate the output sequence embedding based on the processed input

- ❖ **Attention**

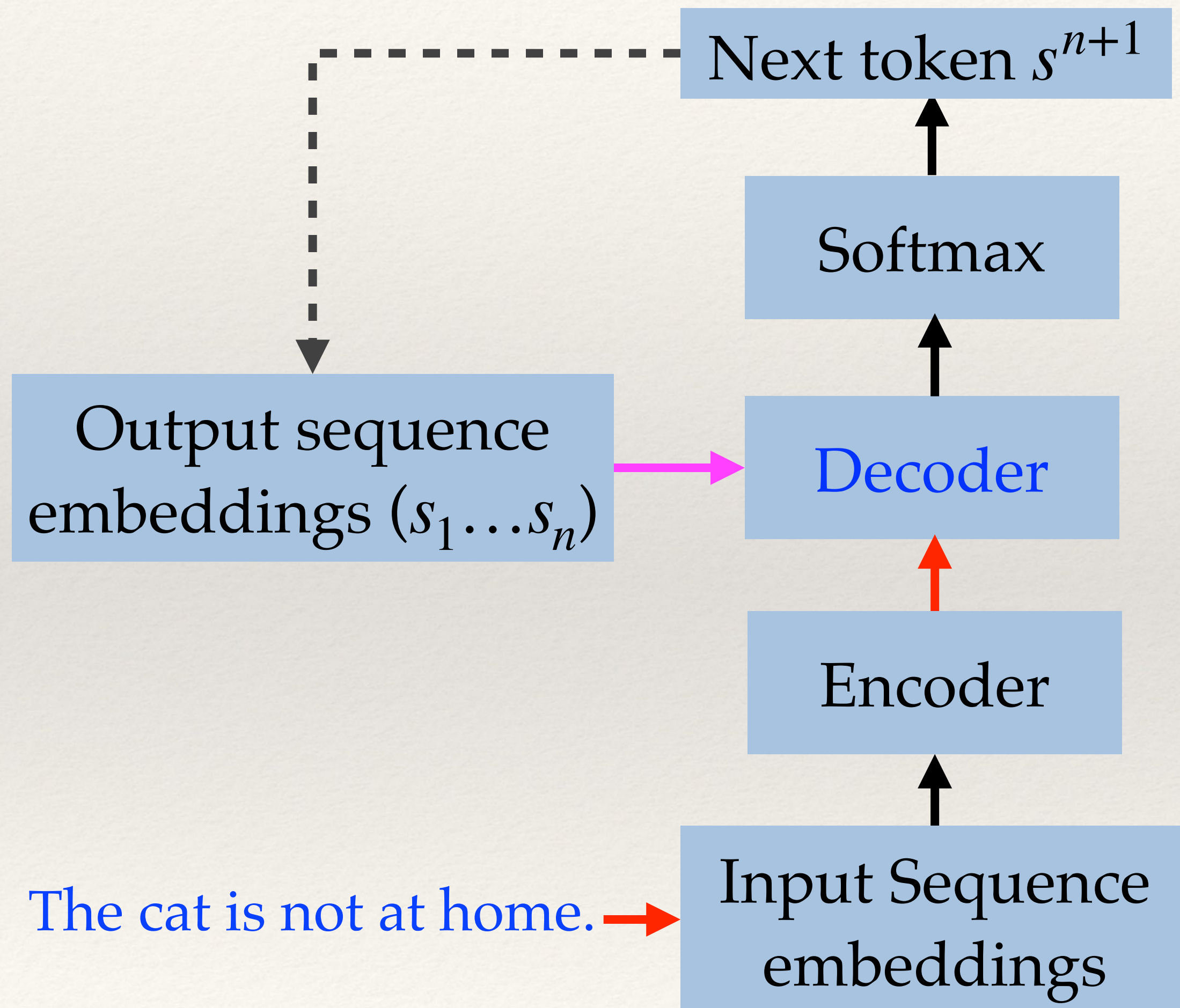
  - Captures relationships and dependencies between tokens regardless of separation

- ❖ **Output Sequence**

  - Transformed sequence of tokens

# High-Level Transformer Architecture

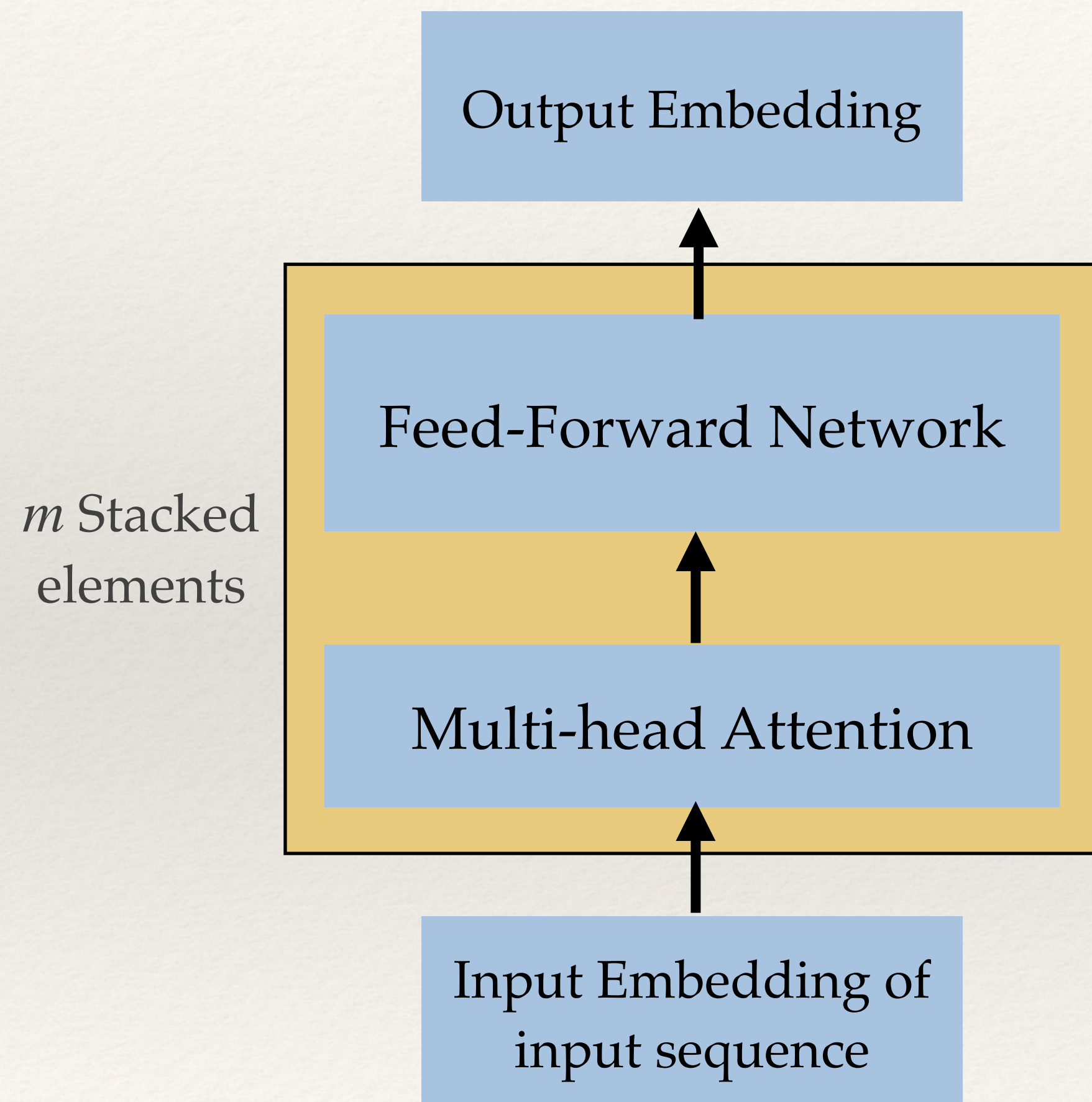
La chatte n'est pas a la maison.



- ❖ Input sequence  $\rightarrow$  Encoder
- ❖ The cat is not at home.
- ❖ **Encoder output** + **output sequence**  $\rightarrow$  **Decoder**
- ❖ La chatte n'est pas a la maison.
- ❖ Decoder output into Softmax for token prediction



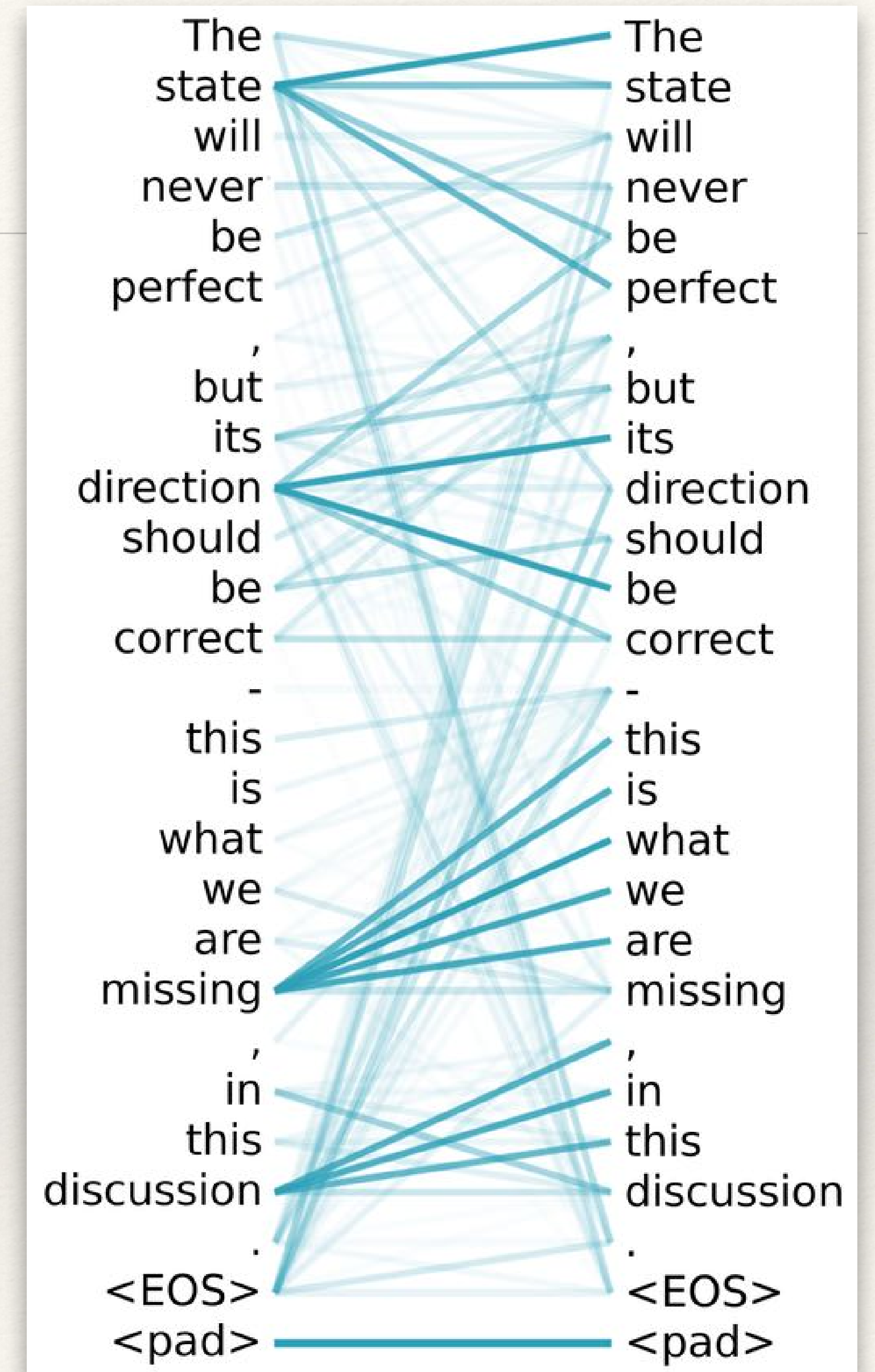
# Simplified Transformer Encoder



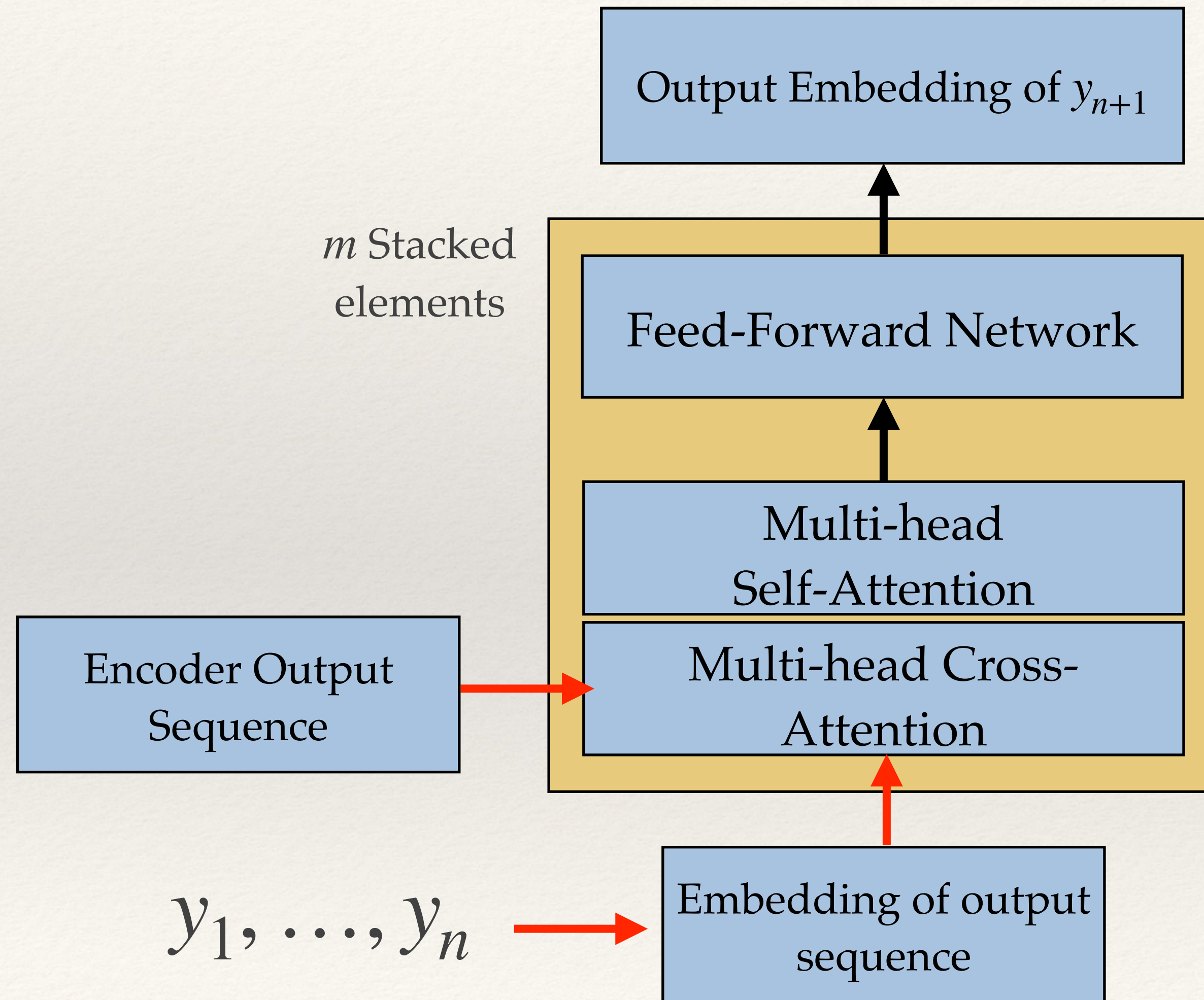
- ❖  $s = x_n \rightarrow$  Input embedding token
- ❖ embedding: vector  $\in \mathbb{R}^{N \times d}$
- ❖ Context:  $N$  tokens
- ❖ Multi-head attention
  - ❖ see next slide
- ❖ Output: intermediate representation
- ❖  $N$  tokens, each in embedding space  $\mathbb{R}^d$

# Self-Attention

- ❖ Each word (token) relates to every other word
- ❖ These links have different strengths
- ❖ Cost grows quadratically with context length
- ❖ **Active research:**
  - ❖ Speed up attention
  - ❖ Reduce memory requirements
  - ❖ Increase context length

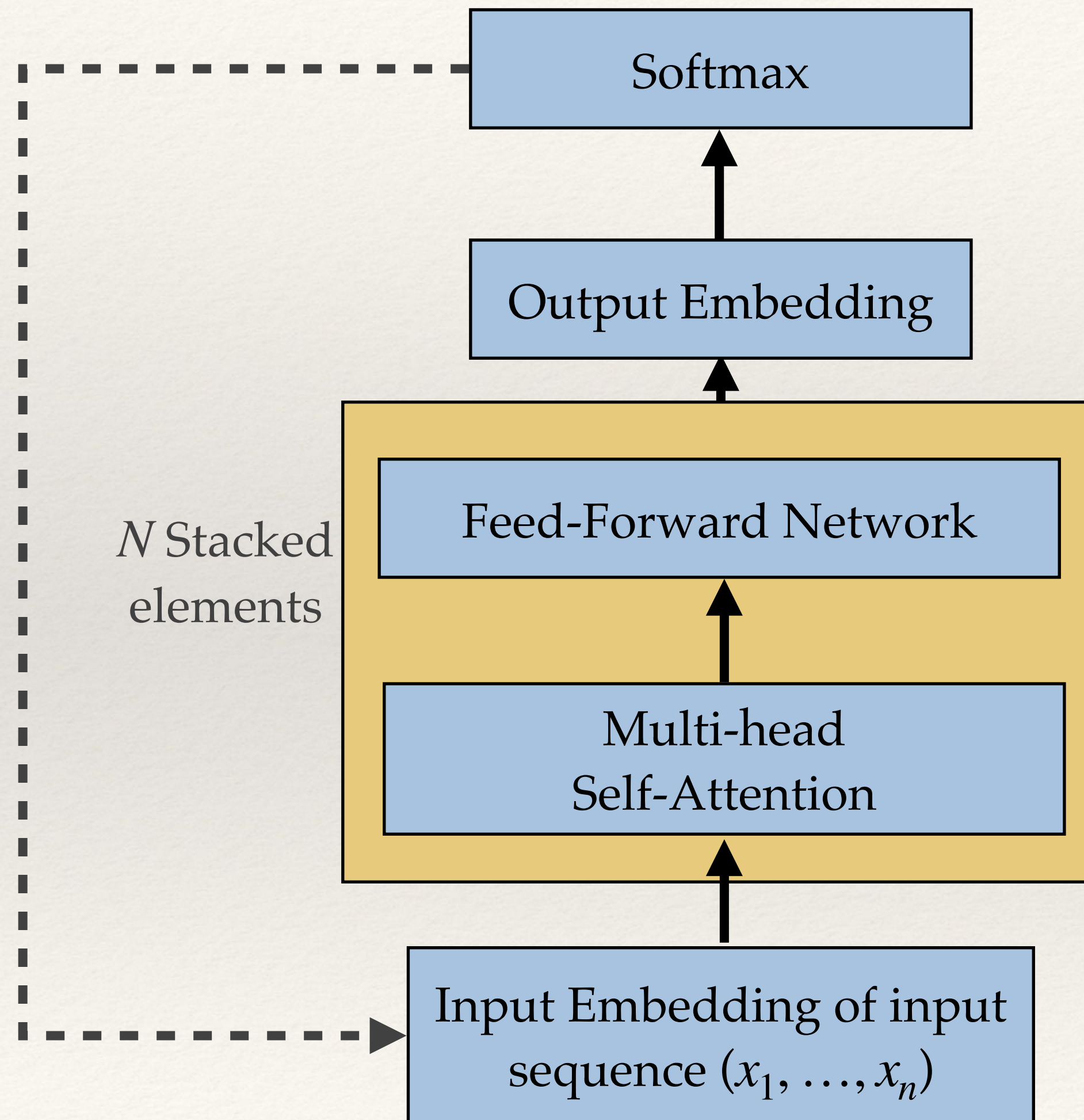


# Simplified Transformer Decoder



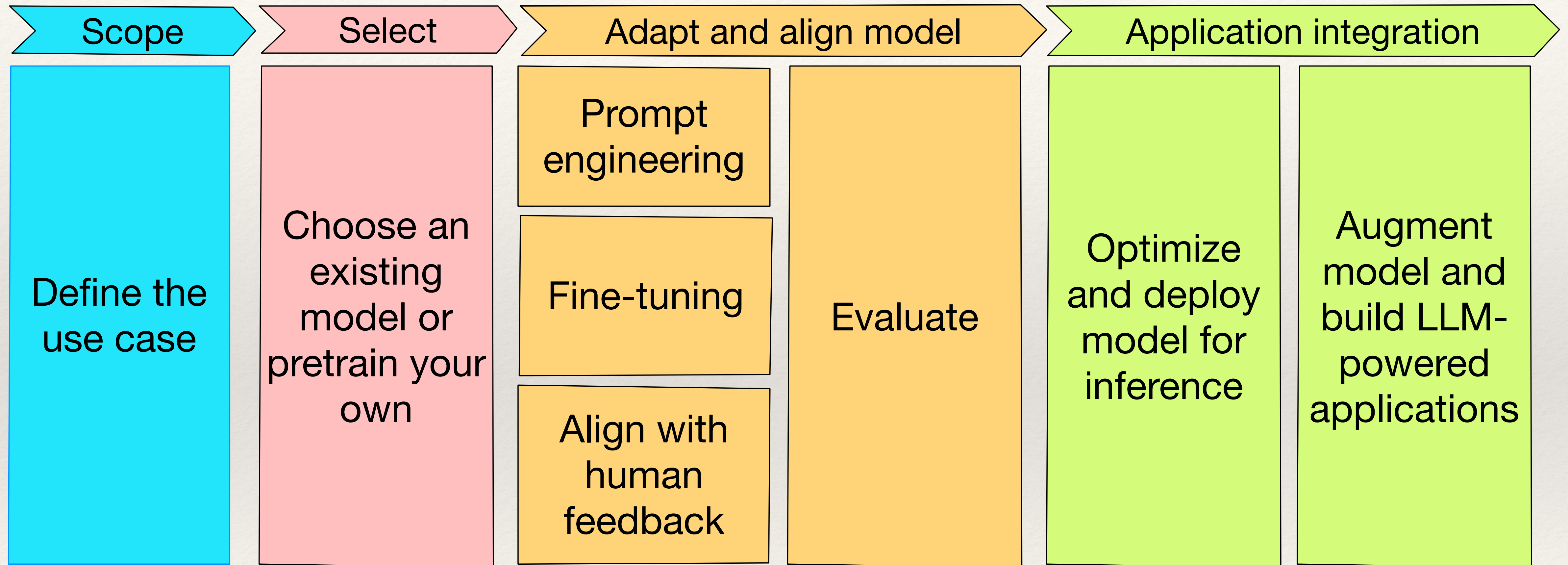
- ❖ Input embedding
- ❖ masked tokens  $\rightarrow \mathbb{R}^{N \times d}$
- ❖ Context: max of  $N$  tokens
- ❖ Multi-head cross attention
  - ❖ Similar to self-attention with input from the encoder
- ❖ Feed-forward network for mixing
- ❖ Output:  $N$  tokens, each in  $\mathbb{R}^d$

# Large-Language Model: Decoder-Only



- ❖ Used when input and output sequences are of the same nature (i.e., english text)
- ❖ Given tokens  $(x_1, \dots, x_n)$ , predict token  $x_{n+1}$ 
  - ❖ *Prompt:*  
“Paris is the capital of?”
  - ❖ *Most likely reply of a good model with next word prediction:*  
“France.”
  - ❖ *ChatGPT reply:*  
“Paris is the capital of France.”

# Life Cycle of a LLM



Source: <https://medium.com/@kanikaadik07/generative-ai-project-life-cycle-55ce9092e24a>

# Prompting

---

# Generating good prompts

---

- ❖ Consider whether you'd expect the answer to your question to exist on the internet
  - ❖ Asking about the composition of remote stars
    - ❖ probably poor response
  - ❖ Asking about life on Ferenginar
    - ❖ probably good response



---

# Prompting Techniques

---

- ❖ Single shot learning
- ❖ Multi-shot learning
- ❖ Step-by-Step prompting
- ❖ Chain-of-thought prompting
- ❖ Assign a Role



---

# Zero-Shot Prompting

---

## User Prompt:

- ❖ Which is a faster way to get to work?
- ❖ Option 1: Take a 1000-minute bus, then a half-hour train, and finally a 10-minute bike ride.
- ❖ Option 2: Take an 800-minute bus, then an hour train, and finally a 30-minute bike ride.

## AI reply

Option 1 is a faster way to get to work.

# Multi-Chain Prompting

## User Prompt:

Which is a faster way to get home?

Option 1: Take an 10 minutes bus, then an 40 minute bus, and finally a 10 minute train.

Option 2: Take a 90 minutes train, then a 45 minute bike ride, and finally a 10 minute bus.

Option 1 will take  $10+40+10 = 60$  minutes.

Option 2 will take  $90+45+10=145$  minutes.

Since Option 1 takes 60 minutes and Option 2 takes 145 minutes, Option 1 is faster.

Which is a faster way to get to work?

Option 1: Take a 1000 minute bus, then a half hour train, and finally a 10 minute bike ride. Option 2: Take an 800 minute bus, then an hour train, and finally a 30 minute bike ride.

## AI reply:

Option 1 will take  $1000+30+10 = 1040$  minutes.

Option 2 will take  $800+60+30 = 890$  minutes.

Since Option 2 takes 890 minutes and Option 1 takes 1040 minutes, Option 2 is faster.

---

# Single-Shot Learning

---

## Single user prompt with a single example

- ❖ **Q:** What is a dog?
- ❖ **A:** A dog is a domesticated mammalian carnivore that is commonly kept as a pet or for hunting. Dogs typically have a long snout, an acute sense of smell, non-retractile claws, and bark to communicate. Common breeds include Labrador, German Shepherd, and Poodle.

**Q:** What is a horse?

---

# Multi-Shot Learning

---

## Single User Prompt with multiple examples

- ❖ **Q:** What is a dog?
- ❖ **A:** A dog is a domesticated mammalian carnivore that is commonly kept as a pet or for hunting. Dogs typically have a long snout, an acute sense of smell, non-retractile claws, and bark to communicate. Common breeds include Labrador, German Shepherd, and Poodle.
- ❖ **Q:** What is a cat?
- ❖ **A:** A cat is a domesticated carnivorous mammal that is valued for its companionship and ability to hunt pests. Cats have retractile claws, whiskers, excellent night vision and hearing, and a talent for climbing. Common breeds include Siamese, Persian, and Maine Coon.
- ❖ **Q:** What is a horse?

---

# Application of Transformers: Beyond Text

---

- ❖ Generative Image Generation
  - ❖ Dall-E 2, Midjourney, Leonardo, Stable Diffusion, etc.
  - ❖ Combines GAN, Transformers, diffusion
- ❖ Caption generators
- ❖ Music Generators
- ❖ Video generation




## **Near future:**

- Video to text
- Music to text
- Video to music  
etc.

---

# Overleaf and Word

---

- ❖ Collaborative tools for Latex report writing
- ❖ AI tools integrated with Overleaf
  - ❖ Quillbot 
  - ❖ WriteFull 
  - ❖ Latext AI (Chrome plugin) 
- ❖ More tools for Word

# Comparison of AI Models

---

# LLMs in my Daily Work

---

- ❖ ChatGPT / GPT4 [🔗](#)
- ❖ Claude-2 [🔗](#)
- ❖ Perplexity [🔗](#)



---

# ChatGPT / GPT4 by OpenAI

---

- ❖ My go-to tool
- ❖ Vast knowledge
- ❖ Limitation: relatively small context length
- ❖ GPT4 (\$\$) includes:
  - ❖ Plugins
  - ❖ Code Interpreter
  - ❖ Web browser (currently disabled)

---

# Claude-2 by Anthropic

---

- ❖ 100k token context length
- ❖ Allows file uploading
  - ❖ Query the uploaded files
  - ❖ “Converse” about your papers
  - ❖ Very polite, strives to please

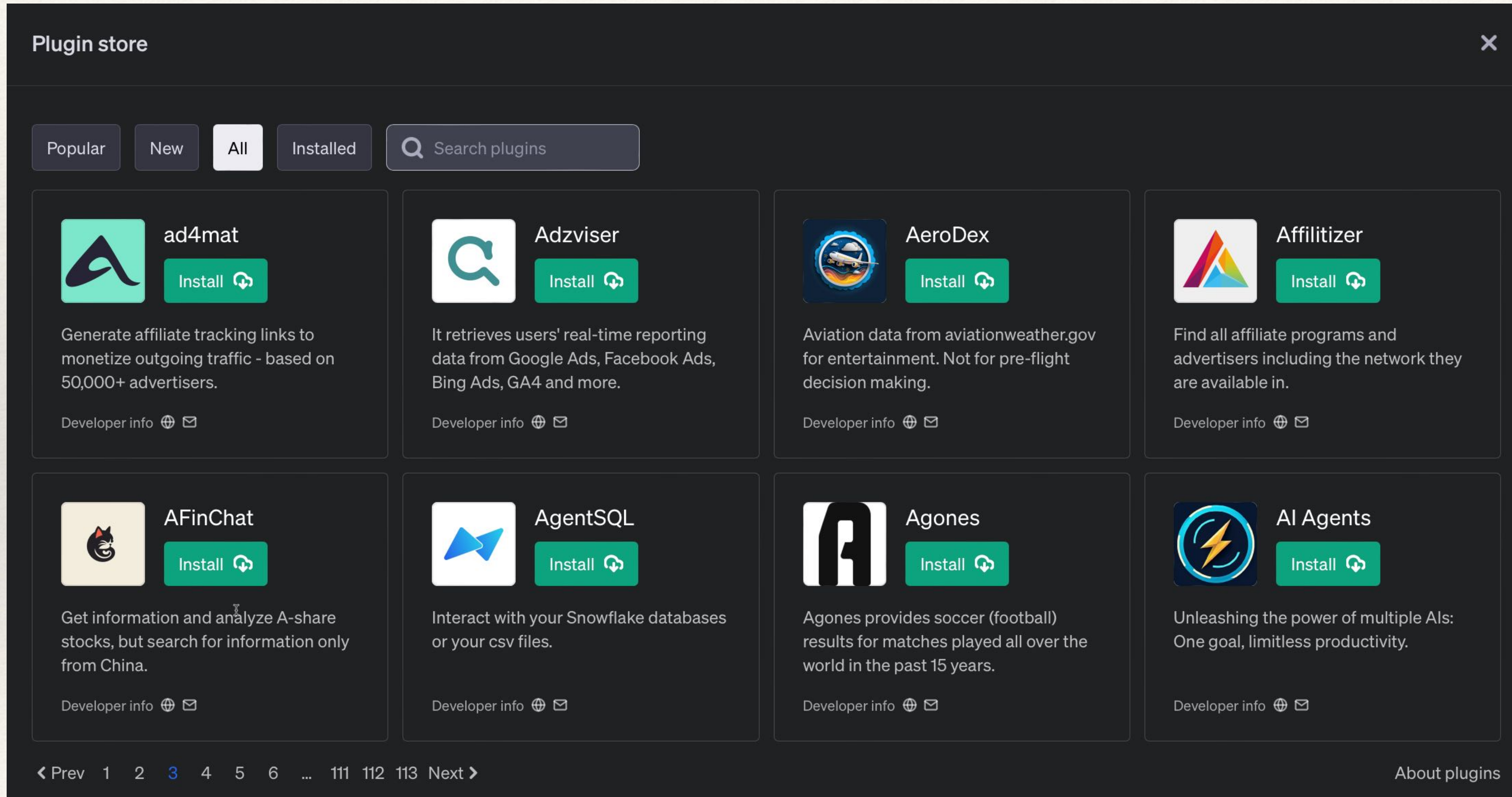
---

# Perplexity.ai by Perplexity AI

---

- ❖ Powered by GPT3.5
- ❖ Conversational search engine
- ❖ Ask questions in natural language
- ❖ Searches the internet
- ❖ Collates the results and returns a reply with citations in natural language
- ❖ Ask follow-up questions
- ❖ Its answers are more accurate than other Chat engines

# GPT-4 plugins






The screenshot displays the 'Plugin store' interface with a search bar and filters (Popular, New, All, Installed). It features eight plugin cards, each with an icon, name, 'Install' button, and a brief description:

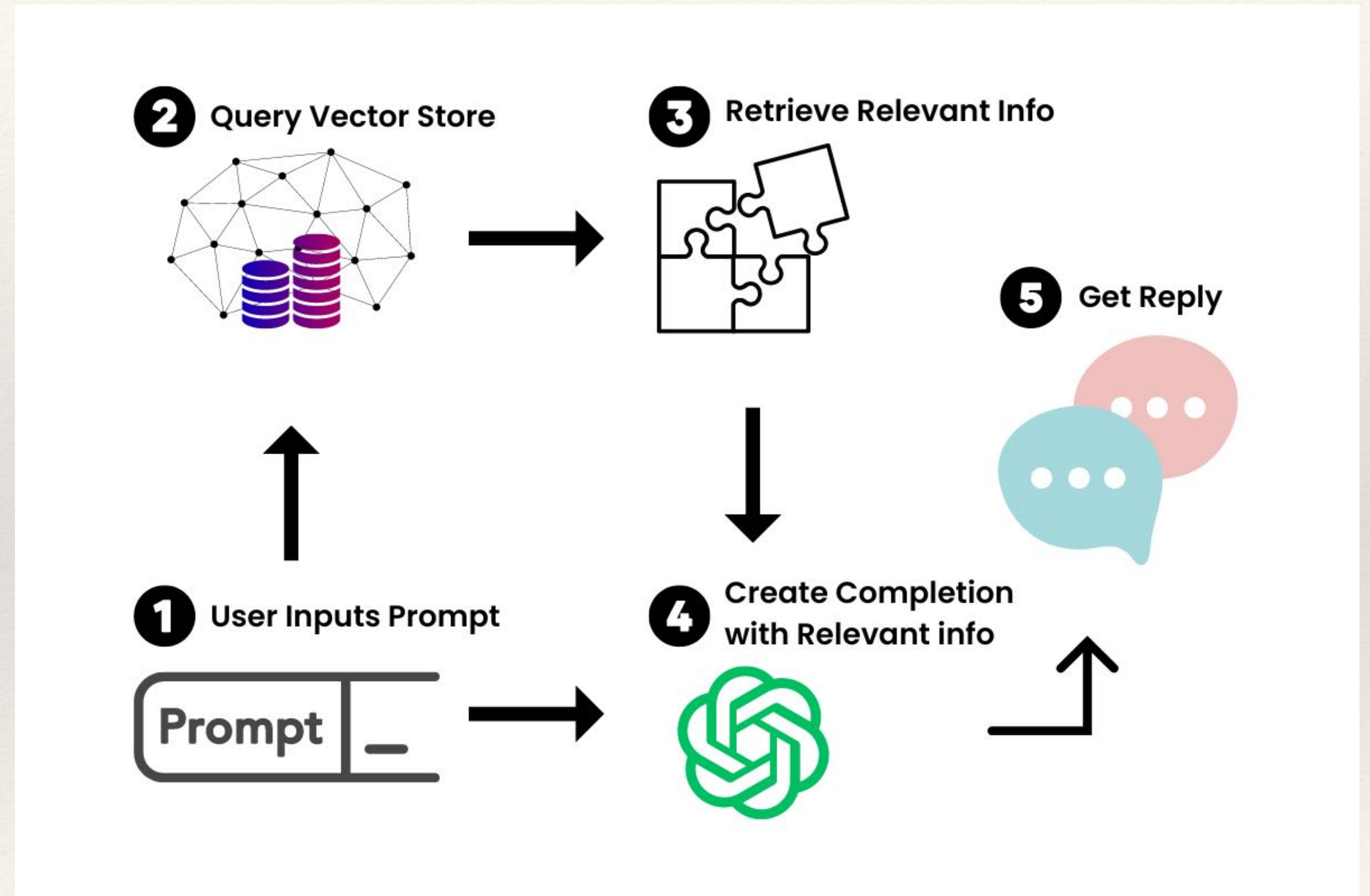
- ad4mat**: Generate affiliate tracking links to monetize outgoing traffic - based on 50,000+ advertisers.
- Adzviser**: It retrieves users' real-time reporting data from Google Ads, Facebook Ads, Bing Ads, GA4 and more.
- AeroDex**: Aviation data from aviationweather.gov for entertainment. Not for pre-flight decision making.
- Affilitizer**: Find all affiliate programs and advertisers including the network they are available in.
- AFinChat**: Get information and analyze A-share stocks, but search for information only from China.
- AgentSQL**: Interact with your Snowflake databases or your csv files.
- Agones**: Agones provides soccer (football) results for matches played all over the world in the past 15 years.
- AI Agents**: Unleashing the power of multiple AIs: One goal, limitless productivity.

At the bottom, there is a pagination control showing '1 2 3 4 5 6 ... 111 112 113 Next' and a link for 'About plugins'.

- ❖ Close to 1000 plugins
- ❖ Wide range of capability
- ❖ Will get better in time
- ❖ Plugins connect GPT4 to external applications
- ❖ Open up to three plugins
- ❖ GPT4 chooses which plugin(s) to use

# Chat Agents

- ❖ [Langchain](#) 
- ❖ [Llama-index](#) 
- ❖ [SimpleAIChat](#) 
- ❖ Create your own chatbot
- ❖ Extend the context length with a Vector Database
- ❖ Create your own Perplexity



---

# Ethical Considerations

---

- ❖ Avoid misuse of LLM
  - ❖ disinformation
  - ❖ fake videos
  - ❖ using LLM agents to create chaos, sow fear, etc.
- ❖ LLM's might have captured personal information from the public web
  - ❖ avoid using this information
- ❖ Dangers in providing advice to people based on LLM output
  - ❖ e.g. law, medicine, financial advice

# AI's Role in Education

# AI-Driven Personalized Learning

## My prompts to GPT-4

- ❖ I do not understand the physics of combustion. Please explain it to me via increasingly complex examples.
- ❖ Thanks. It makes sense. Please explain combustion in a car engine in more detail. I am a third year undergraduate student in Mechanical Engineering.
- ❖ Can you draw a simple diagram of a car engine?

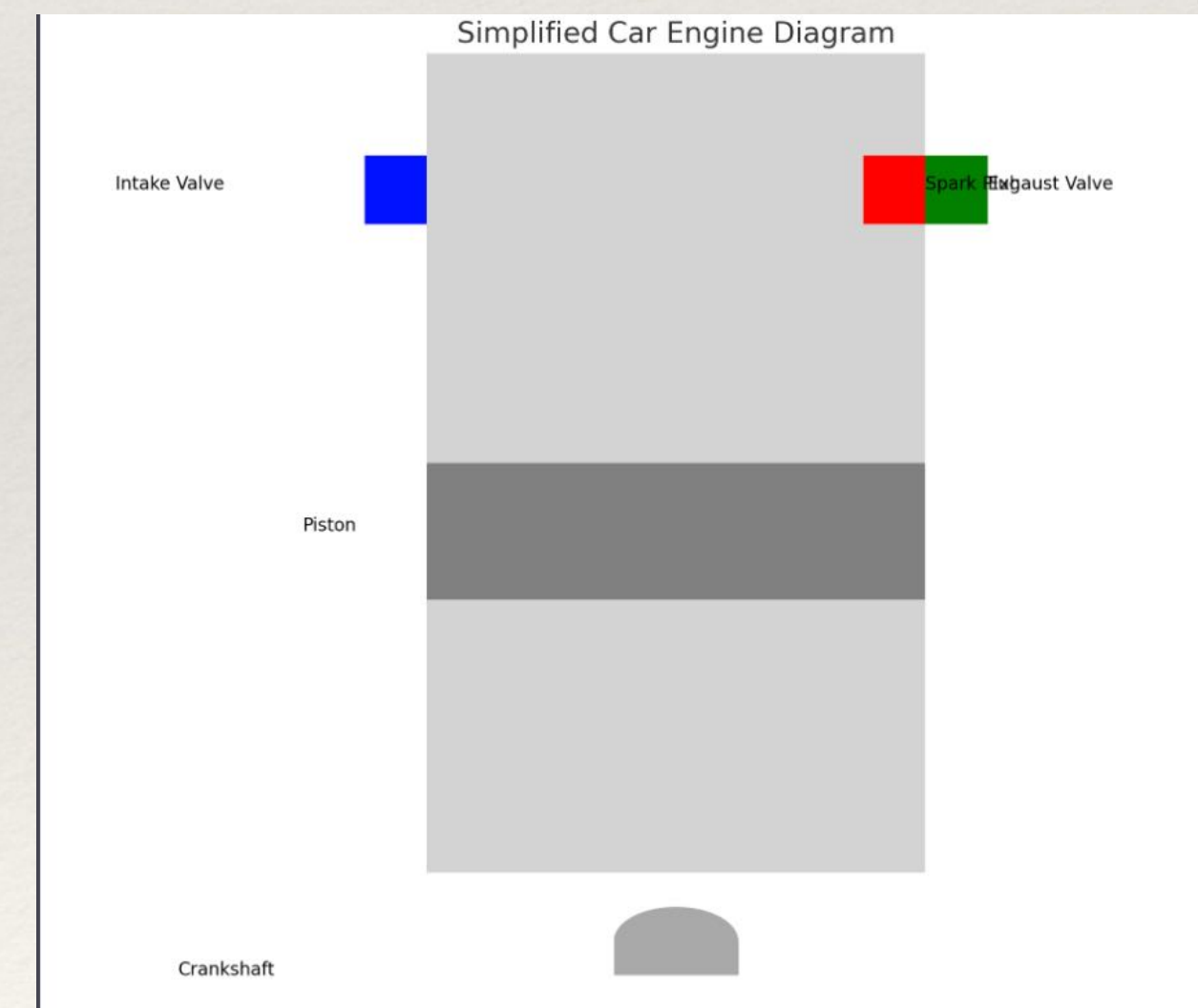
```
import matplotlib.pyplot as plt

def draw_engine_diagram():
    fig, ax = plt.subplots(figsize=(10, 8))

    # Cylinder
    cylinder_rect = plt.Rectangle((0.3, 0.2), 0.4, 1.2, facecolor='lightgrey')
    ax.add_patch(cylinder_rect)

    # Piston
    piston_rect = plt.Rectangle((0.3, 0.6), 0.4, 0.2, facecolor='grey')
    ax.add_patch(piston_rect)
```

Partial Python code





---

# Create a Lesson Plan

---

❖ Example: Create a lesson plan about interpolation 

❖ **My initial prompt:**

You are a professor of numerical algorithms, and I am interested in learning about 1-D interpolation techniques over a set of points. Develop a series of lessons starting with the basics and progressively becoming more complex. After each lesson, propose a quiz (a mixture of true/false and multiple choice) to test my knowledge. Once I answer, provide the correct results.

---

# Create a Quiz

---

## My prompts

- ❖ You are a professor in Computer Science and an expert in the C programming language. I will ask you to create a series of quizzes with various characteristics. If you need equations, write them in Latex. Code should be delimited by three double quotes on either side of the code, and indents must be maintained. Do you understand?
- ❖ Please create a True/False quiz with 5 questions on the structure of single and nested for loops.
- ❖ Thanks. Create a quiz with three questions. In each questions, provide a multiple choice questions with three choices. Only one choice must be correct and the two others false.
- ❖ Now create a quiz with the following structure: create a code snippet, and ask the student what the code accomplishes. Also provide the correct answer. Concentrate on different aspects of function arguments.

# Avoiding Cheating

- ❖ Step 1:

- ❖ Create a quiz of any type that Canvas can auto-grade

1. Does the sun radiate heat?

True

False

Standard approach on Canvas

- ❖ Step 2:

- ❖ Put the quiz questions on paper provided to the students

1.

True

False

Suggested approach On Canvas

- ❖ On canvas, only put the choices for the student to fill out

1. Does the sun radiate heat?

Paper test Student answers on Canvas

# Automatic Grading: Gradescope

Paper-based Assignment  Programming Project

6. (10 pts) Refer to the previous problem for an explanation of the context of this code. Fill in the missing line.

It can be solved with one line but there are multiple possible approaches. If your solution requires two or three lines, fill in those lines above and below the blank as needed.

```
/** replace last factor with the value i */
public void replaceLastFactor(int i) {
```

```
    int prev = data.set(data.size()-1, i);
}
```

3 +5.0

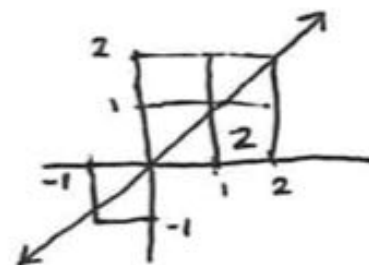
Partial credit: An answer that has the form `x.set(x.size()-1, i)` but where `x` is some variable that is either out of scope, or an inappropriately declared instance variable.

4 +5.0

(c) Suppose that  $f(x)$  is a continuous function where  $\int_2^7 f(x) dx = 11$ .

For the following expression, evaluate it or state that there is not enough information.

$$\int_2^{-1} (f(x) + x) dx + \int_{-1}^7 f(x) dx$$



$$\int_2^{-1} f(x) dx + \int_2^{-1} x dx + \int_{-1}^7 f(x) dx$$

$$\int_2^7 f(x) dx + \int_2^{-1} x dx$$

$$11 + \int_2^{-1} x dx \quad 11 + 2 - \frac{1}{2} = 11 + 1.5 = 12.5$$

Total Points  
0.5 / 1.0 pts

1 -0.0

Correct

2 -0.5

You had trouble calculating

$$\int_{-1}^2 x dx$$

Now that we have spent some time practicing how to integrate, go back and look over this.

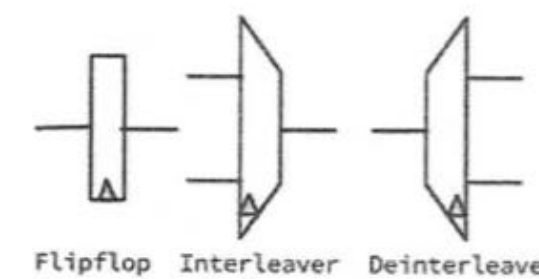
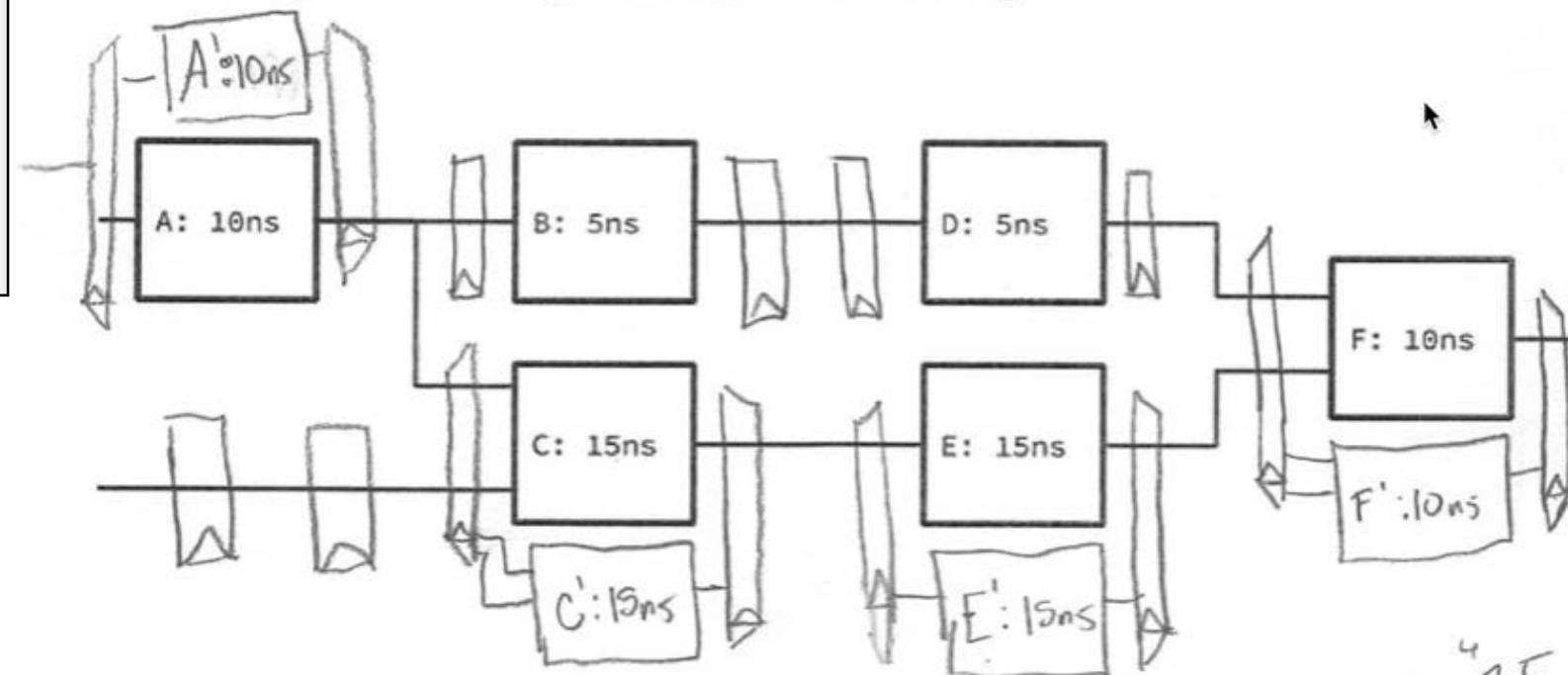
3 -0.5

$$\int_2^{-1} x dx \neq \int_{-1}^2 x dx$$

Why not? What is the big difference? Can you use the net change lens to justify this?

Question 2 ( points total):

Pipeline the circuit below. Optimize throughput (inputs processed per ns) and cost (\$). You may add any number of the blocks in the circuit (A-F), edge-triggered flipflops, edge-triggered interleavers, an edge-triggered de-interleaver (see figure below). No other modifications are permitted. The latency of flipflops, interleavers, and de-interleavers is 0ns. The cost of any item (whether already there or one you add), regardless of which item is \$1. You have a total budget of \$25 and the components already drawn below cost \$6. Draw your answer on top of the diagram below if at all possible.



$$\text{Cost} = \$6 + \$18 = \$24$$

$$\text{Latency} = 7.5 \times 8 = 60 \text{ ns}$$

$$\text{Throughput} = \frac{1}{7.5 \text{ ns}}$$

Total Points

24.0 / 33.0 pts

1 +6.0

Correct interleaving concept

2 +6.0

Correct interleaving of multiple inputs

3 -3.0

Abused interleavers (too many outputs/inputs)

4 +6.0

Well-formed ("balanced" latch count)

5 +8.0

Correct extra stages for long-latency interleaved approach

6 -4.0

---

# AI: Embrace or Resist?

---

- ❖ Some schools tried to ban GPT technology
  - ❖ Doomed to failure
- ❖ AI tools are proliferating
- ❖ Technology improving at accelerated pace
- ❖ Students with AI usage expertise will get better jobs
- ❖ AI will allow students to learn more and faster

# Practical Applications in Coding

---

# Visual Studio Code

---

- ❖ Probably the best IDE for Python, Javascript, and Julia code
- ❖ Under heavy development
- ❖ Hundreds of Extensions, in particular
  - ❖ Copilot: plugin, free for students and faculty
- ❖ Genie: Interface to multiple chatbots (next slide)
  - ❖ Interface to OpenAI via an API Key

---

# Copilot

---

- ❖ Coding Assistant
  - ❖ Improves command-line completion
  - ❖ Can write entire functions
  - ❖ Use meaningful names (think of best programming practice)
  - ❖ Divide your problem in digestible steps
  - ❖ Strongly dependent on user input and previous code
- ❖ Operates on Github, Visual Studio Code (ctrl-cmd-V)



```
1 %load_ext autoreload
2 %autoreload 2
```

✓ 0.0s

```
1 import torch
2 import torch.tensor as tt
3 import torch.nn as nn
4 import torch.nn.functional as F
5 import numpy as np
```

✓ 0.8s

```
1 # Create a module with multiple activations.
2 # Input is x, which branches off to two activations, which are then recombined
3 # The recombination is done by a linear layer.
4 class MultiActivation(nn.Module):
5     def __init__(self, in_features, out_features, activation1, activation2):
6         super().__init__()
7         self.activation1 = activation1
8         self.activation2 = activation2
9         self.linear = nn.Linear(in_features, out_features)
10
11     def forward(self, x):
12         x1 = self.activation1(x)
13         x2 = self.activation2(x)
14         x = torch.cat([x1, x2], dim=1)
15         x = self.linear(x)
16         return x
```

# Multi-activation

- ❖ Construct a layer that executes:

$$y = W_2\sigma(Wx + b_1) + W_3\text{ReLU}(W_4x + b_2) + b_3$$

- ❖

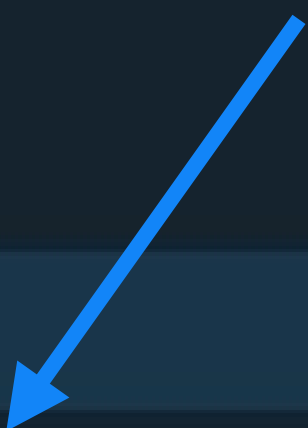
```
1 %load_ext autoreload
2 %autoreload 2
```

✓ 0.0s

```
1 import torch
2 import torch.tensor as tt
3 import torch.nn as nn
4 import torch.nn.functional as F
5 import numpy as np
```

✓ 0.8s

Instructions  
to Copilot



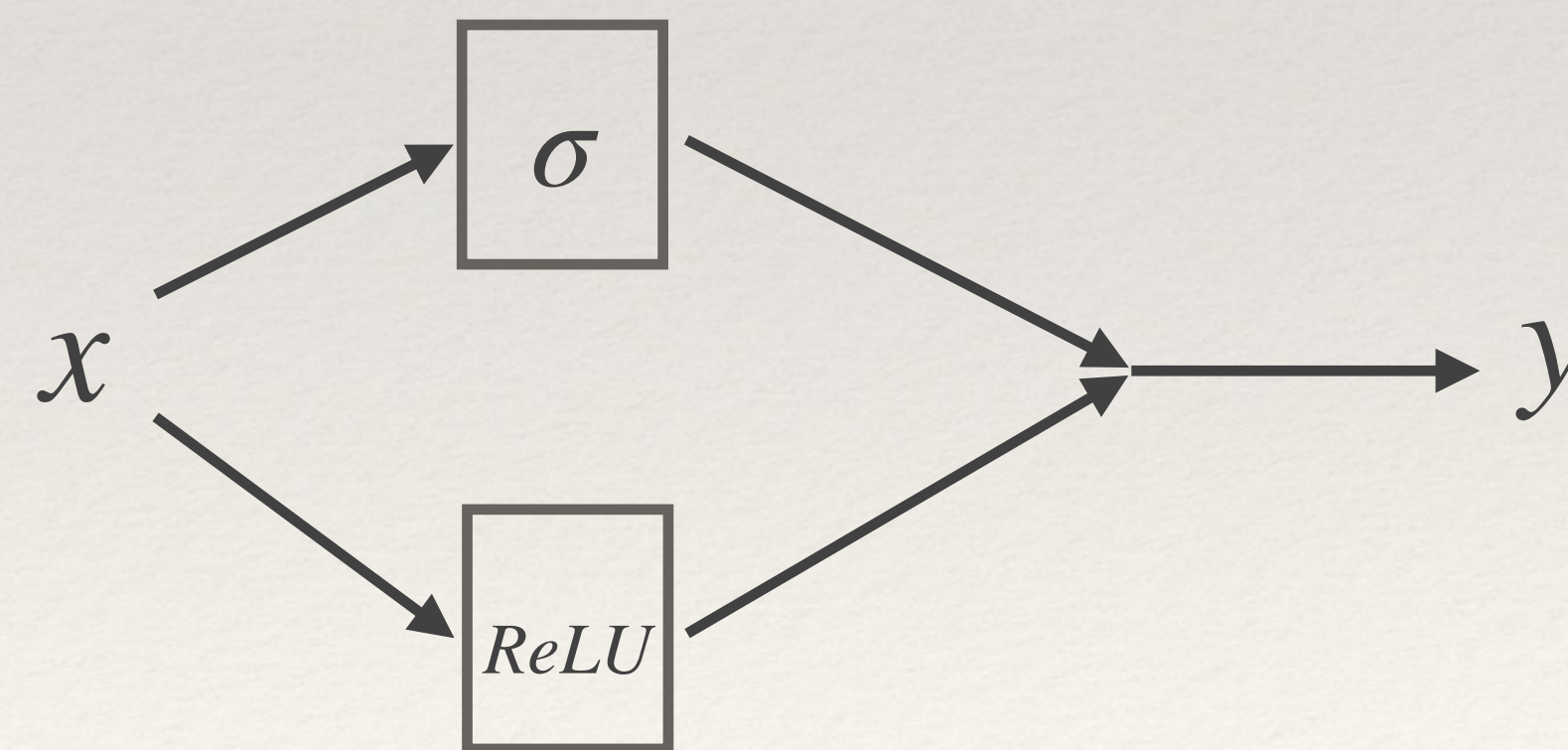
```
1 # Create a module with multiple activations.
2 # Input is x, which branches off to two activations, which are then recombined
3 # The recombination is done by a linear layer.
4 class MultiActivation(nn.Module):
5     def __init__(self, in_features, out_features, activation1, activation2):
6         super().__init__()
7         self.activation1 = activation1
8         self.activation2 = activation2
9         self.linear = nn.Linear(in_features, out_features)
10
11     def forward(self, x):
12         x1 = self.activation1(x)
13         x2 = self.activation2(x)
14         x = torch.cat([x1, x2], dim=1)
15         x = self.linear(x)
16         return x
```

# Multi-activation

- ❖ Construct a layer that executes:

$$y = W_2\sigma(Wx + b_1) + W_3ReLU(W_4x + b_2) + b_3$$

- ❖



---

# Alternatives to Copilot

---

- ❖ IDX
- ❖ CodeGeex AI
- ❖ Tabbing
  
- ❖ They all have similar capabilities to Copilot
- ❖ Tools will greatly improve in the very near future

---

# Ethical Considerations



---

- ❖ Copilot was trained in public repositories
  - ❖ some code has copyrights
- ❖ Copilot does not provide sources for its suggestions
- ❖ Proposed approach:
  - ❖ For substantial code segments generated by AI state attribution

---

# Creative Uses of LLM

---

- ❖ Drawing a house 
- ❖ Uniformize the style of a grant written by multiple participants
- ❖ Generate latex for complex equations
- ❖ Remove grammatical errors in a paper 
- ❖ Build citation lists
- ❖ Copy / paste a table from the Internet and translate it into a variety of formats
- ❖ Convert code from one language to another
- ❖ Generate quizzes for a class ([demonstrate with GPT4](#))
- ❖ Generate a lesson plan to learn ([demonstrate with GPT4](#))

---

# Conclusion & Future Outlook

---

- ❖ 2022 was the year of Generative AI
  - ❖ Chatbots
  - ❖ Image generators
- ❖ 2023 and beyond
  - ❖ Bigger general models
  - ❖ Speculation is that GPT-5 will be released in 1-2 years
  - ❖ Multimodal AIs that combine text, images, video in both directions

---

# Bold Predictions

---

- ❖ AI coding will improve.
- ❖ Easier idea-to-code conversion.
- ❖ Focus more on algorithms.
- ❖ Fewer AI hallucinations.
- ❖ Prioritize the "what" over the "how".
- ❖ Manipulate math as easily as natural language.
- ❖ Improved presentation software
- ❖ Coding language importance will diminish.
- ❖ Eliminate the need for a shared oral language.
- ❖ Embrace the change!



The end

Questions?