

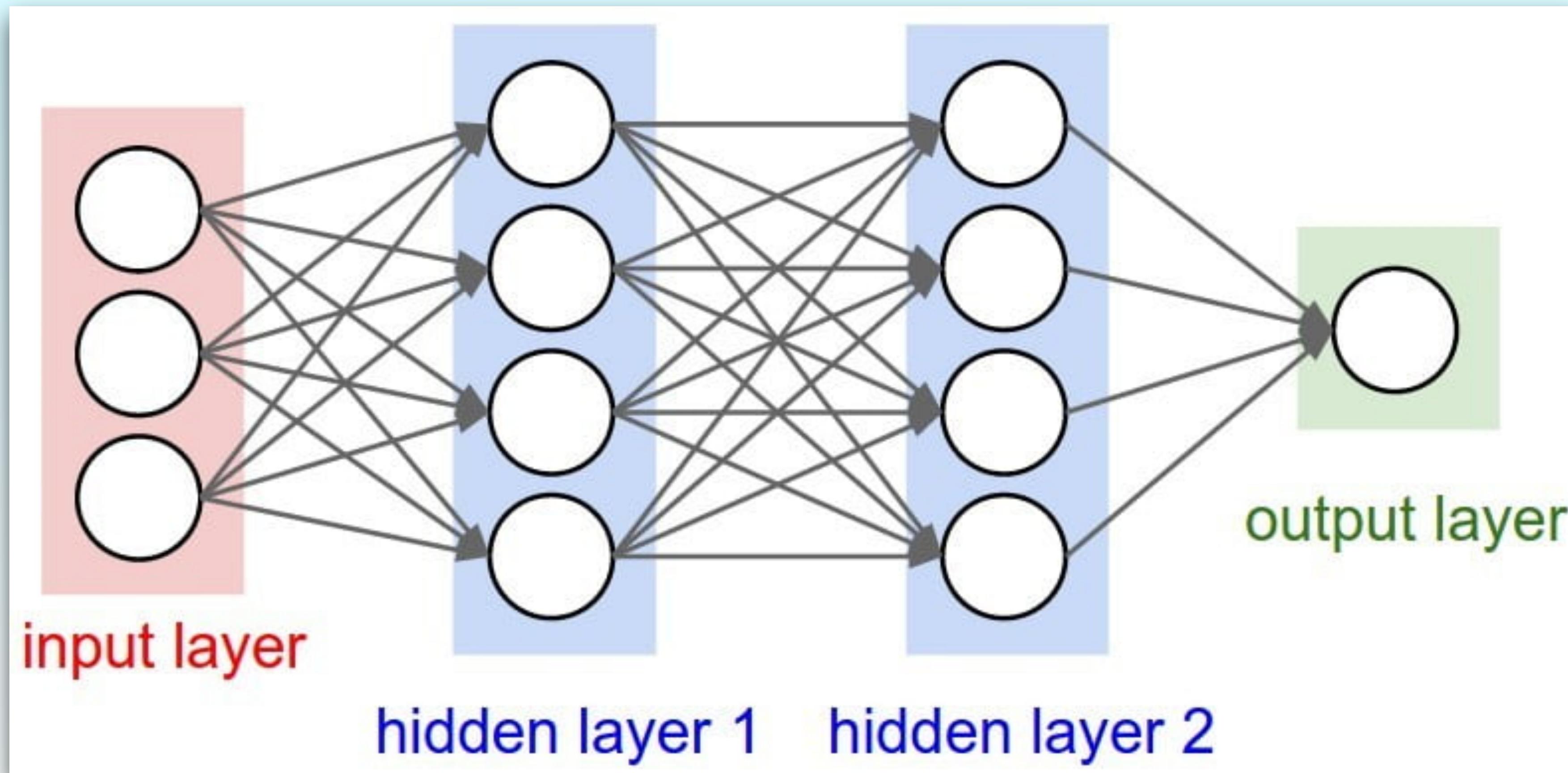
# NLP

## LauzHack Deep Learning Summer Bootcamp

**July 2024**

# Let's have a recap!

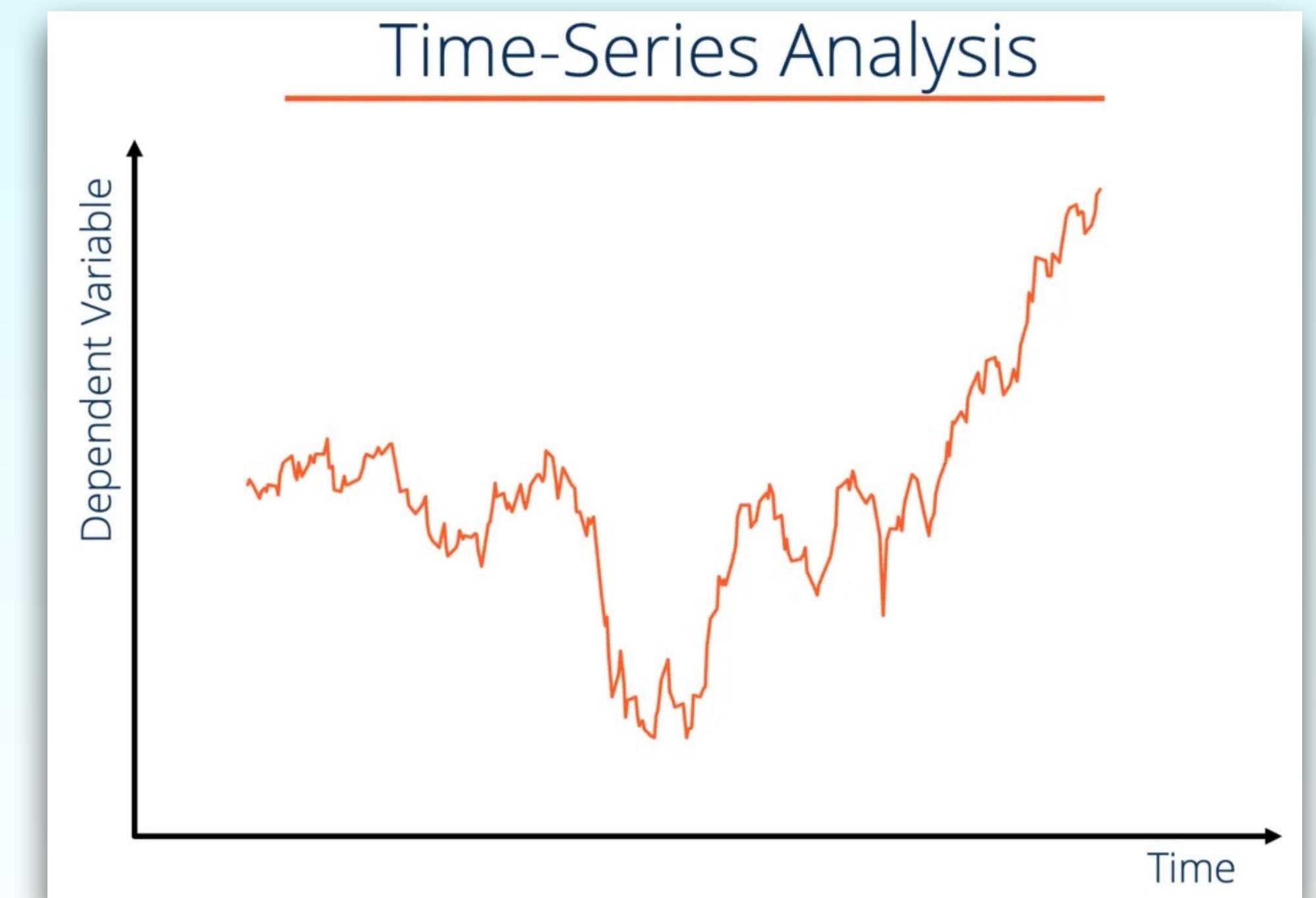
## Deep Neural Networks



# Let's have a recap!

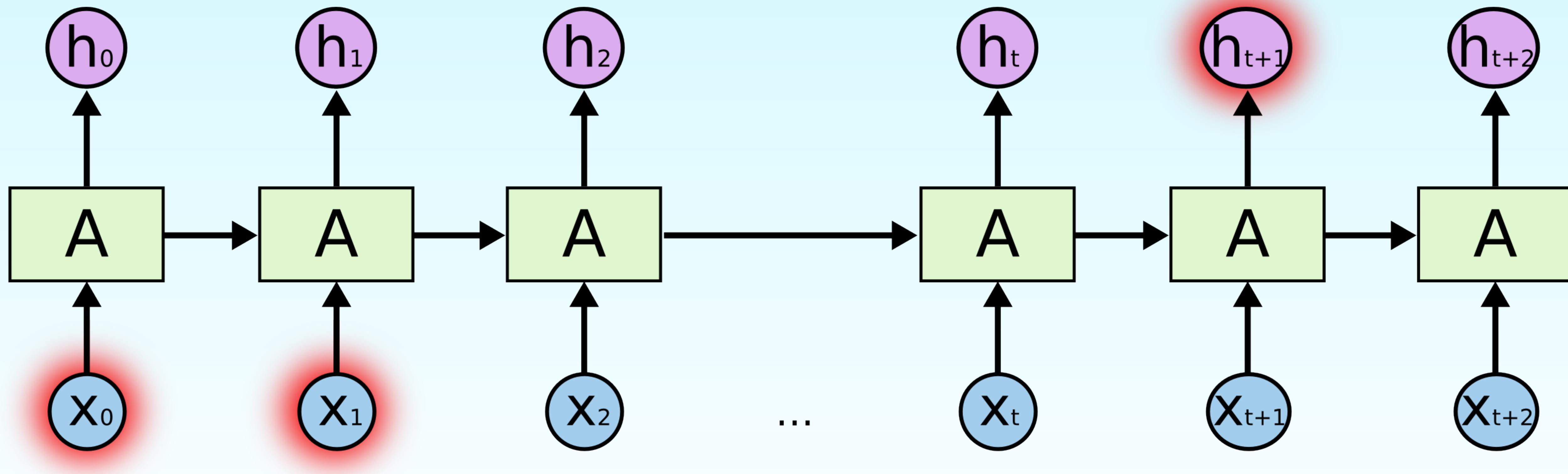
## Time series

- Data measured at regular intervals, or step-by-step. Examples:
  - Hourly temperature in a city
  - Heights of ocean tides
  - Heartbeats per minute
  - The stock price of a company
  - Answers of students to exam questions
  - A written text (!): *EPFL I like vs. I like EPFL*
- Tasks: *classification* or *forecasting*
- Tip: if data semantics change by permutation



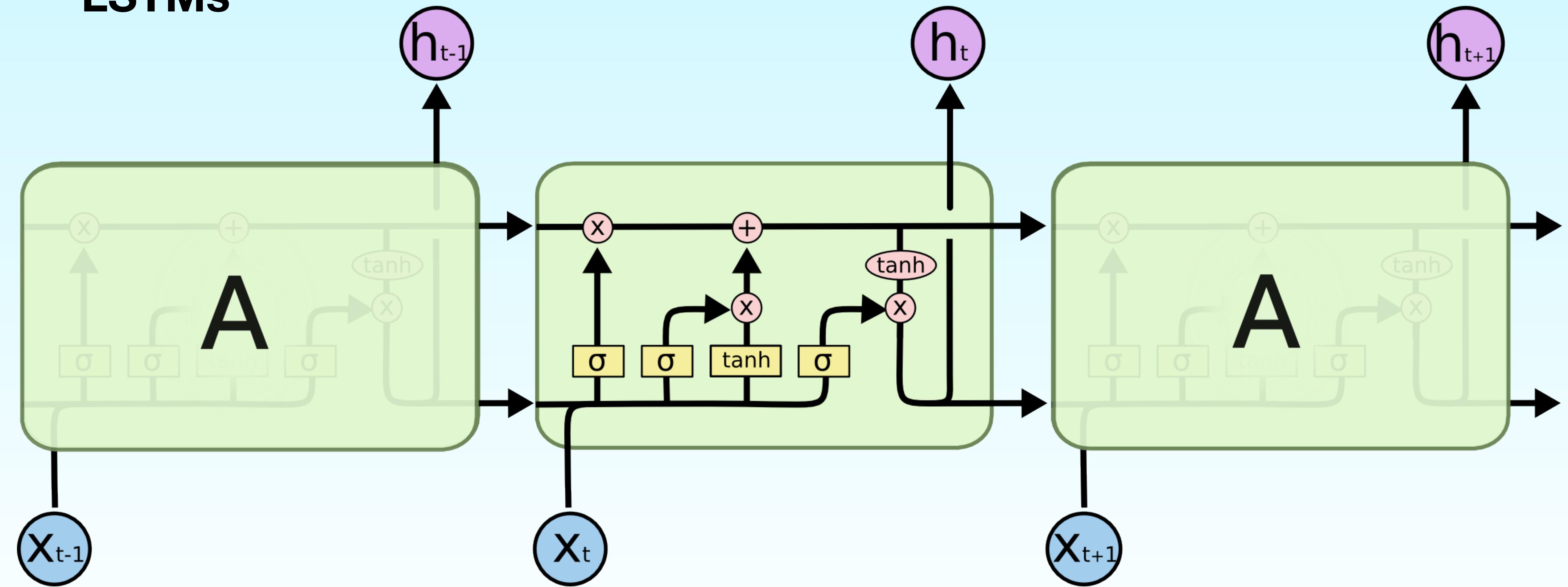
# Let's have a recap!

## Recurrent Neural Networks

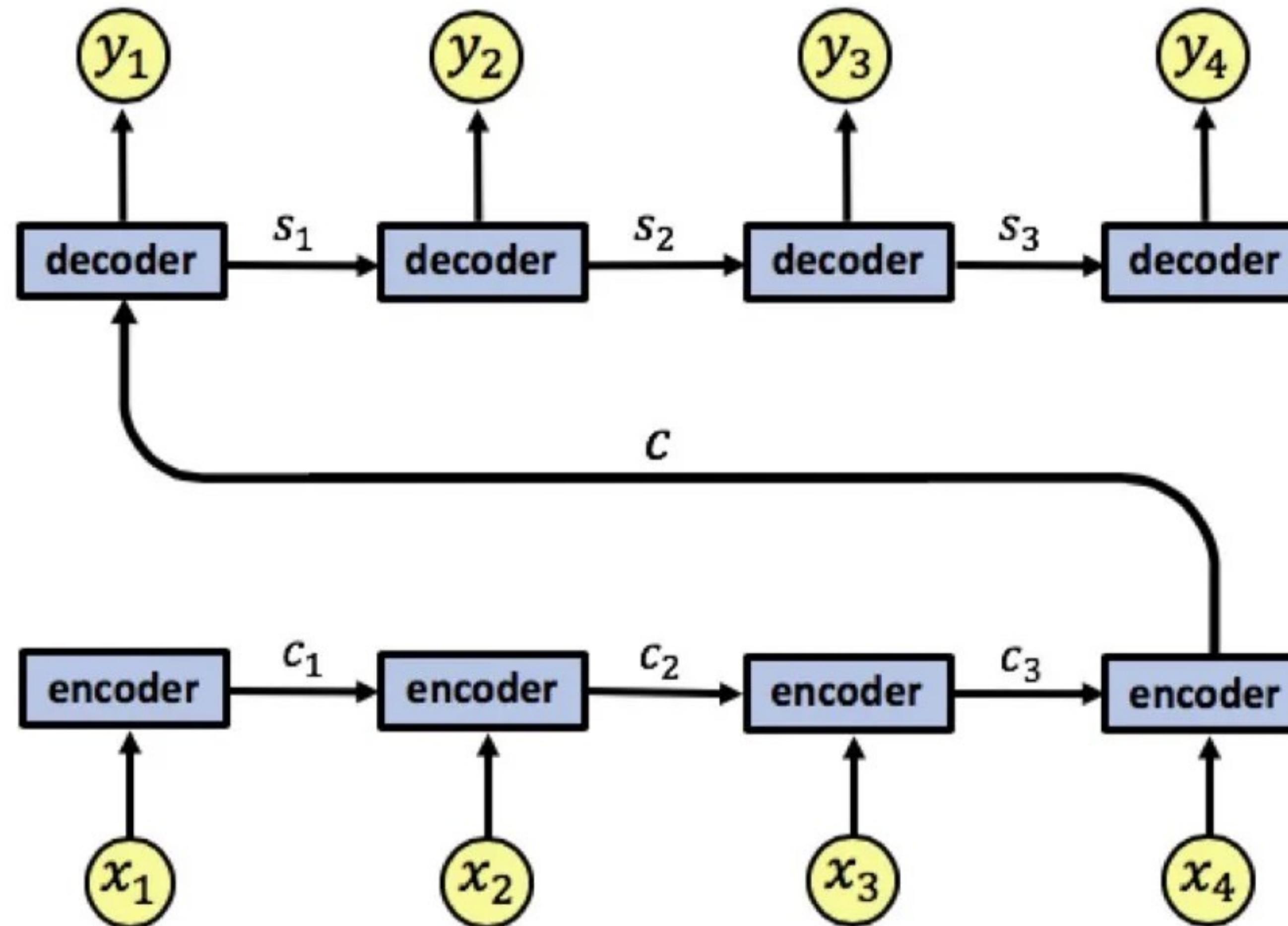


# Let's have a recap!

## LSTMs



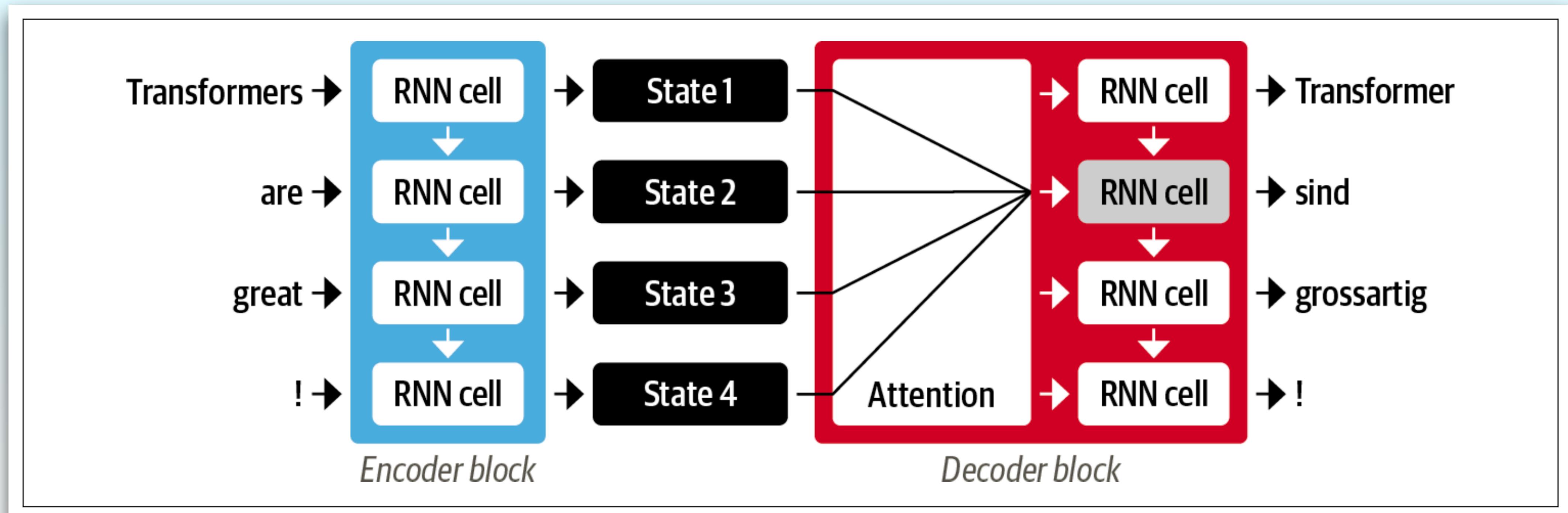
# Seq2seq Models



# Attention Mechanism

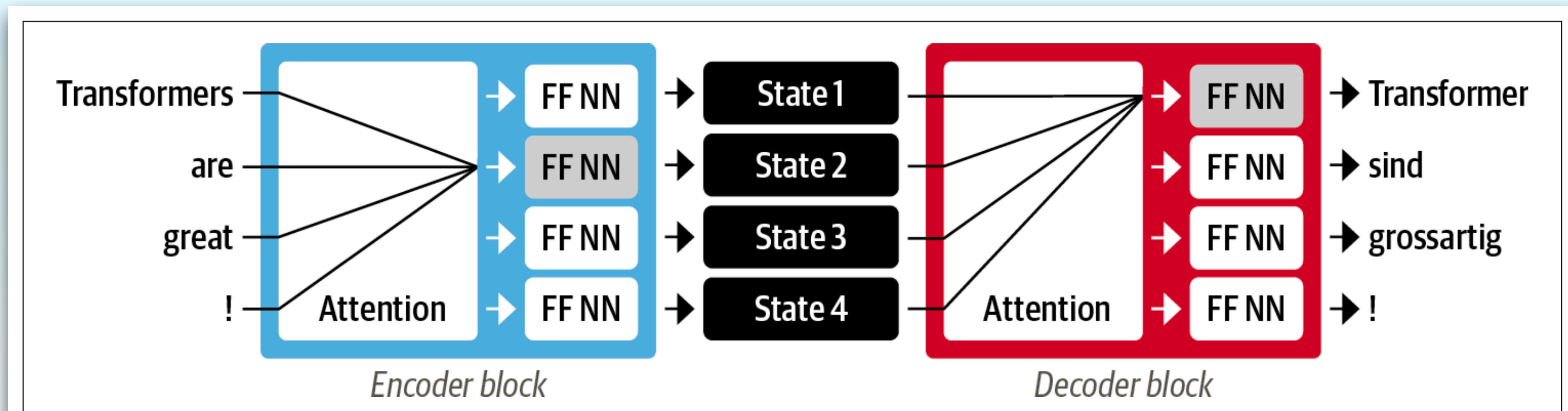
From Tunstall et al.:

“The main idea behind attention is that instead of producing a single hidden state for the input sequence, the encoder outputs a hidden state at each step that the decoder can access.”



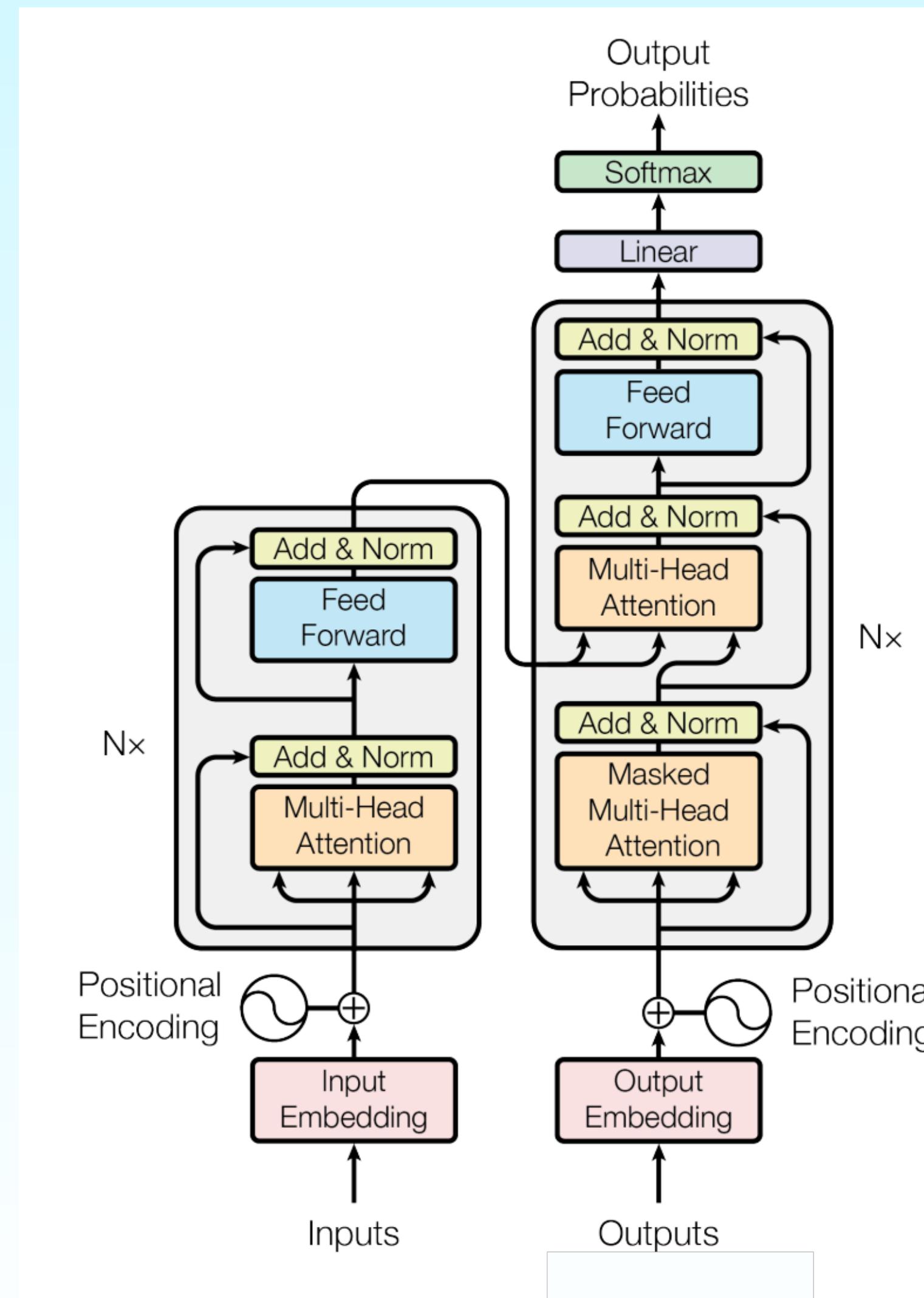
# Transformer

- Remove the recurrence! (Self-)attention is all you need!
- Encoder and Decoder
- Input can be text, image, ... (but has small details to take care of!)



*Figure 1-6. Encoder-decoder architecture of the original Transformer*

# Transformer in one view!



**Let's explore NLP as a specific  
deep learning task**

# Natural Language Processing

- Any task that works with textual data
- Imagine a model saying if a certain sentence is happy or sad
- Three components:
  - 1) Represent words in a computer-understandable way: **embeddings**
  - 2) The model that gets the words as input
  - 3) Mapping between model outputs and predictions (e.g., the last layer)
- You already know 2 and 3...

# Tokenization

- Each sentence contains several words
- Each “token” is a meaningful unit of characters -> discretization!
- “I am attending the summer bootcamp!”
  - I + am + attending + the + summer + bootcamp + !
- *Is this the only way to tokenize the sentence above?*

# Why not each word as a token?

- “I am attending the summer bootcamp!”
  - I + am + attending + the + summer + bootcamp + !
- How do you tokenize “check-in” or “six-years-old” or “isn’t”?
- “attend” and “attending” are then totally different tokens, so why not be close?
  - The number of words increases a lot for each variation -> computation
- Some languages do not use space as English does: Korean, Chinese, etc.
- Example alternative:
  - I + am + attend + ing + the + summer + boot + camp + !
- Specific tokens might be needed, e.g., <UNK>

# Vocabulary

- Set of all possible tokens
- Any token not in the vocabulary  $V \rightarrow \text{<UNK>}$
- We should represent each word of  $V$  as a *vector*
- One early solution: one-hot vectors of size  $V$ 
  - I  $\rightarrow [1, 0, 0, 0, \dots]$
  - am  $\rightarrow [0, 1, 0, 0, \dots]$
  - attend  $\rightarrow [0, 0, 1, 0, \dots]$
- Downside?

# Embeddings

- We want word vectors to contain semantic information
- One-hot does not provide it; how can we find the similarity between these vectors?
  - like -> [0 0 ..... 1 ..... 0]
  - love -> [0 0 ..... 1 ..... 0]
  - parameter -> [0 0 ..... 1 ..... 0]
- Normal time series has semantic information: numbers can be huge, small, or have patterns
  - So, in time series for NLP, the inputs (word vectors) should have meanings
  - We should use vectors with real numbers as entries, but how to find them?

# Embeddings: learn from the model

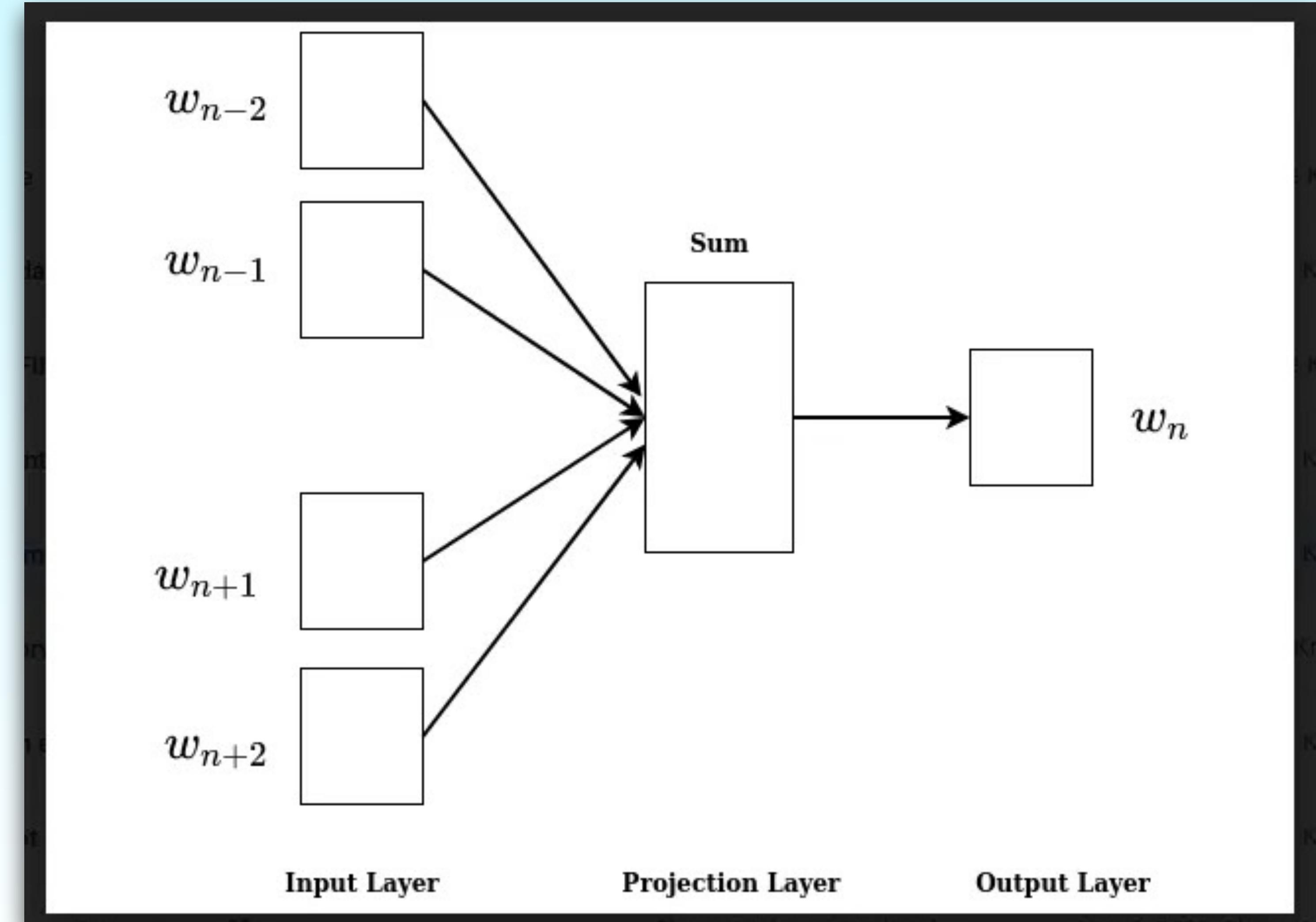
- One possible solution: add feedforward networks, converting one-hot to real-valued vectors
- Is it enough?
- Usually labeled training data is rare, not enough to capture enough dependencies between words
- Idea: use vast amounts of unlabeled text (e.g., on the internet) to find the embeddings, then use those embeddings in our models trained using labeled data
  - Fictional example: use the whole Wikipedia (billions of words) to find embeddings, then use them to train a Transformer or RNN on 1000 sentences to classify sad/happy

# Self-supervised Learning of Embeddings

- Find patterns by looking at large bodies of text and words that come next to each other
  - *Imagine a model looking at the whole internet; it should learn that Bern, Zurich, Lausanne, etc. come with the word Switzerland regularly, so they are Swiss cities*
  - One pre-transformers approach: word2vec continuous bag of words (CBOW)
    - A model to *predict* a word in a gap, from the surrounding
    - I am ..... the summer bootcamp!
      - attending, enjoying, ...

# CBOW

- A certain window size
- When training the model to predict the missing word, it automatically infers the embeddings
- On a big corpus
  - For example: “brown” (1 million words)
  - Why not directly training on the targeted data?



**Let's see in the code!**

# Embeddings

- Now we know how to form the inputs to the models
  - Previously: models with limited labeled data also had limited semantic information of word similarities
  - Now the models work with limited labeled data, but with embeddings trained on a vast corpus
- Next step: why not also *train* the models on a vast amount of data?
- ... Transfer Learning, similar to computer vision

# Transfer Learning in NLP: pre-train + fine-tune

- Model: body + head
  - Body: main weights of the model, constant between tasks; Head: task-specific
- Procedure: train a model on a vast amount of data, then for each specific task, train the head on that task
  - As Tunstall et al. say: “During training, the weights of the body learn broad features of the source domain”
  - Allows much less training data to be effective!
  - Also learns good embeddings in the process
  - OpenAI was a pioneer around 2017 / 2018 :)
- But how to initially train the model? What is the objective?
  - In CBOW, it was predicting the gap word

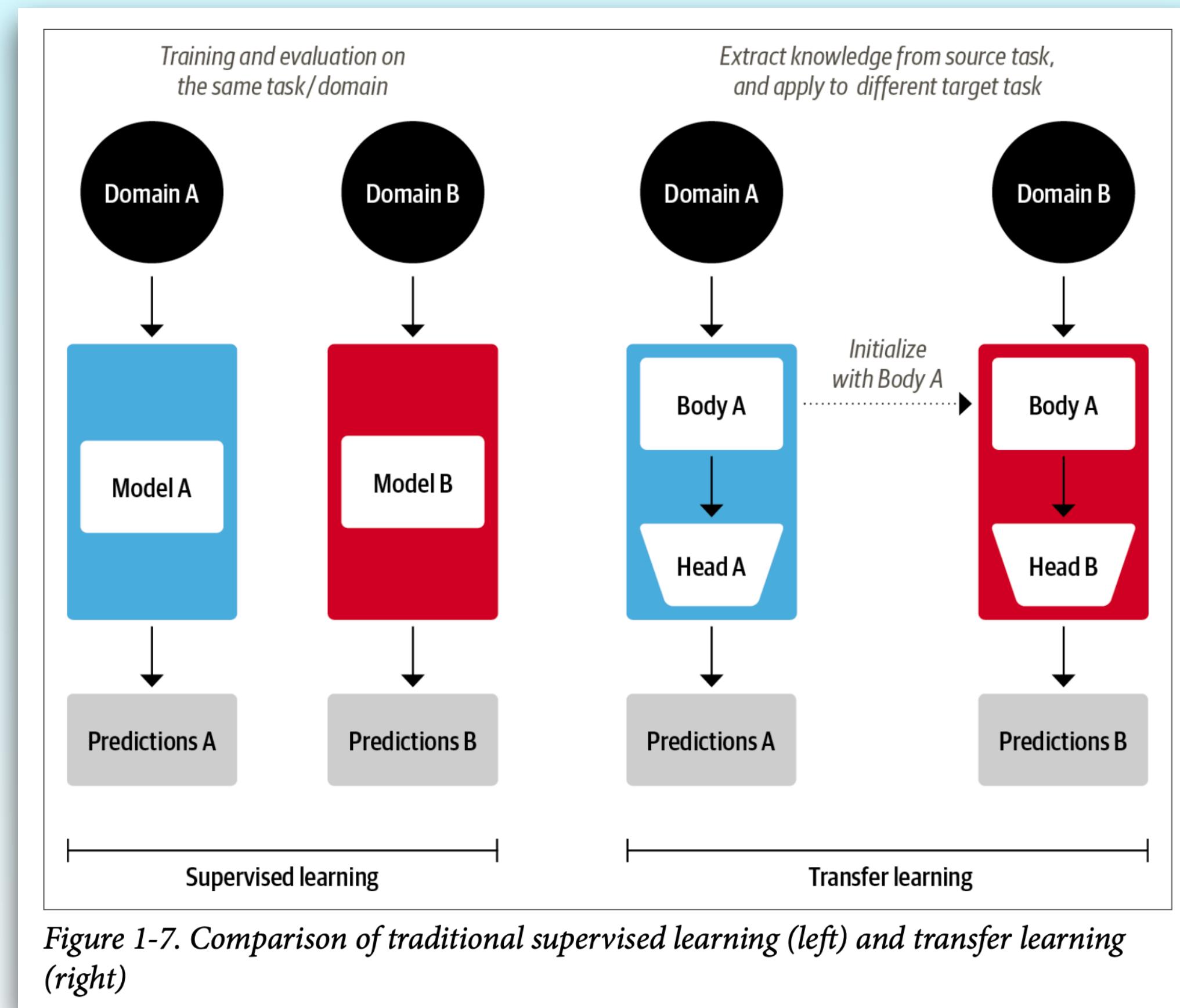


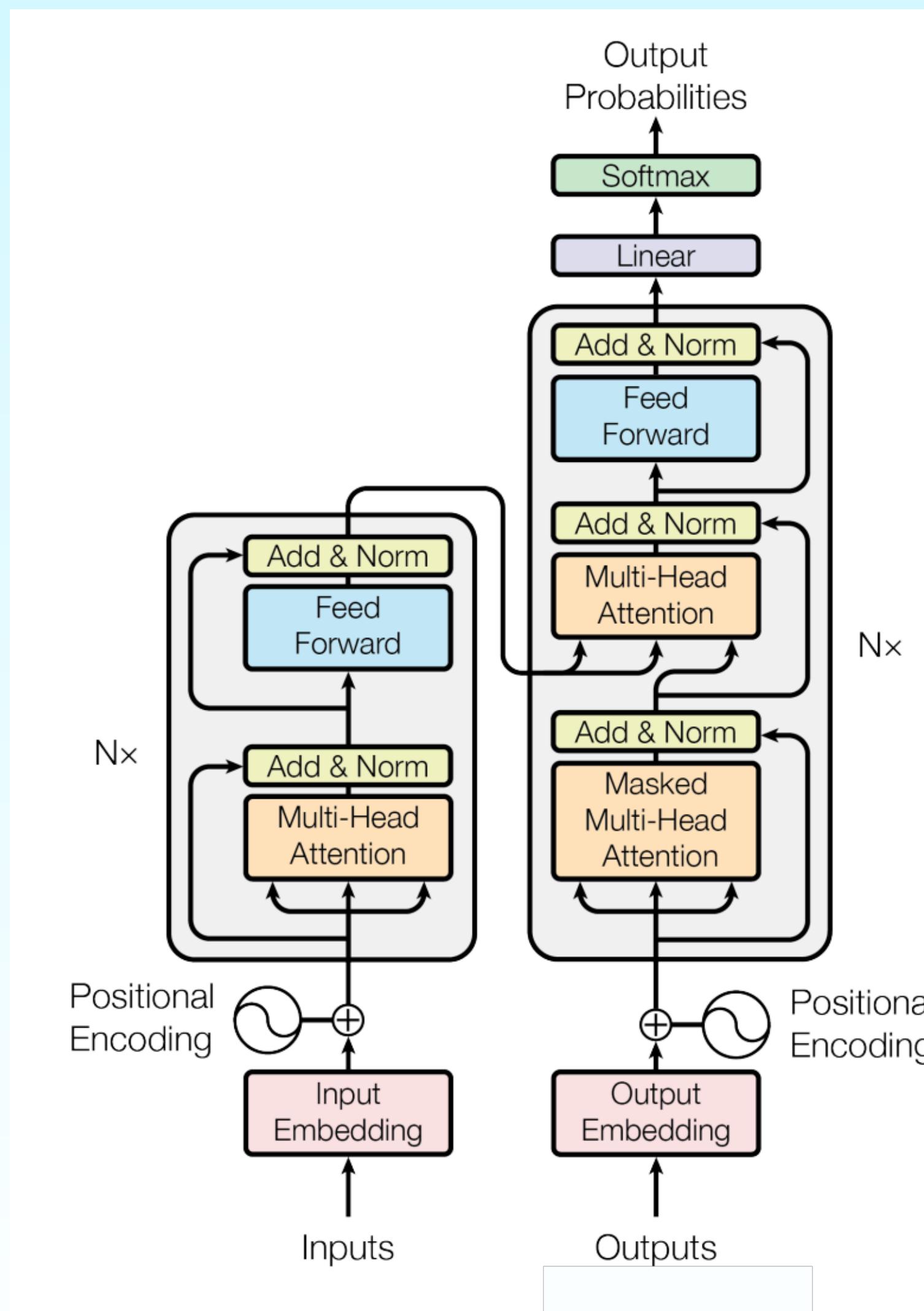
Figure 1-7. Comparison of traditional supervised learning (left) and transfer learning (right)

Tunstall, Lewis, Leandro Von Werra, and Thomas Wolf. Natural language processing with transformers. " O'Reilly Media, Inc.", 2022.

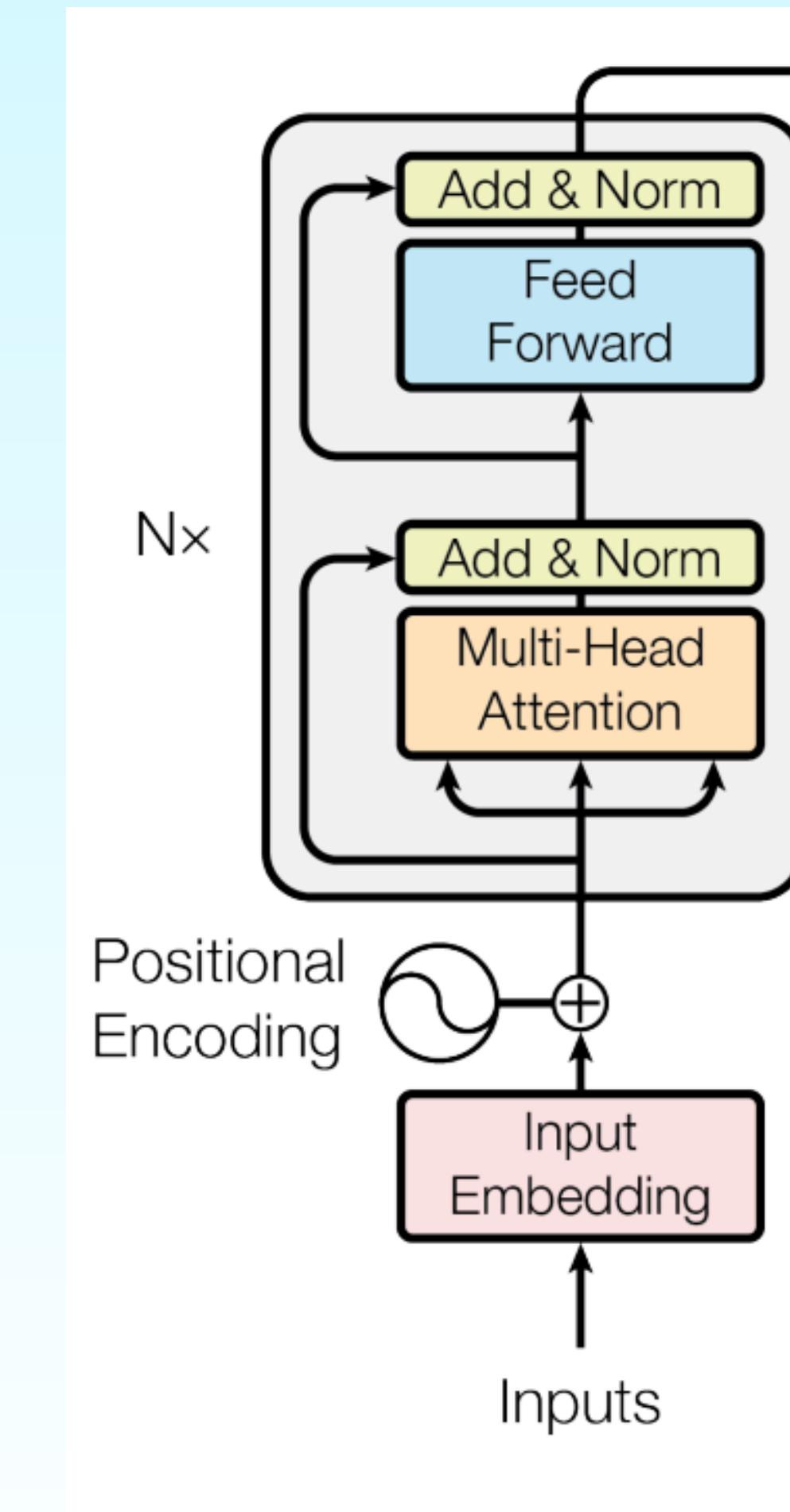
# Transformers + Transfer Learning

*Two ideas beyond a simple seq2seq!*

# Recap: Transformers



# Bidirectional Encoder Representations from Transformers (BERT)



# BERT

- By Google AI: Devlin et al., 2018
- BERT-base: 110 million parameters
- BERT-large: 340 million parameters

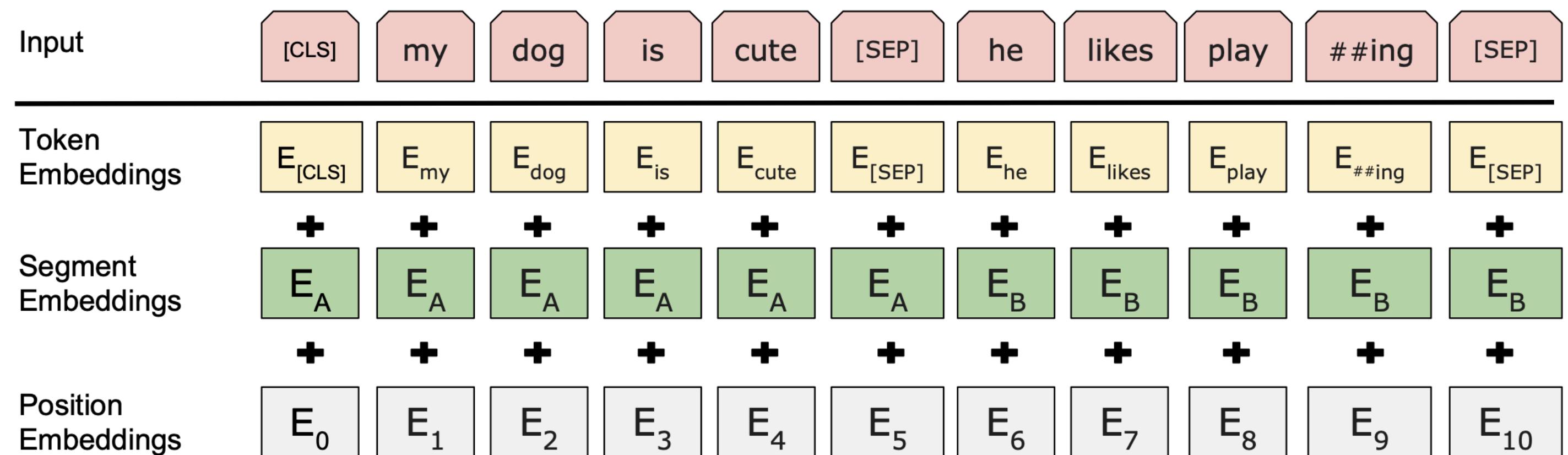
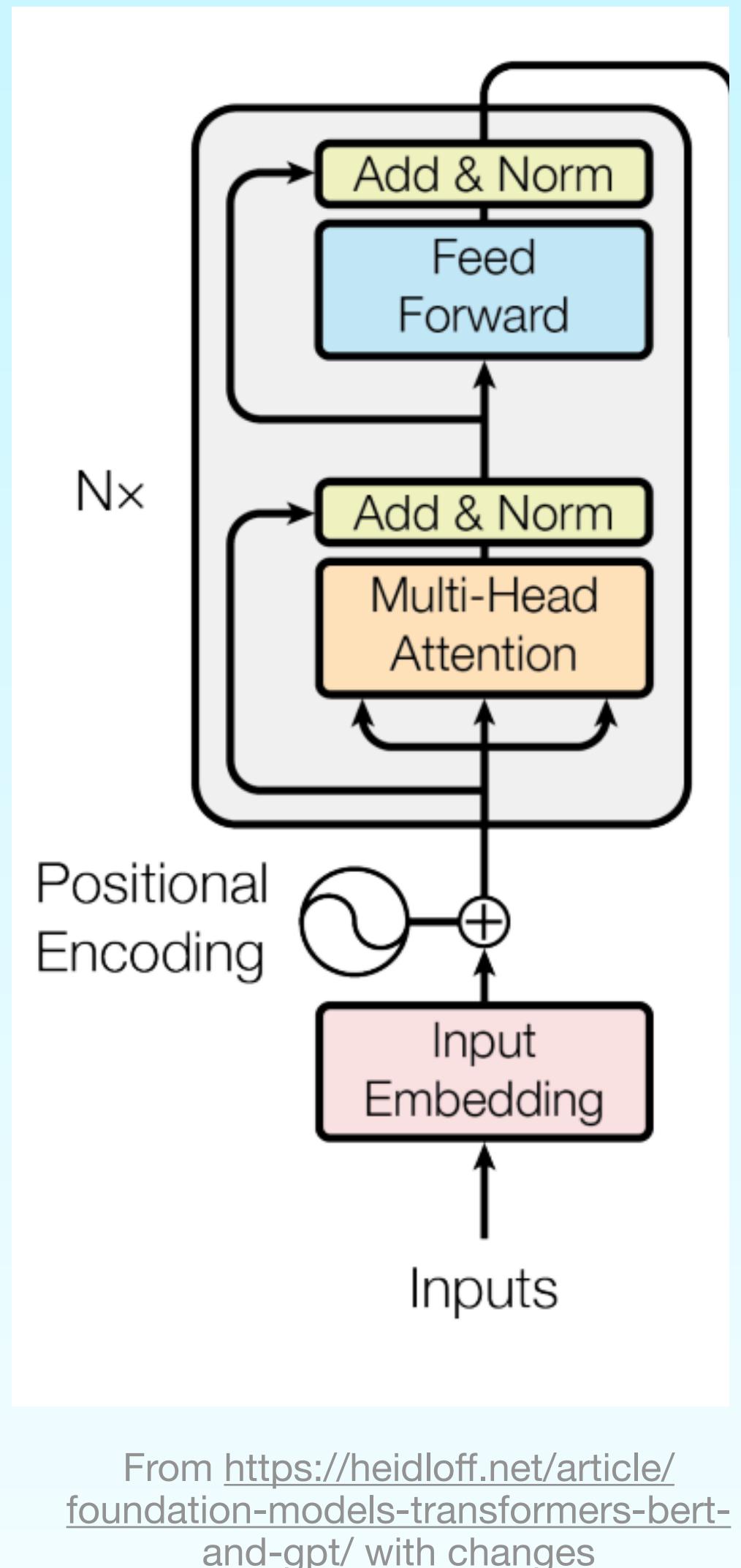


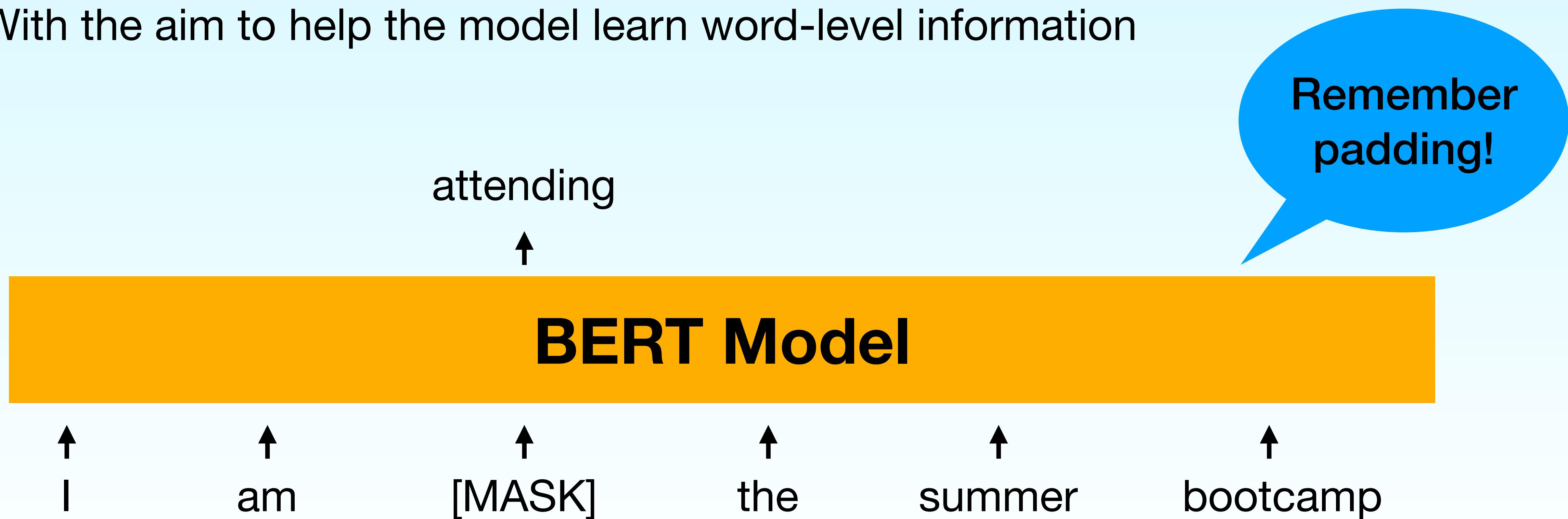
Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.



# BERT Pre-training (*simplified version*)

## 1) Masked Language Modeling

- I am [MASK] the summer bootcamp!
- With the aim to help the model learn word-level information



# BERT Pre-training (*simplified version*)

## 2) Predicting Text Continuation

- With the aim to help the model learn sentence-level information



I learned a lot



Steve Jobs was CEO of Apple

# BERT fine-tuning

For tasks other than filling missing words, e.g., classification

- Use output corresponding to [CLS], then a logreg – why [CLS] at *start*?

1  
↑

**BERT Model**

[CLS] I enjoyed attending the summer bootcamp

0  
↑

**BERT Model**

[CLS] I hated following the summer bootcamp

# Let's see in a code together!

**[bit.ly/bert-fine-tune-lauzhack](https://bit.ly/bert-fine-tune-lauzhack)**

More details in the fine-tuning  
session later this week



# **One downside of BERT?**

**Needs a lot of memory and inference time**

# How to reduce memory/time?

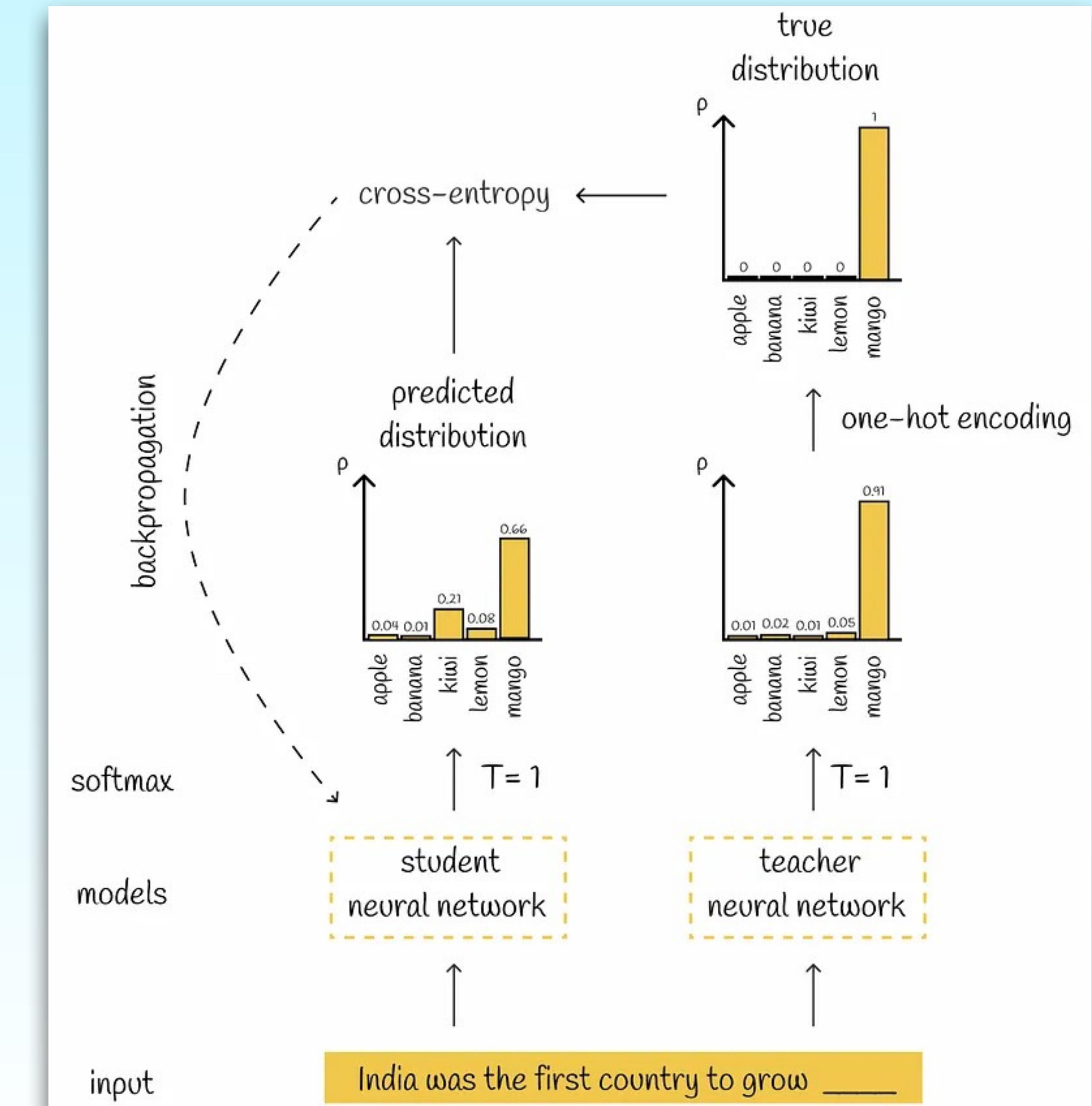
- One way: *quantization*
- Show numbers with less precision, e.g., float16 instead of float32
  - Some information loss
  - The benefits of less memory/time can be worth the small performance hits
- Can also convert to int -> should project certain ranges to certain integers

# How to reduce memory/time?

- Another way: *knowledge distillation* -> DistilBERT
  - We train a smaller model (*student*) to replicate/imitate how a bigger model (*teacher*) behaves

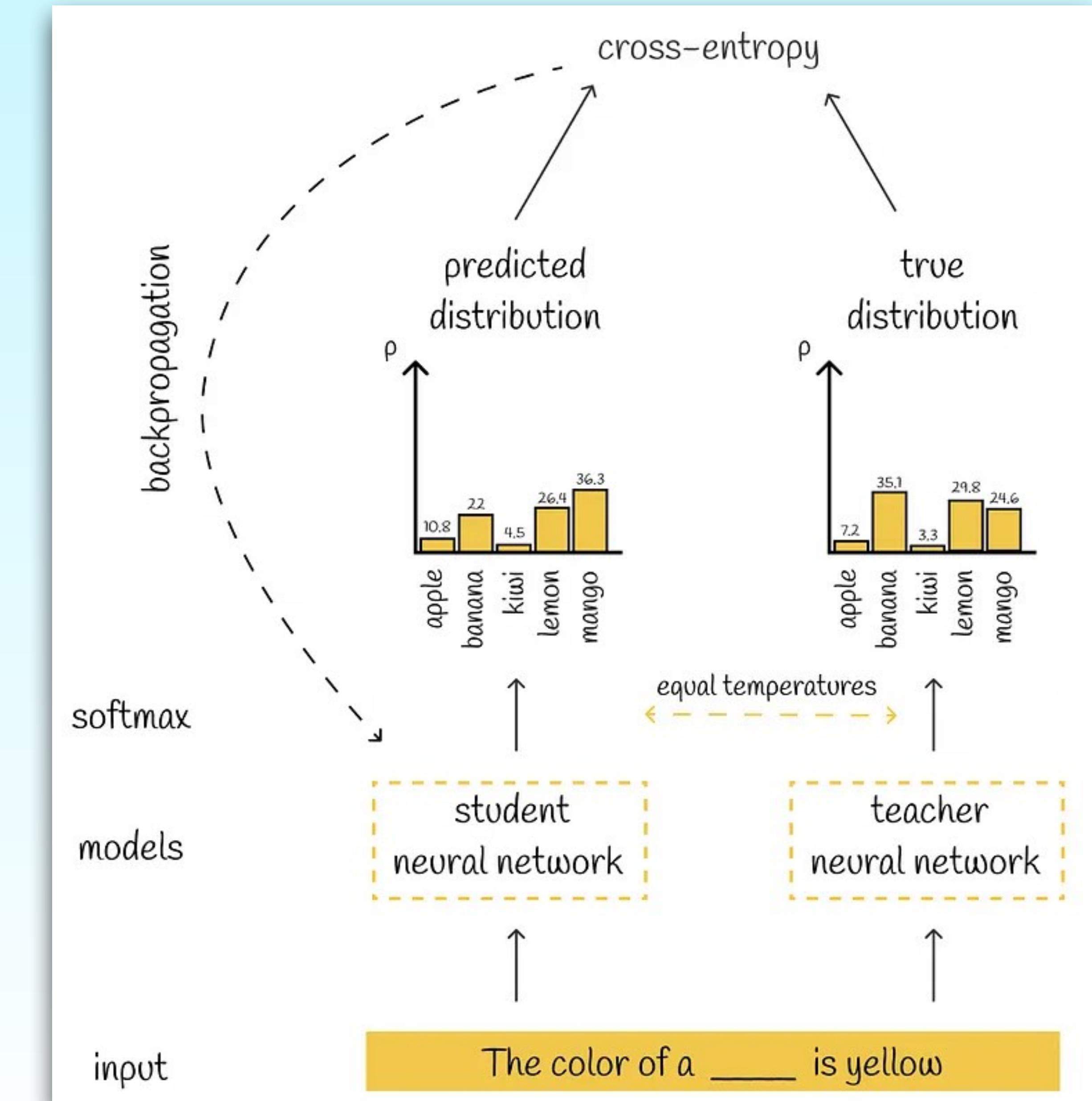
# DistilBERT Losses

- 1) MLM loss, but compares with the teacher model



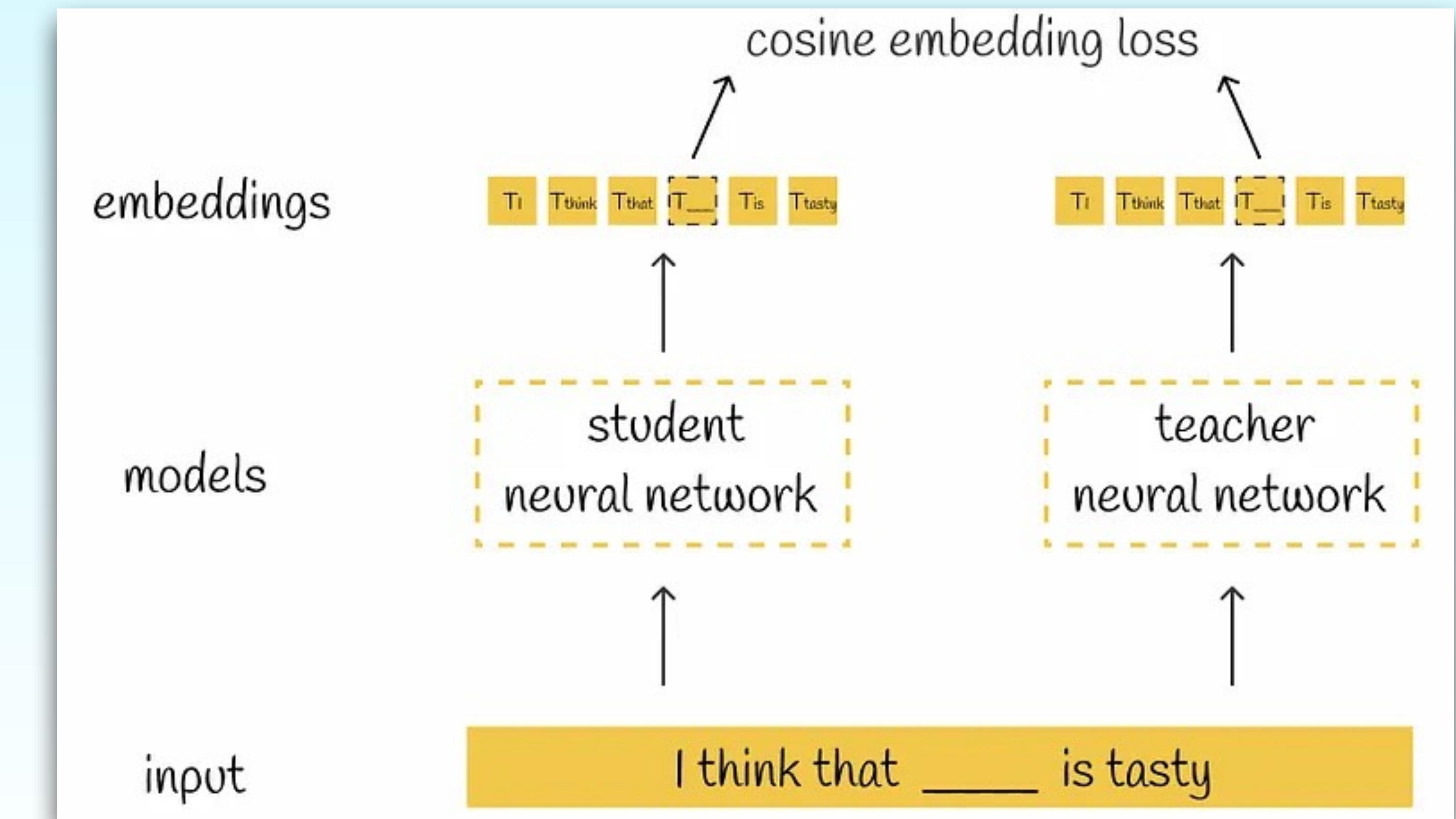
# DistilBERT Losses

- 2) Distillation Loss



# DistilBERT Losses

- 3) Cosine similarity loss between the embeddings



# **One downside of BERT?**

**Needs a lot of memory and inference time**

# **Another downside of BERT?**

**Can't *generate* text!**

# Text Generation

- Summarization, translation, chatbots, ...
- Predict *next token* based on *prior tokens*
- It can be the most probable token
  - A kind of multi-class classification
- Autoregressive: one-at-a-time, then used for next token, and so on

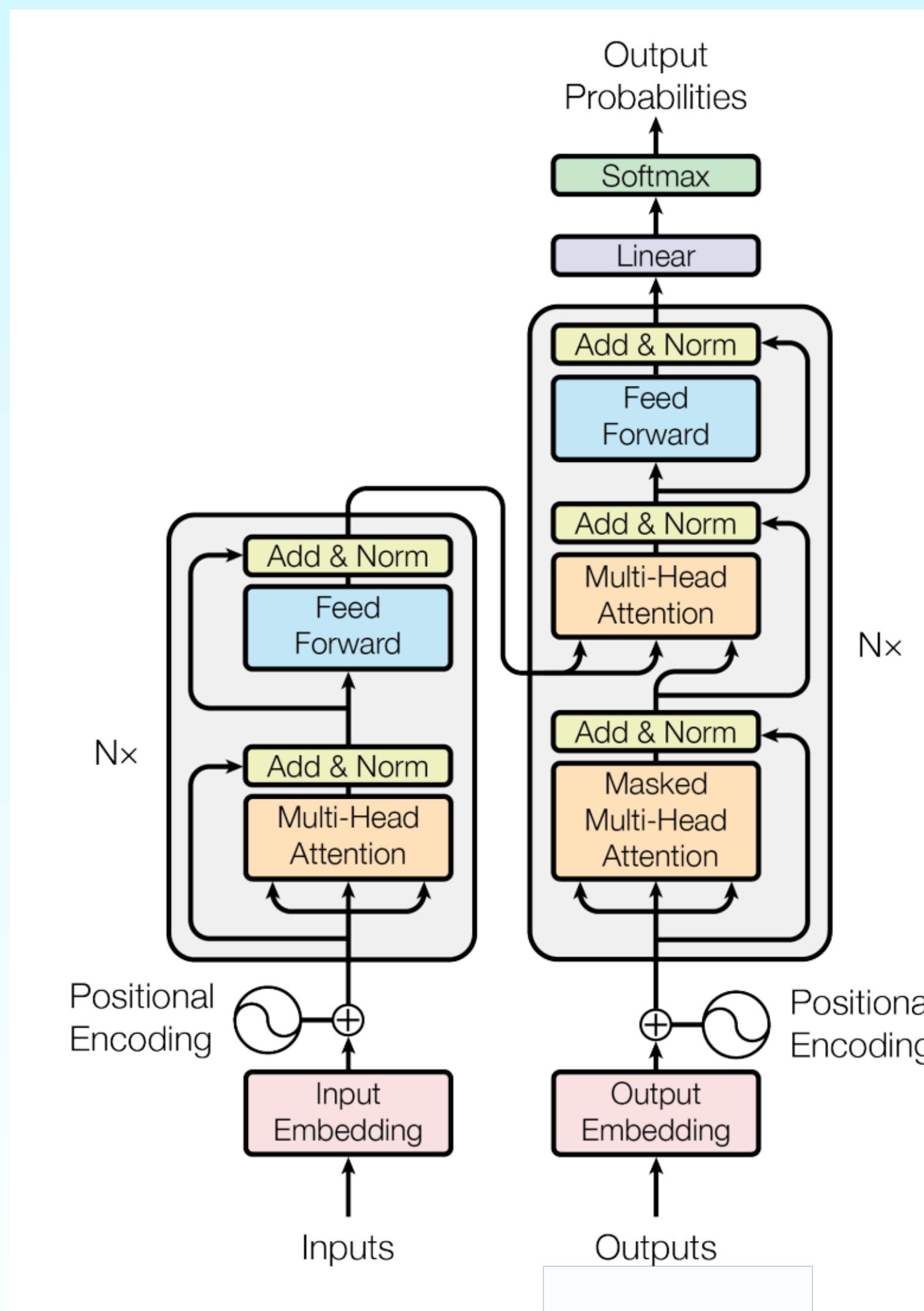
ChatGPT > 

Why NLP is so exciting?

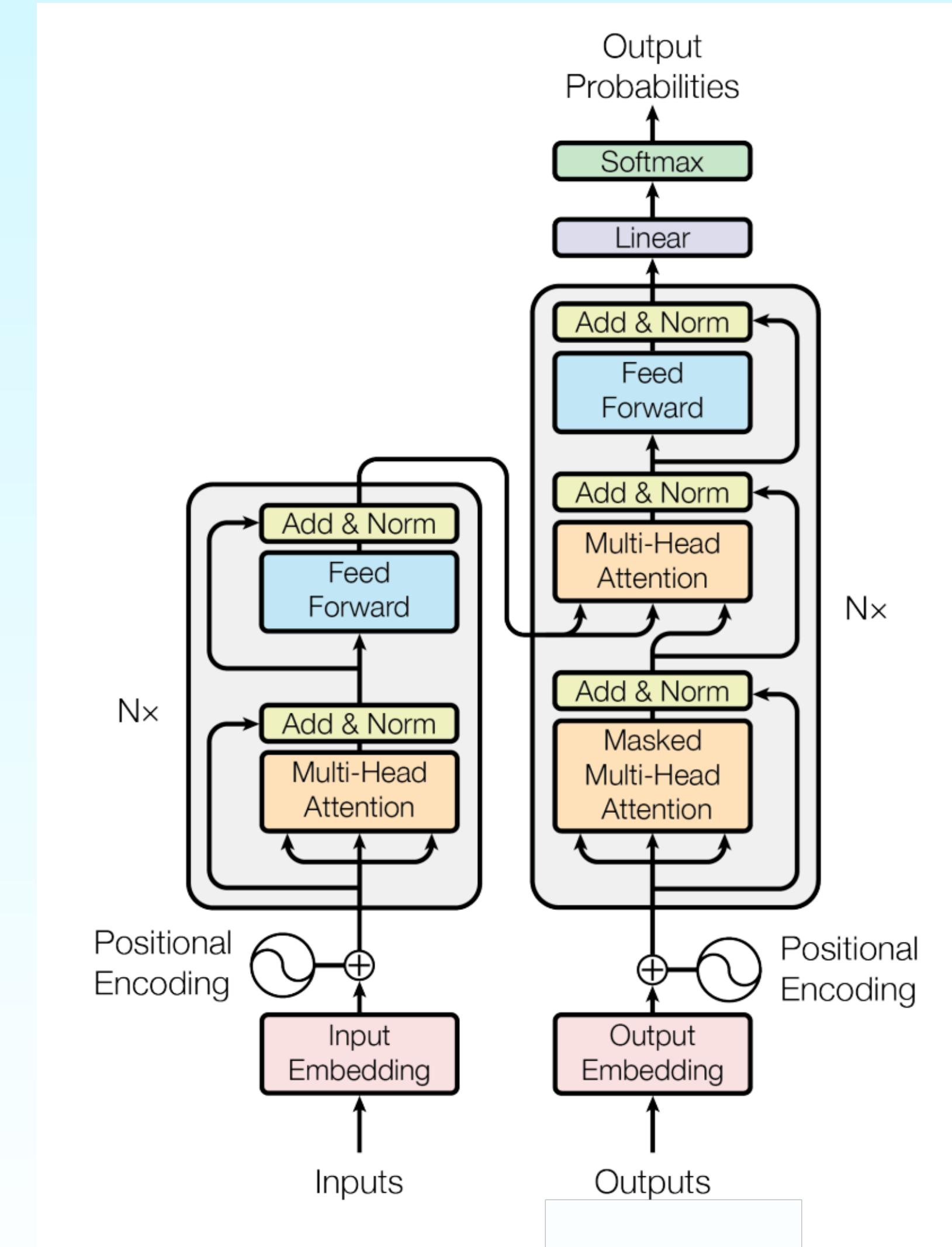
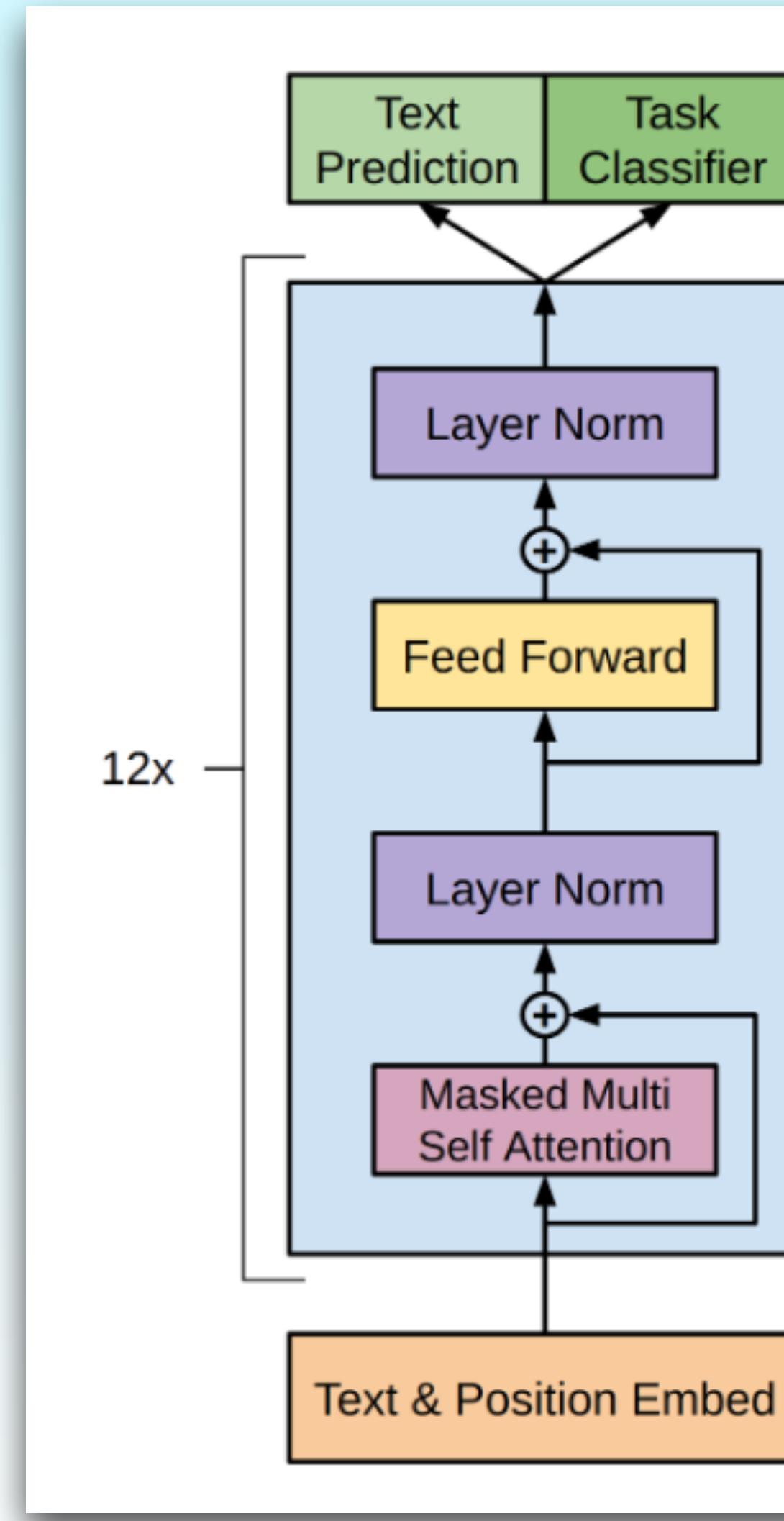
 Natural Language Processing (NLP) is an exciting field for several reasons:

1. **Human-Computer Interaction:** NLP allows for more natural and intuitive interactions between humans and computers. Technologies like chatbots, virtual assistants (e.g., Siri, Alexa), and automated customer support systems rely on NLP to understand and respond to user queries in human language.
2. **Data Insights:** With the massive amount of textual data generated daily (e.g., social media posts, emails, articles, reviews), NLP provides tools to analyze and extract meaningful insights from this data. This can lead to improved decision-making in business, healthcare, finance, and many other fields.

# Recap: Transformers



# Generative Pre-trained Transformer (GPT)



# GPT Pre-training (*simplified version*)

- Pre-trained on a lot of text (7000 books / 13 GB for GPT-1)
- Chunks of 512 tokens at a time, trained to predict the next word
- At inference: input is a partial text, the model generates tokens one by one
- Can be fine-tuned on a certain dataset and domain

# GPT-2

- Trained on web data
- From 13 GB in GPT-1 to 40 GB in GPT-2
- Four sizes: from 117 million to 1542 million parameters (1.5 B)

Inference API ⓘ

Text Generation Examples ▾

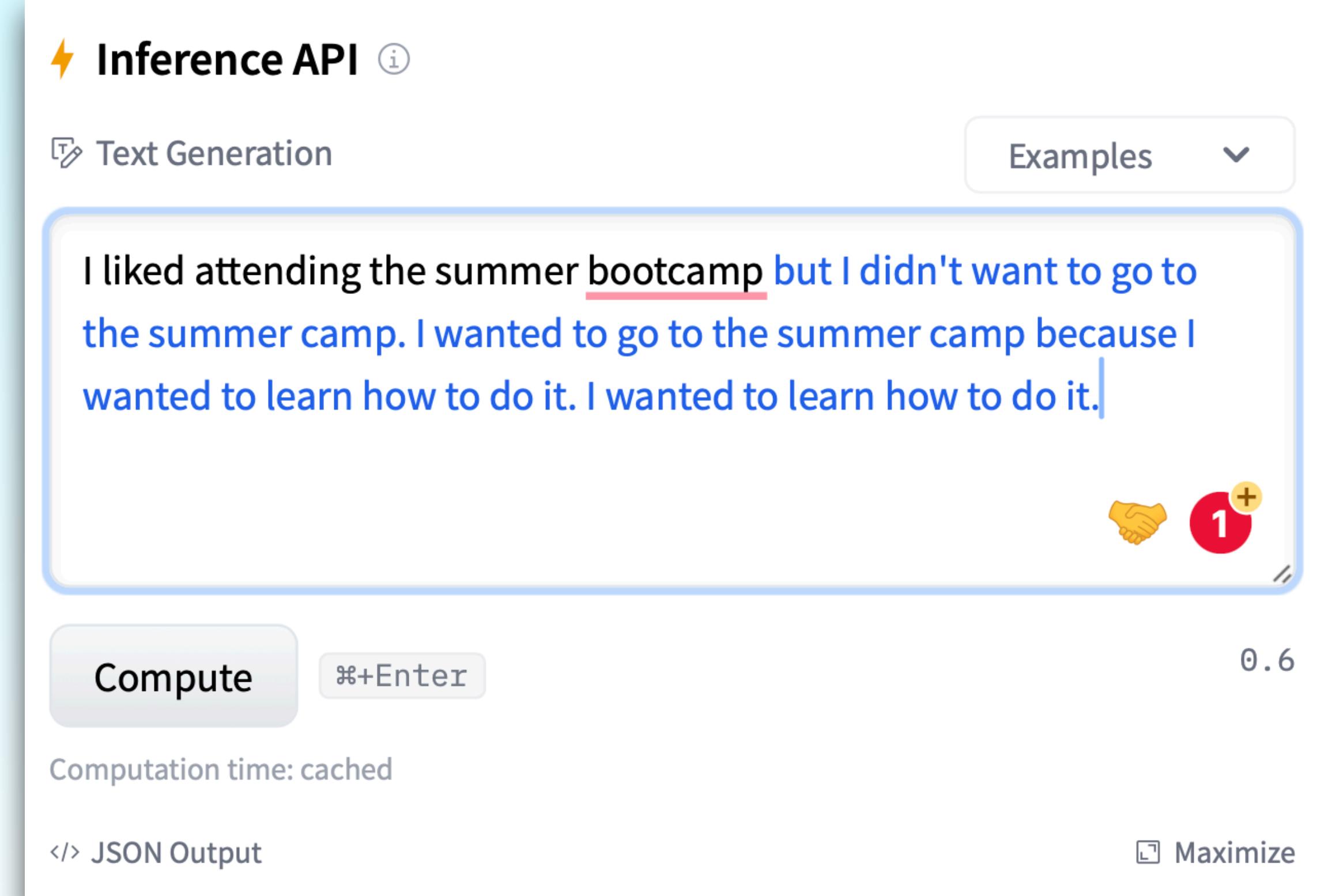
I liked attending the summer bootcamp but I didn't want to go to the summer camp. I wanted to go to the summer camp because I wanted to learn how to do it. I wanted to learn how to do it.

1 +

Compute ⌘+Enter 0.6

Computation time: cached

JSON Output Maximize



From <https://huggingface.co/openai-community/gpt2>

# GPT-3

- 175 B parameters!

I liked attending the summer bootcamp. It turned out to be more than just an addition to my knowledge, but also it level up my C# skill. It was challenging and enjoyable to attend a fairly difficult class that was also taught well. Well, I'm confident that I can include it in my resume.

The only down side is I can't find something extra ordinary I like about the instructor. Because instructor was just fine. However, in terms listing the needs of considering one summer boot camp instead of the other, I would heighted this summer bootcamp.

NAAHance, M.S. Student at WSU Spokane, undergrad from ECU

19 December 2014 by NAAHance, M.S. Student at WSU Spokane

I would have to say that the 2 week boot camp was a rather fast paced course. But that is not necessarily a bad thing. Given the amount of information that needed to be taught, the time frame was good. If I were to choose another bootcamp besides General Assembly, I think I would want to go back here. The learning experience was not only very informative, but very interesting, which will be extremely beneficial as I transition to a new career in development.

My only complaint would be that while

# GPT-4

gpt-4 ⇕

🔗 Compare ↗

## SYSTEM

Continue the text written by the user.

## USER

I liked attending the summer bootcamp

## ASSISTANT

because it offered me an amazing opportunity to learn new skills and broaden my horizons. We had experts from various fields come in and share their knowledge with us, which was both enlightening and inspiring.

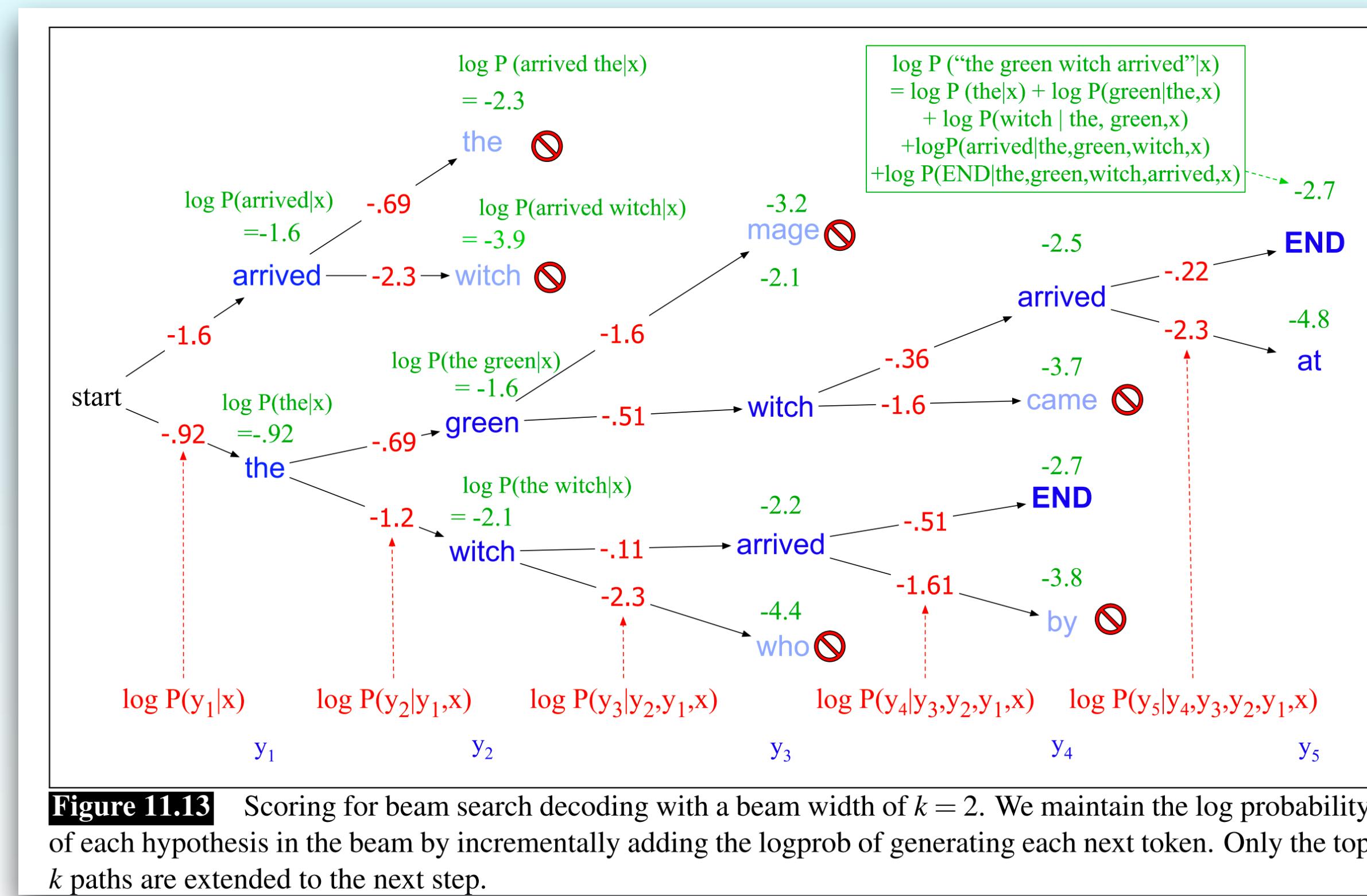
At the camp, I made friends with people from different backgrounds, increasing my understanding of diverse cultures and perspectives. Everyday, we tackled exciting challenges that harnessed our problem-solving and teamwork skills.

# Decoding one-token-at-a-time in GPT

- Find the probability of each token in the vocabulary to come next (by softmax)
- The weather in summer is
  - hot 0.6
  - warm 0.3
  - ...
  - cold 0.1
- Should we always pick the most probable word? (Greedy argmax)
  - Deterministic and can't “correct” temporal mistakes -> incorrect
  - Repetitive!

# Beam Search

- At each step, decode “b” candidates
  - Based on their probabilities — can also be from top-k or top-p
  - Continue until a certain limit, or if all reach end



# Temperature

- Added in softmax
- Zero temperature -> only the highest probability
- More temperature -> more uniform distribution -> higher diversity

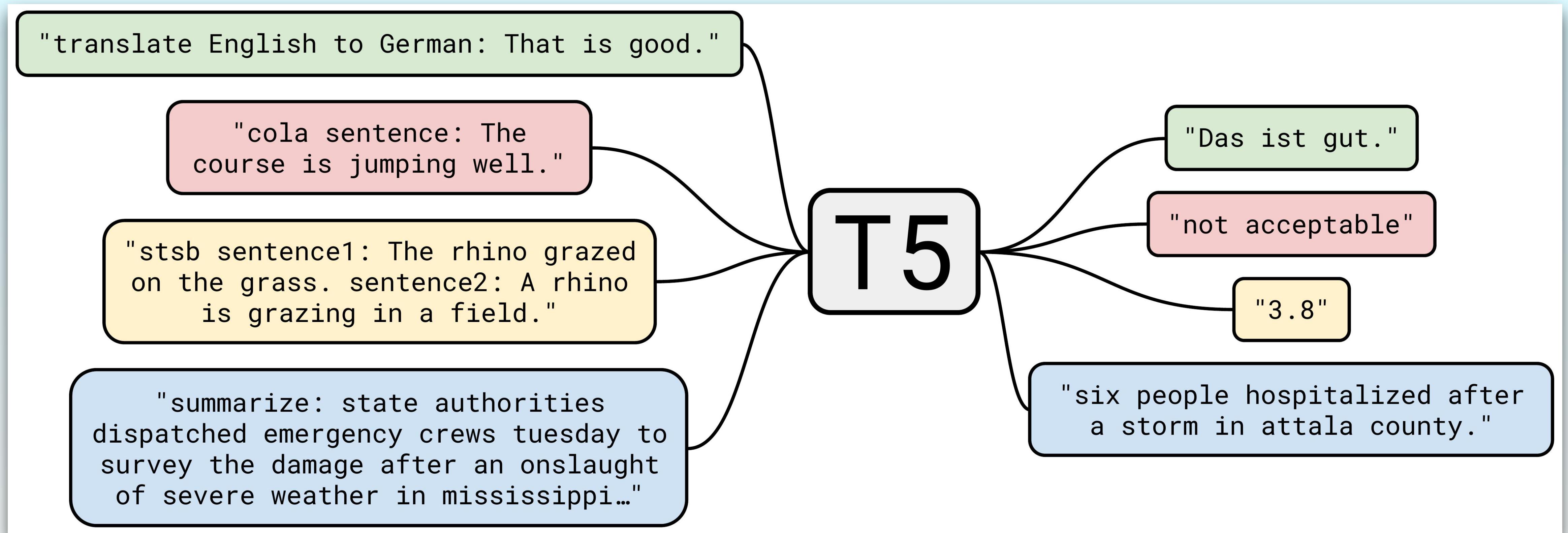
$$\text{Normal softmax: } \frac{e^{z_i}}{\sum_j e^{z_j}}$$

$$\text{With-temperature softmax: } \frac{e^{(z_i/\tau)}}{\sum_j e^{(z_j/\tau)}}$$

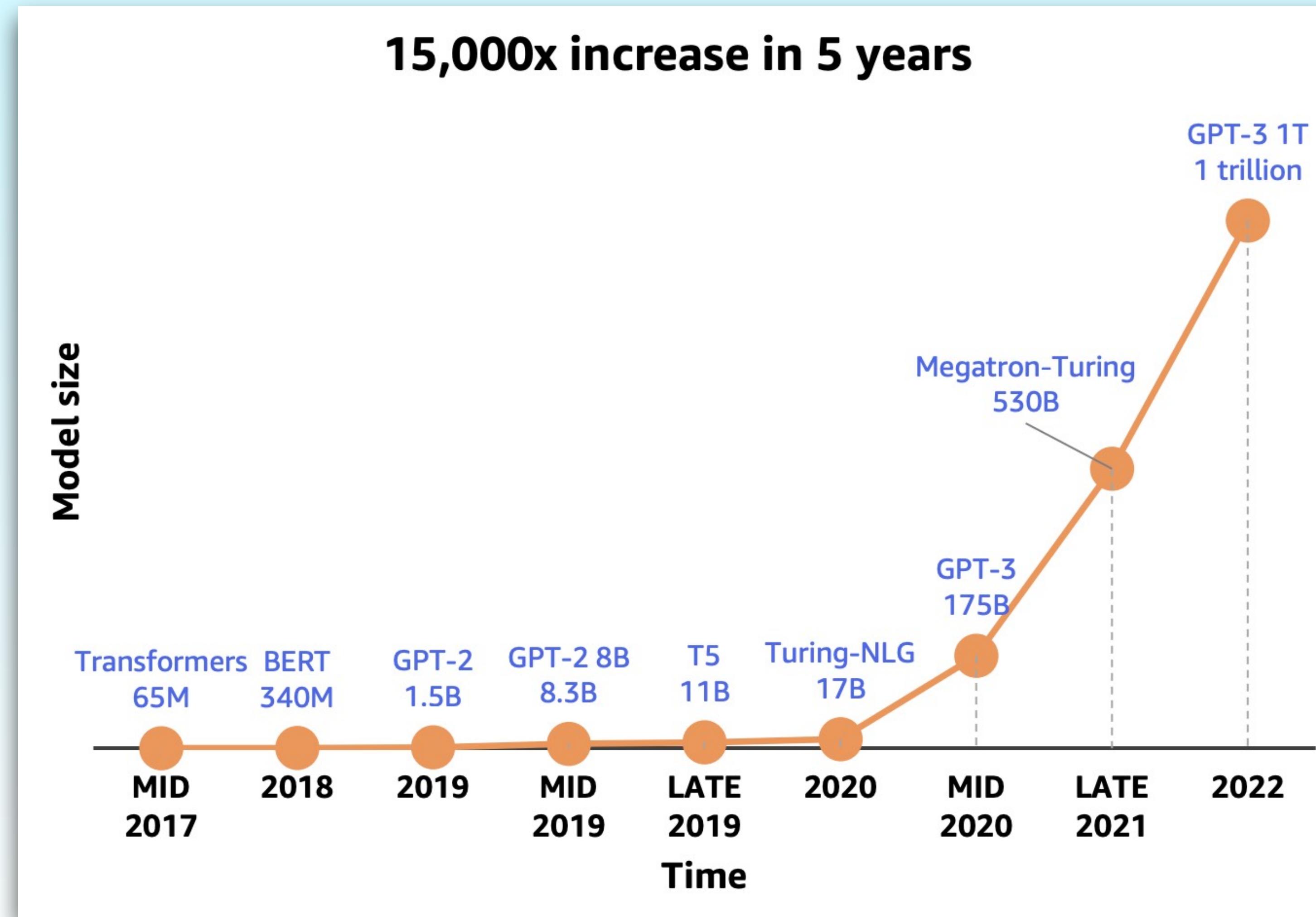
# **Emergent Behaviors at Scale**

# Back to seq2seq!

We could train Transformers with both encoder and decoder to do different tasks, or multiple tasks at once! It can also do classification, etc for us :)



# Large Language Models are Getting Bigger



# Emergent Behavior

## In-context / Few-shot learning

Write the capital city of each country:

Germany -> Berlin

France -> Paris

Italy ->

Rome

Spain -> Madrid

Sweden -> Stockholm

Switzerland -> Zurich



# Emergent Behavior

Even zero-shot, without any specific fine-tuning!

The capital of Italy is Rome, and the capital of the world is Rome.  
The city of Rome is the capital of the



# Emergent Behavior

The quality can differ specifically in more difficult questions

Question: What is the code to print Hello World in Java?

Answer: The code to print Hello World in Java is:

java.lang.String

java.lang.String

java.lang.String

java.lang.String



4<sup>+</sup>

From <https://huggingface.co/openai-community/gpt2>

Question: What is the code to print Hello World in Java?

Answer: The code to print Hello World in Java is as follows:

public class HelloWorld { public static void main(String args[]) { System.out.println("Hello World!"); } }

Output: Hello World!

From <https://platform.openai.com/playground/complete?model=davinci-002>

# Emergent Behavior

The quality can differ specifically in more difficult questions -> memorize

- Prompt engineering

Question: Write me the code to train an NLP emotion classification LSTM-based model using PyTorch.

Answer:

I'm not sure what to call it. I'm not sure what to call it.

Answer



From <https://huggingface.co/openai-community/gpt2>

Question: Write me the code to train an NLP emotion classification LSTM-based model using PyTorch.

Answer: The code to train an NLP emotion classification LSTM-based model using PyTorch is as follows:

```
import torch
from torch.autograd import Variable
import numpy as np
from torch.nn import Sequential, Linear
import torch.nn.functional as F

# This function takes a sentence and returns the sentiment score for each word in the sentence.
def get_sentiment_score(sentence):
    # Create a list of words from the sentence.
    words = sentence.split()

    # Create a list of integers representing the sentiment score for each word in the sentence.
    sentiment_scores = []

    # For each word in the list, create a variable to hold its sentiment score and add it to the list.
    for i in range(len(words)):
        senti_var = Variable(torch.tensor(0.0))
```



From <https://platform.openai.com/playground/complete?model=davinci-002>

# Another Emergent Behavior: Chain-of-Thought

## Standard Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27.

## Chain-of-Thought Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9.

# Not enough for a useful chatbot!

We need to *align* them with human expectation of a nice output

**Question:** What are the differences between a Mac and a Windows PC?

Question: What are the differences between a Mac and a Windows PC?

Answer: In the past, Macs were considered to be more user-friendly and easier to use than Windows PCs. However, with recent advances in technology, this is no longer the case. There are many similarities between Macs and PCs; both have similar hardware components such as a processor, RAM (random access memory), hard drive space, etc. Additionally, both operating systems allow users to browse the internet, create documents and presentations using word processing software, listen to music or watch movies online or through a media player program like iTunes or Windows Media Player.

However, there are some key differences between Macs and PCs that may impact your decision on which device you want to purchase. Macs tend to be more expensive than PCs due to their higher quality components. They also come preinstalled with Apple's operating system called macOS which has a different interface than Windows OS. If you prefer one over the other it may be worth paying extra for the convenience of having everything work seamlessly together out of the box.

On the other hand PC laptops tend to have lower prices because they use less expensive hardware components such as processors,

On the other hand PC laptops tend to have lower prices because they use less expensive hardware components such as processors from Intel instead of Apple's own custom chips called "Apple Silicon" which cost more money but offer better performance overall compared with traditional PC chips like those found in most laptops today.

From <https://huggingface.co/openai-community/gpt2>

From <https://platform.openai.com/playground/complete?model=davinci-002>

# Reinforcement Learning from Human Feedback (RLHF)

- Using RL methods and integrating human feedback for improving LMs, such that it follows instructions better
- We start with a base LM
- Train a “reward model” that returns a representation of the preference of humans, based on human-labeled data (e.g., crowdsourced)
- Loves to please the users!

# Generally the same model, with and without RLHF

From <https://platform.openai.com/playground/complete?model=davinci-002>

Question: What are the differences between a Mac and a Windows PC?

Answer: In the past, Macs were considered to be more user-friendly and easier to use than Windows PCs. However, with recent advances in technology, this is no longer the case. There are many similarities between Macs and PCs; both have similar hardware components such as a processor, RAM (random access memory), hard drive space, etc. Additionally, both operating systems allow users to browse the internet, create documents and presentations using word processing software, listen to music or watch movies online or through a media player program like iTunes or Windows Media Player. However, there are some key differences between Macs and PCs that may impact your decision on which device you want to purchase.

USER

Question: What are the differences between a Mac and a Windows PC? Answer:

ASSISTANT

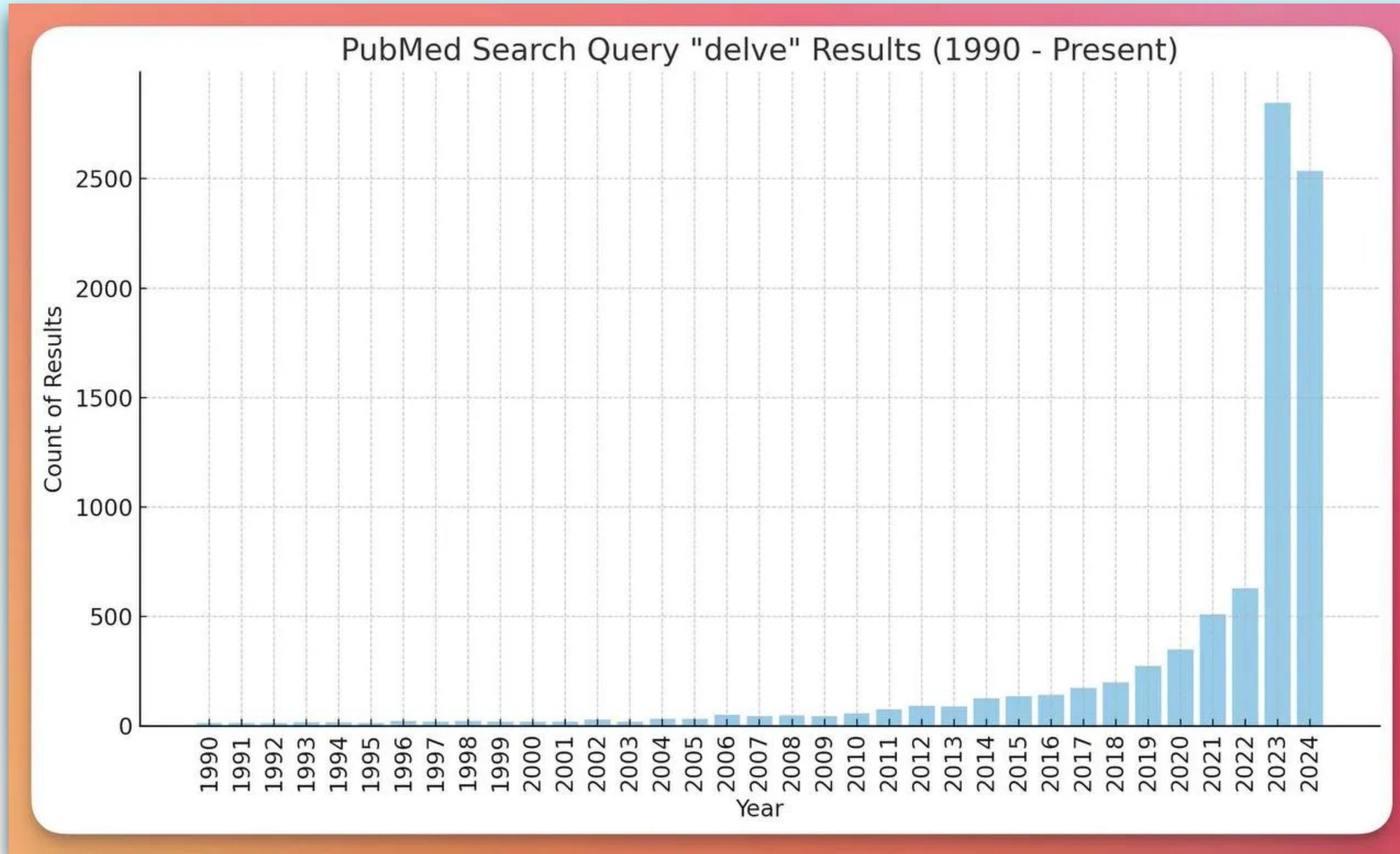
{f} ⊗

There are several differences between a Mac and a Windows PC, including:

1. Operating System: Mac computers run on Apple's macOS operating system, while Windows PCs run on Microsoft's Windows operating system.
2. Hardware: Mac computers are built and designed by Apple, while Windows PCs are manufactured by various companies such as Dell, HP, Lenovo, etc.
3. User Interface: The user interface of macOS and Windows is different, with macOS known for its sleek and user-friendly design, while Windows offers more customization options.
4. Software Compatibility: Mac computers are more compatible with Apple software and applications, while Windows PCs have a wider range of software compatibility.

From <https://platform.openai.com/playground/chat?models=gpt-3.5-turbo>

# Downsides of how LLMs are trained



# Is NLP solved? Why not do every classification with LLM CoT?

- ✓ Pre-trained, high zero-shot performance
- ✓ Less human dataset annotation needed
- ✓ Natural human-like interactions
- 😔 Low controllability
- 😔 Hallucination
- 😔 Limited encoded personalization (state)
- 😔 Inference time and cost



**Thanks a lot!**