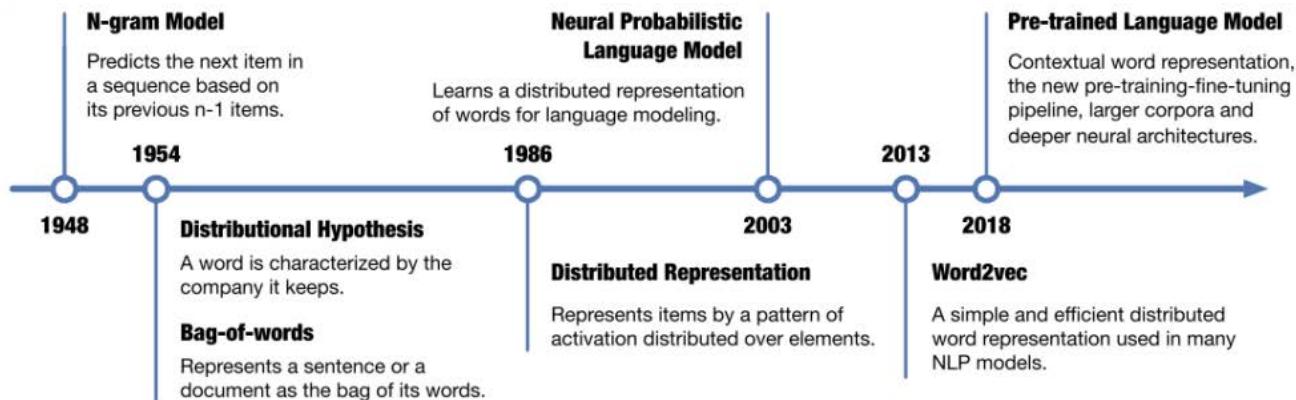


What are Large Language models (LLMs)?

# Background

- language model



# What is language modeling?

A **language model** assigns a probability to a N-gram

$$f: V^n \rightarrow R^+$$

# What is language modeling?

A **language model** assigns a probability to a N-gram

$$f: V^n \rightarrow R^+$$



*Sfkllkjf fskjhfkjsh kjfs fs kjhkjhs fsjhfkshkjfh*

**Low** probability



*ChatGPT is all you need*

**high** probability

# What is language modeling?

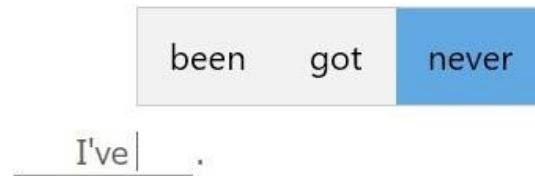
A **language model** assigns a probability to a N-gram

$$f: V^n \rightarrow R^+$$

A **conditional language model** assigns a probability of a word given some conditioning context

$$g: (V^{n-1}, V) \rightarrow R^+$$

And  $p(w_n | w_1 \dots w_{n-1}) = g(w_1 \dots w_{n-1}, w) = \frac{f(w_1 \dots w_n)}{f(w_1 \dots w_{n-1})}$



# What is language modeling?

A **language model** assigns a probability to a N-gram

$$f: V^n \rightarrow R^+$$

A **conditional language model** assigns a probability of a word given some conditioning context

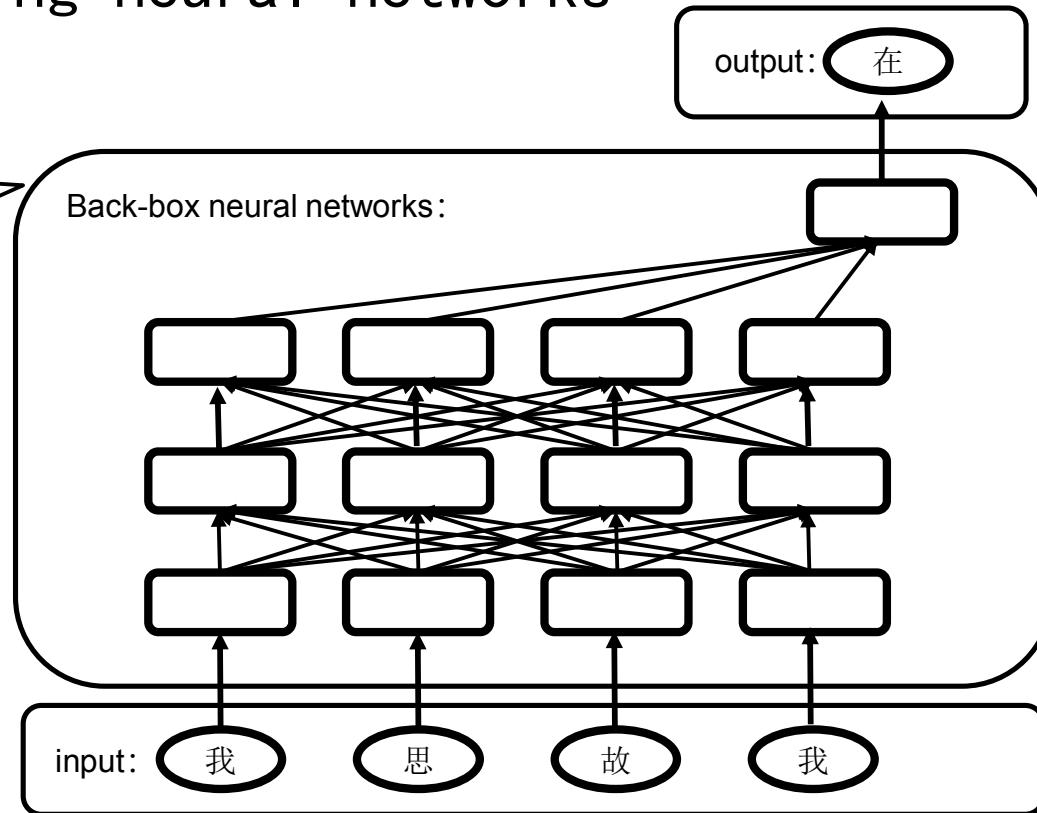
$$g: (V^{n-1}, V) \rightarrow R^+$$

And  $p(w_n | w_1 \dots w_{n-1}) = g(w_1 \dots w_{n-1}, w) = \frac{f(w_1 \dots w_n)}{f(w_1 \dots w_{n-1})}$

$p(w_n | w_1 \dots w_{n-1})$  is the foundation of **modern large language models** (GPT, ChatGPT, etc.)

# Language model using neural networks

GPT-3/ChatGPT/GPT4 have  
175B+ parameters  
Humans have 100B+  
neurons



# Language models: Narrow Sense

A probabilistic model that assigns a probability to every finite sequence (grammatical or not)

Sentence: “the cat sat on the mat”

$$\begin{aligned} P(\text{the cat sat on the mat}) &= P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat}) \\ &\quad * P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on}) \\ &\quad * P(\text{mat}|\text{the cat sat on the}) \end{aligned}$$

Implicit order

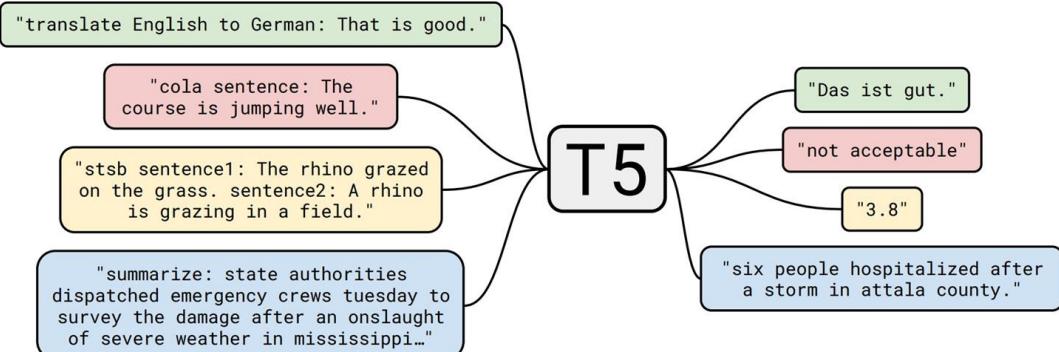
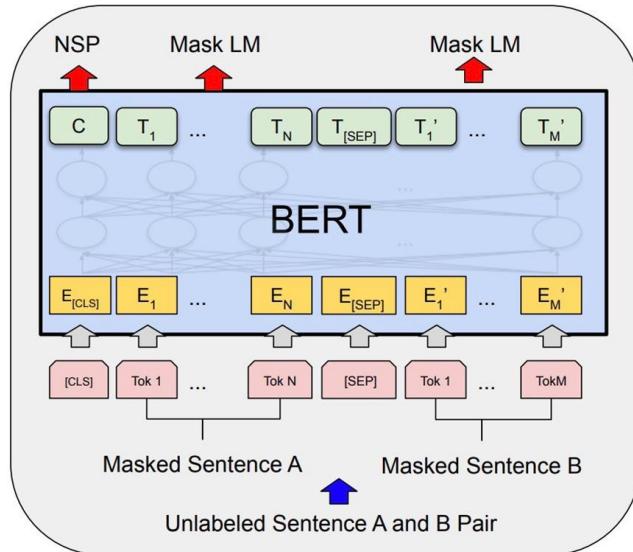


GPT-3 still acts in this way but the model is implemented as a very large neural network of 175-billion parameters!

# Language models:Broad Sense

- ❖ Decoder-only models (GPT-x models)
- ❖ Encoder-only models (BERT, RoBERTa, ELECTRA)
- ❖ Encoder-decoder models (T5, BART)

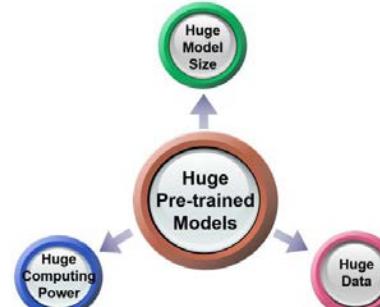
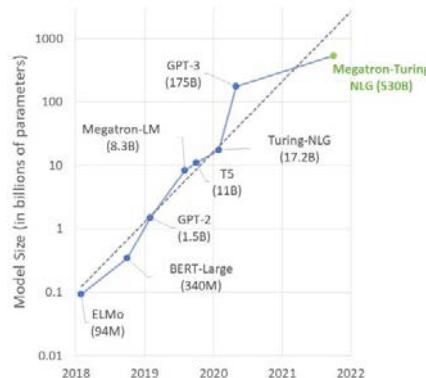
The latter two usually involve a different **pre-training** objective.



# PLM vs. LLM

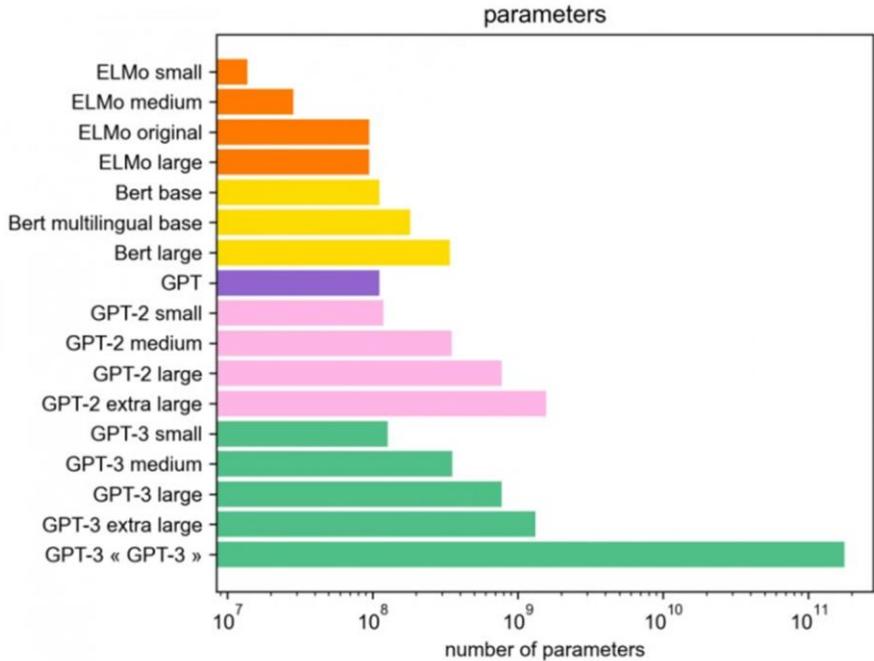
We do not explicitly mention pre-training because pre-training and training use the same language models objective (e.g., autoregressive generation)

- Pre-trained language model
- Large pre-trained Language Model (LLM)

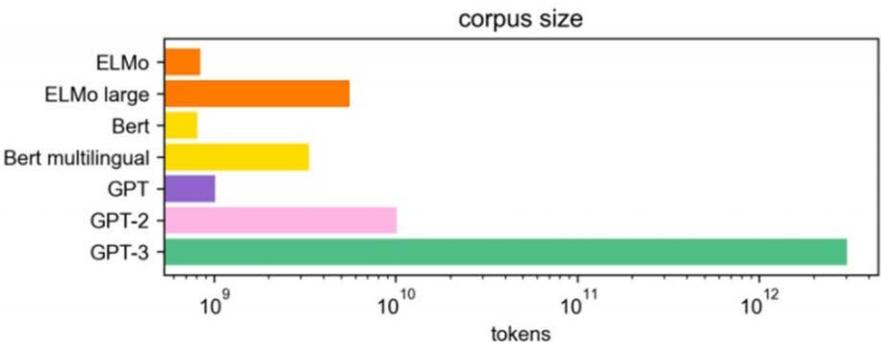


# How Large are “Large” LMs?

models



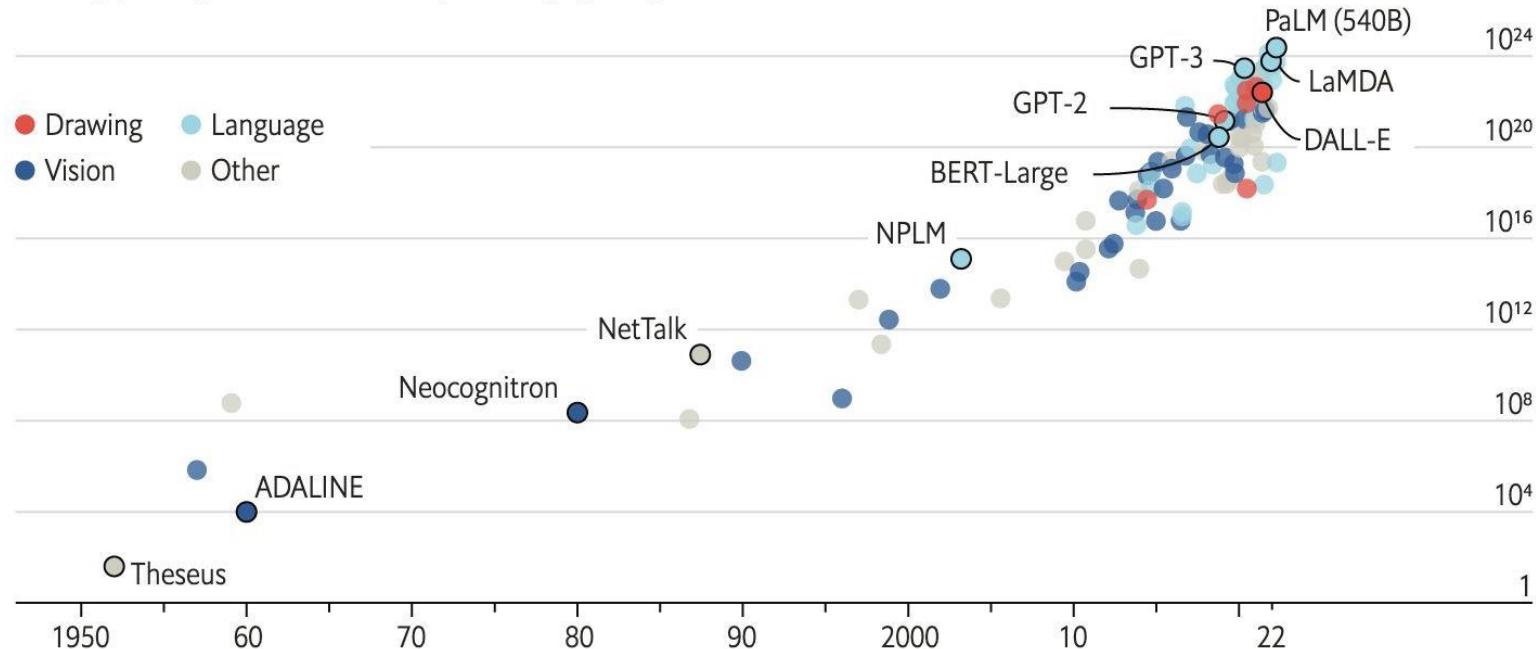
models



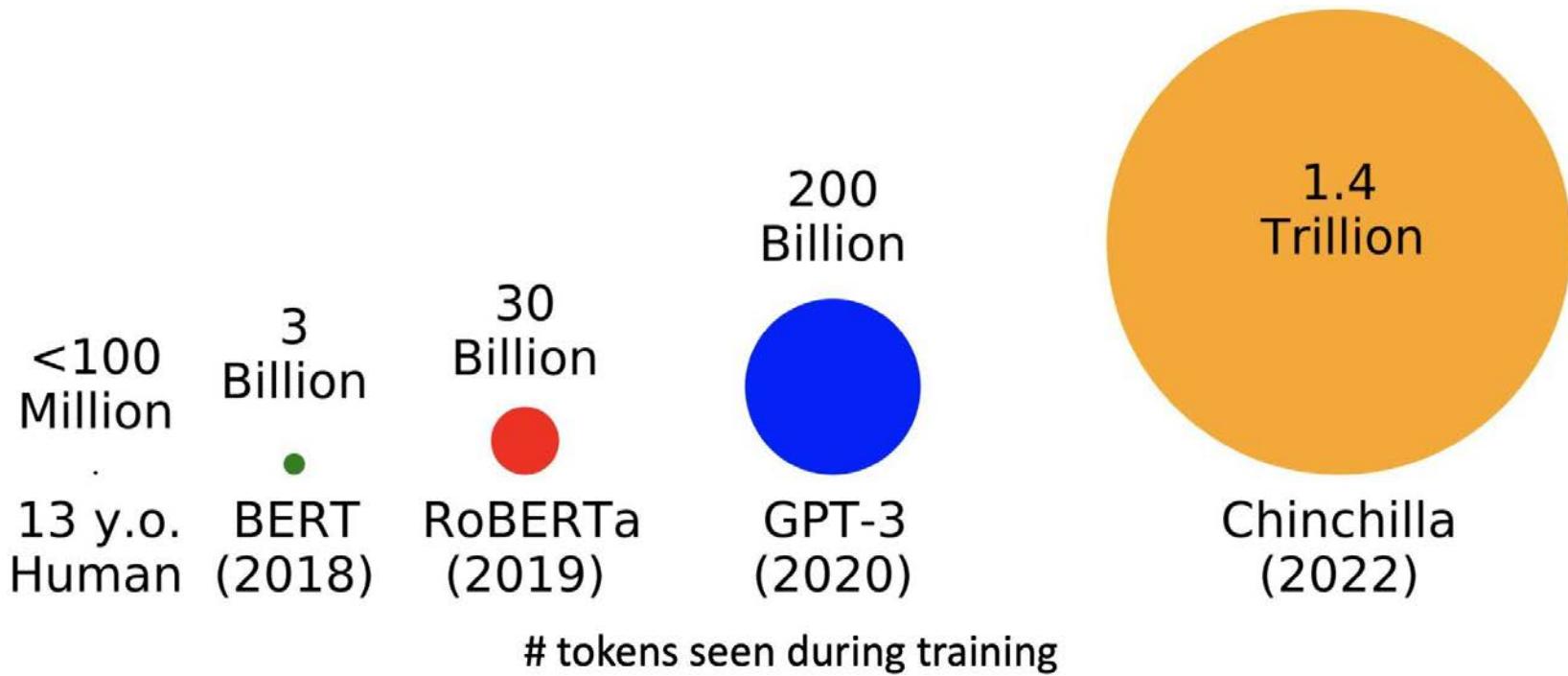
More recent models: PaLM (540B), OPT (175B), BLOOM (176B)...

# Large Language Models

AI training runs, estimated computing resources used  
Floating-point operations, selected systems, by type, log scale

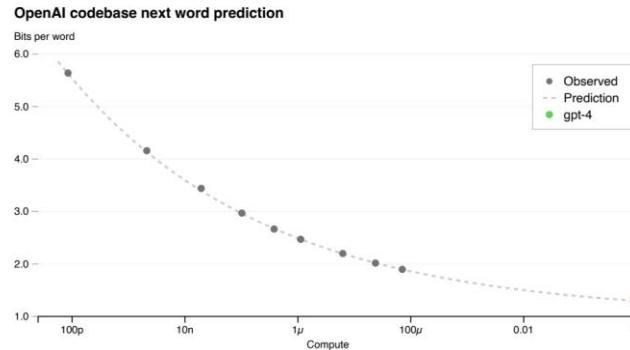


# Large Language Models - **Hundreds of Billions of Tokens**



# Some basics for large language models

- Scalable network **architecture** (Transformer vs. CNN/RNN)
- Scalable **objective** (conditional/auto-regressive LM vs. Masked LM)



- Scalable **data** (plain texts are everywhere vs. supervised data)
  - <https://github.com/esbatmop/MNBVC>

# How Large are “Large” LMs?

- ❖ Today, we mostly talk about two camps of models:
  - Medium-sized models: BERT/RoBERTa models (100M or 300M), T5 models (220M, 770M, 3B)
  - “Very” large LMs: models of 100+ billion parameters
- ❖ Larger model sizes              larger compute, more expensive during inference
- ❖ Different sizes of LMs have different ways to adapt and use them
  - Fine-tuning, zero-shot/few-shot prompting, in-context learning...
- ❖ Emergent properties arise from model scale
- ❖ Trade-off between model size and corpus size

Why LLMs?

# Why Larger language models

- More world **knowledge** (LAMA)
  - Language models as knowledge base?
- Larger capacity to learn problem-solving **Abilities**
  - Coding, revising articles, reasoning etc.
- Better **generalization** to unseen tasks
- **Emergent ability** (涌现能力)

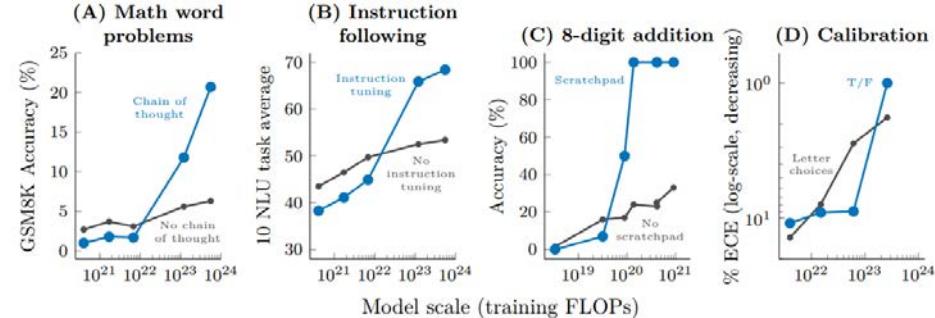
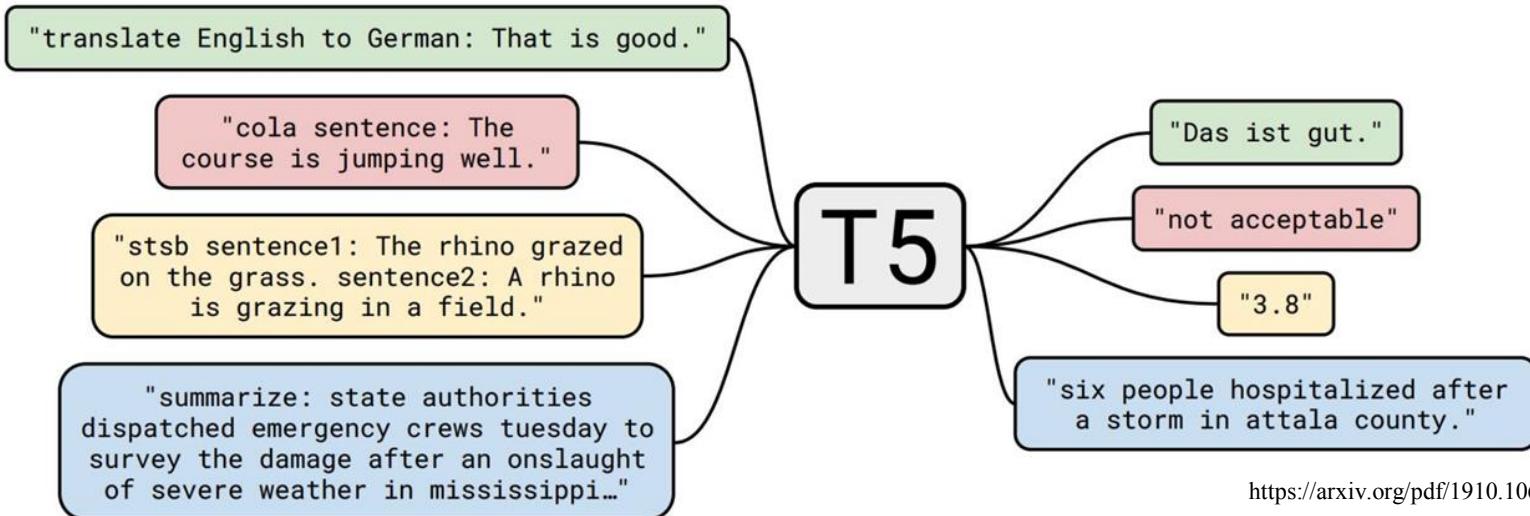


Figure 3: Specialized prompting or finetuning methods can be emergent in that they do not have a positive effect until a certain model scale. A: Wei et al. (2022b). B: Wei et al. (2022a). C: Nye et al. (2021). D: Kadavath et al. (2022). An analogous figure with number of parameters on the x-axis instead of training FLOPs is given in Figure 12. The model shown in A-C is LaMDA (Thoppilan et al., 2022), and the model shown in D is from Anthropic.

# Why LLMs?

## Generalization :

One single model to solve many NLP tasks



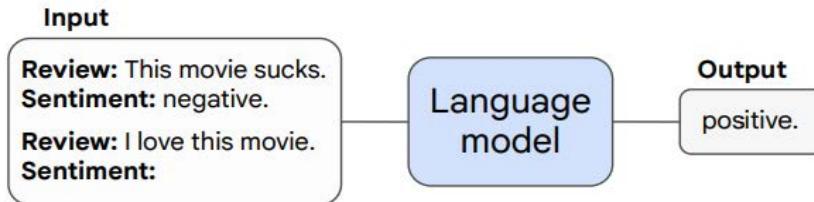
It could even generalize to new tasks, following the phylosity of FLAN

# Why LLMs?

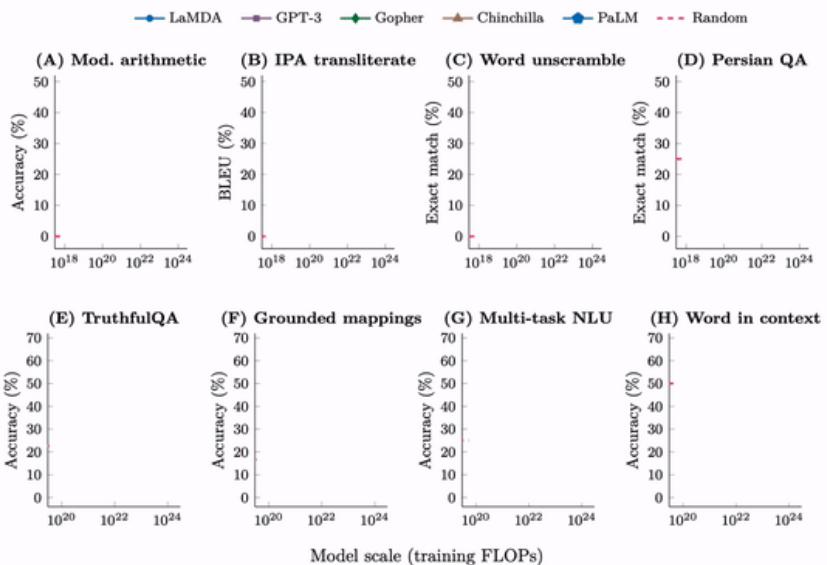
## Emergent properties in LLMs:

Some ability of LM is not present in smaller models but is present in larger models

## Emergent Capability: Few-shot prompting



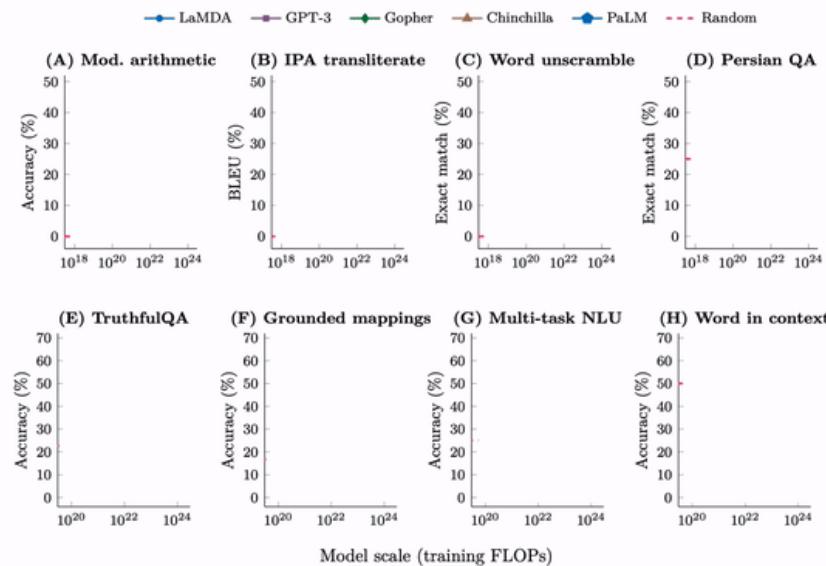
> A few-shot prompted task is emergent if it achieves random accuracy for small models and above-random accuracy for large models.



# Why LLMs?

## ● Emergent Abilities

- Some ability of LM is not present in smaller models but is present in larger models



[https://docs.google.com/presentation/d/1yzbmYB5E7G8lY2-KzhmArmPYwwl7o7CUST1xRZDUu1Y/edit?resourcekey=0-6\\_TnUMoKWCK\\_FN2BiPxmbw#slide=id.g1fc34b3ac18\\_0\\_27](https://docs.google.com/presentation/d/1yzbmYB5E7G8lY2-KzhmArmPYwwl7o7CUST1xRZDUu1Y/edit?resourcekey=0-6_TnUMoKWCK_FN2BiPxmbw#slide=id.g1fc34b3ac18_0_27)

# Emergent Capability - In-Context Learning

## Traditional fine-tuning (not used for GPT-3)

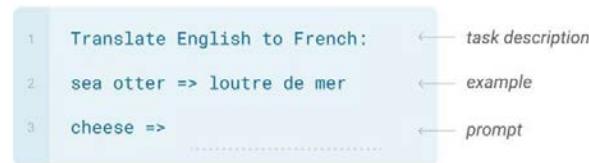
### Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



### One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



### Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

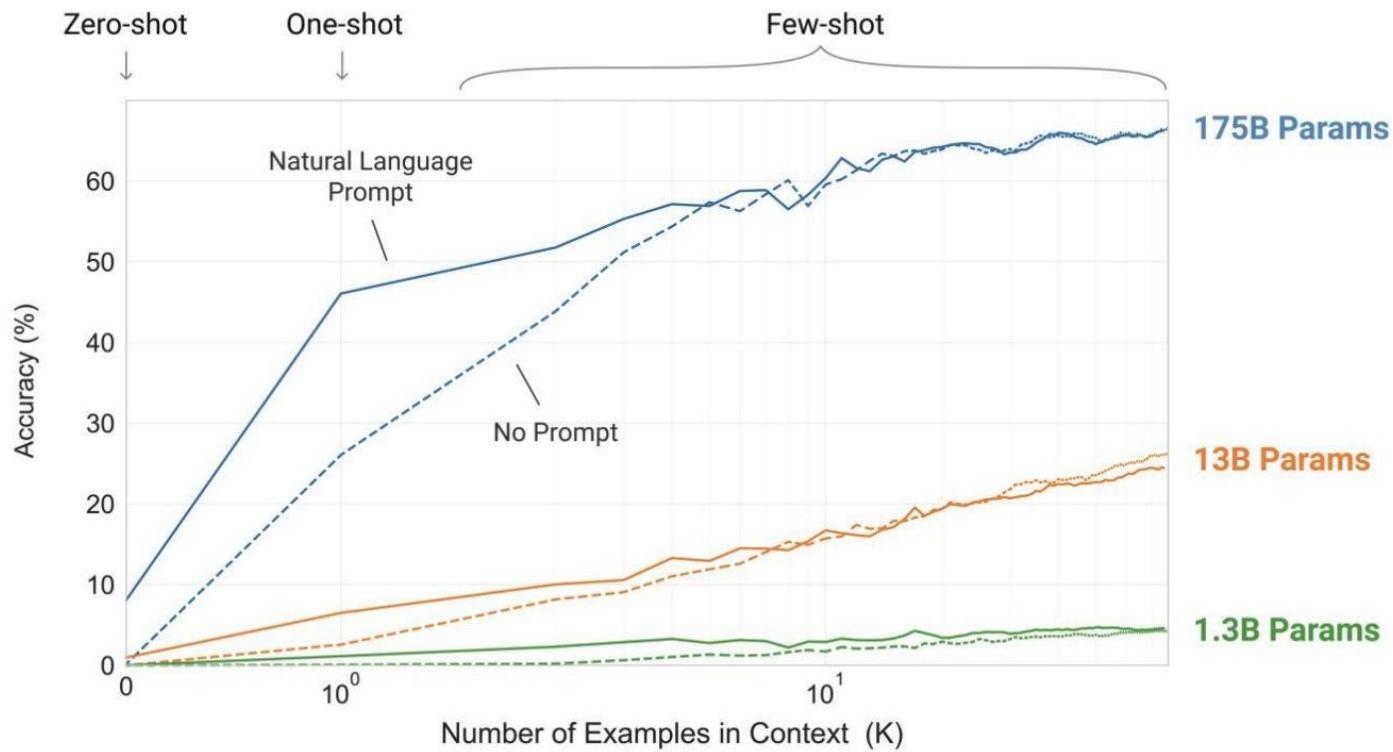


<https://arxiv.org/pdf/2005.14165.pdf>

# Emergent Capability - In-Context Learning

	No Prompt	Prompt
Zero-shot (os)	skicts = sticks	Please unscramble the letters into a word, and write that word: skicts = sticks
1-shot (1s)	chiar = chair skicts = sticks	Please unscramble the letters into a word, and write that word: chiar = chair skicts = sticks
Few-shot (FS)	chiar = chair [...] pciinc = picnic skicts = sticks	Please unscramble the letters into a word, and write that word: chiar = chair [...] pciinc = picnic skicts = sticks

# Emergent Capability - In-Context Learning



# Emergent Capability - Chain of Thoughts Prompting

## 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. X

## 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. ✓

# Emergent Capability - Chain of Thoughts Prompting

**Math Word Problems (free response)**

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.

**Math Word Problems (multiple choice)**

Q: How many keystrokes are needed to type the numbers from 1 to 500?  
Answer Choices: (a) 1156 (b) 1392 (c) 1480 (d) 1562 (e) 1788

A: There are 9 one-digit numbers from 1 to 9. There are 90 two-digit numbers from 10 to 99. There are 401 three-digit numbers from 100 to 500.  $9 + 90(2) + 401(3) = 1392$ . The answer is (b).

**CSQA (commonsense)**

Q: Sammy wanted to go to where the people were. Where might he go?  
Options: (a) race track (b) populated areas (c) desert (d) apartment (e) roadblock

A: The answer must be a place with a lot of people. Race tracks, desert, apartments, and roadblocks don't have a lot of people, but populated areas do. So the answer is (b).

**StrategyQA**

Q: Yes or no: Would a pear sink in water?

A: The density of a pear is about 0.6 g/cm<sup>3</sup>, which is less than water. Thus, a pear would float. So the answer is no.

**Date Understanding**

Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?

A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.

**Sports Understanding**

Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."

A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.

**SayCan (Instructing a robot)**

Human: How would you bring me something that isn't a fruit?

Explanation: the user wants something to eat that isn't a fruit. An energy bar is not a fruit, so I will bring the user an energy bar.  
Plan: 1. find(energy bar).  
put(energy bar) 3. find(user).  
put(energy bar) 4. done().

**Last Letter Concatenation**

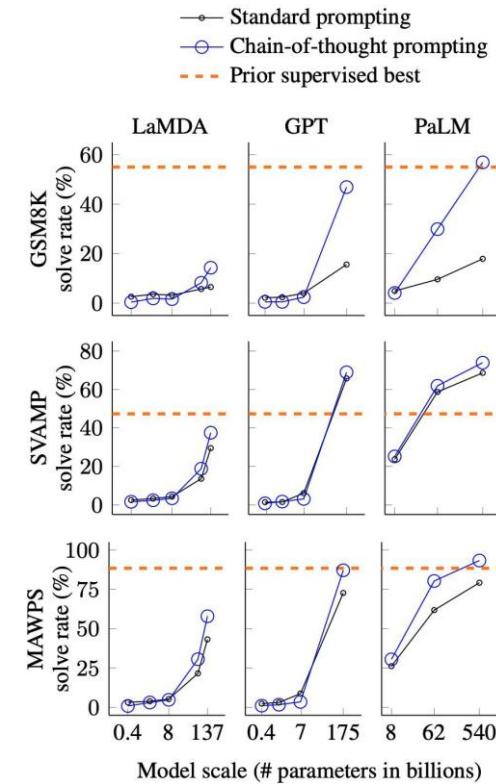
Q: Take the last letters of the words in "Lady Gaga" and concatenate them.

A: The last letter of "Lady" is "y". The last letter of "Gaga" is "a". Concatenating them is "ya". So the answer is ya.

**Coin Flip (state tracking)**

Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.



# Emergent Capability – Augmented Prompting Abilities

## Advanced Prompting Techniques

- Zero-shot CoT Prompting
- Self-Consistency
- Divide-and-Conquer

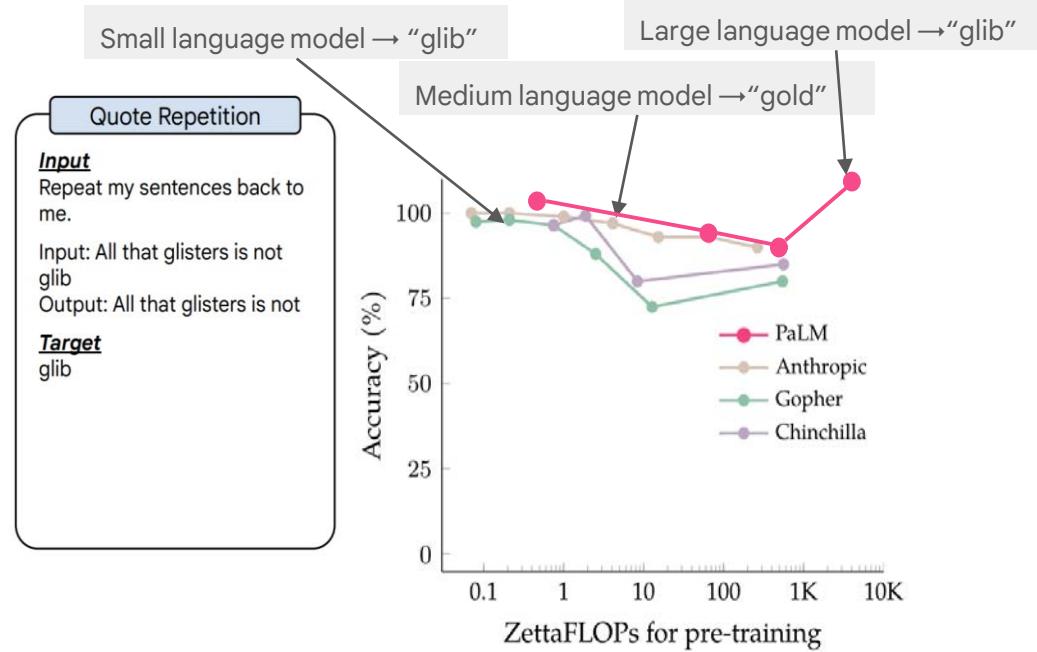
## Ask a human to

- Explain the rationale
- Double check the answer
- Decompose to easy subproblems

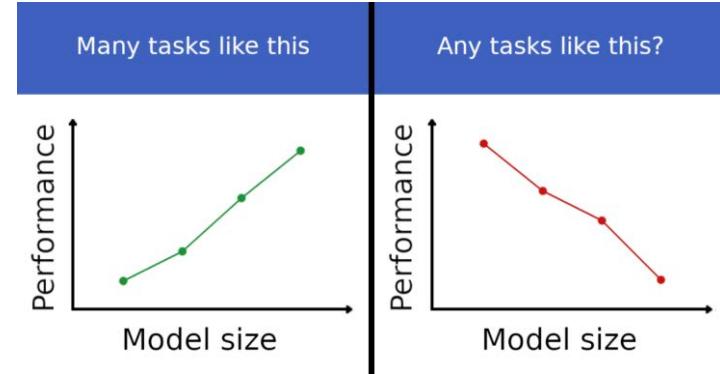
Large Language Models demonstrate some human-like behaviors!

# To be or not to be Large?

Inverse scaling can become U-shaped: To be large ?



Inverse Scaling Prize: Not to be large?

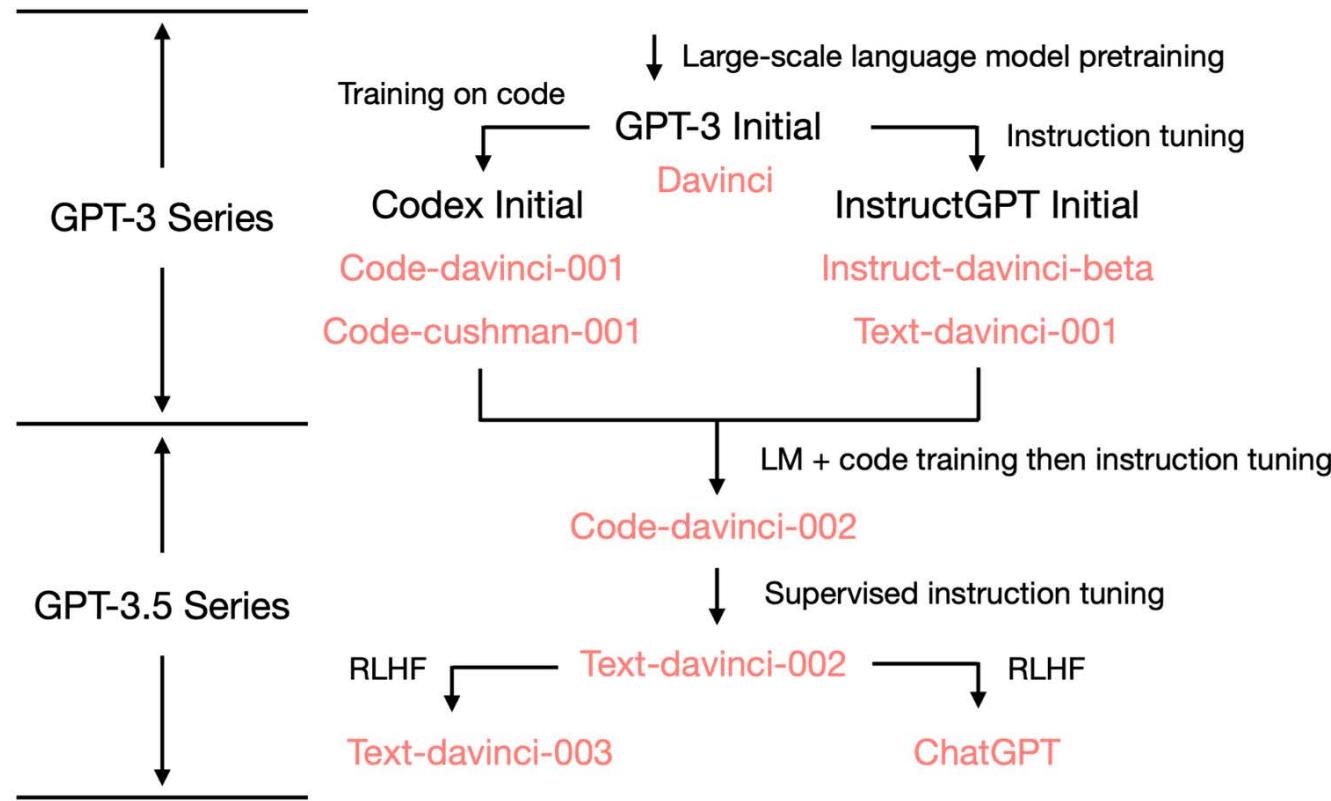


See:

- ❖ [TruthfulQA](#): The largest models were generally the least truthful
- ❖ <https://github.com/inverse-scaling/prize>
- ❖ <https://irmckenzie.co.uk/round1>

What are ChatGPT and GPT-4?

# From 2020 GPT-3 to 2022 ChatGPT



# What's ChatGPT

- Phase 1: pre-training
  - Learn **general** world knowledge, ability, etc.
- Phase 2: Supervised finetuning
  - Tailor to **tasks** (**unlock** some abilities)
- Phase 3: RLHF
  - Tailor to **humans**
  - *Even you could teach ChatGPT to do something*

Most of these were explored by InstructGPT. The only difference is that it is further trained with chat data, as an success of product (plus engineering).

# GPT-4

## What's new?

- **Make progress towards multilingualism:** GPT-4 is able to answer thousands of multiple-choice questions in 26 languages with a high degree of accuracy.
- **Longer memory for conversations:** ChatGPT can process 4,096 tokens. Once this limit was reached, the model lost track. GPT-4 can process 32,768 tokens. Enough for an entire short story on 32 A4 pages.
- **Multimodal input:** not only text can be used as input, but also images in which GPT-4 can describe objects. (**It is not released yet**)

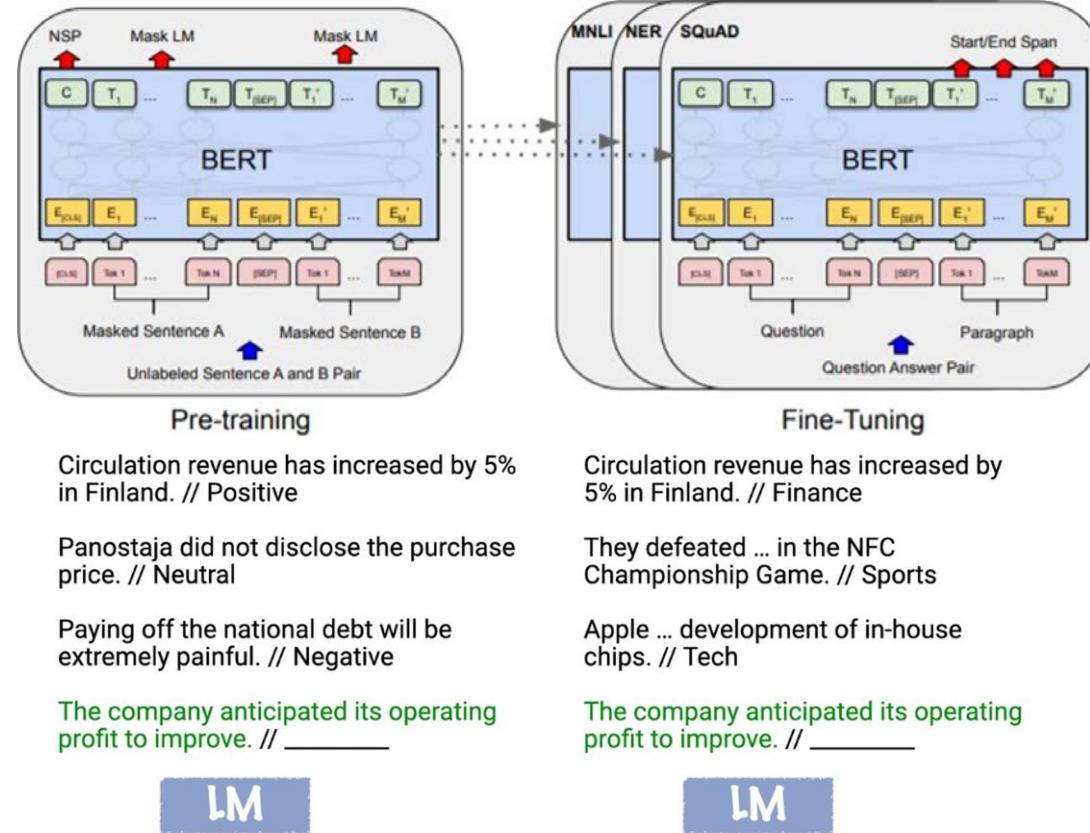
## GPT-4 Technical Report from OpenAI

- **Only contains a small amount of detail:** “[...] given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method or similar.” From [Technical Report](#).
- GPT-4’s score on the bar exam was similar to that of the top ten percent of graduates, while ChatGPT ranked in among the ten per cent that scored the worst.
- OpenAI hired more than 50 experts who interacted with and tested the model over an extended period of time.

It was finished in August 2022. It takes **7 months** for security alignment

How to use Large Language models  
(LLMs)?

# Pretraining + Fine-tuning Paradigm



Pre-training:

Trained on huge amounts of unlabeled text using “self-supervised” training objectives

Adaptation:

How to use a pretrained model for your downstream task?

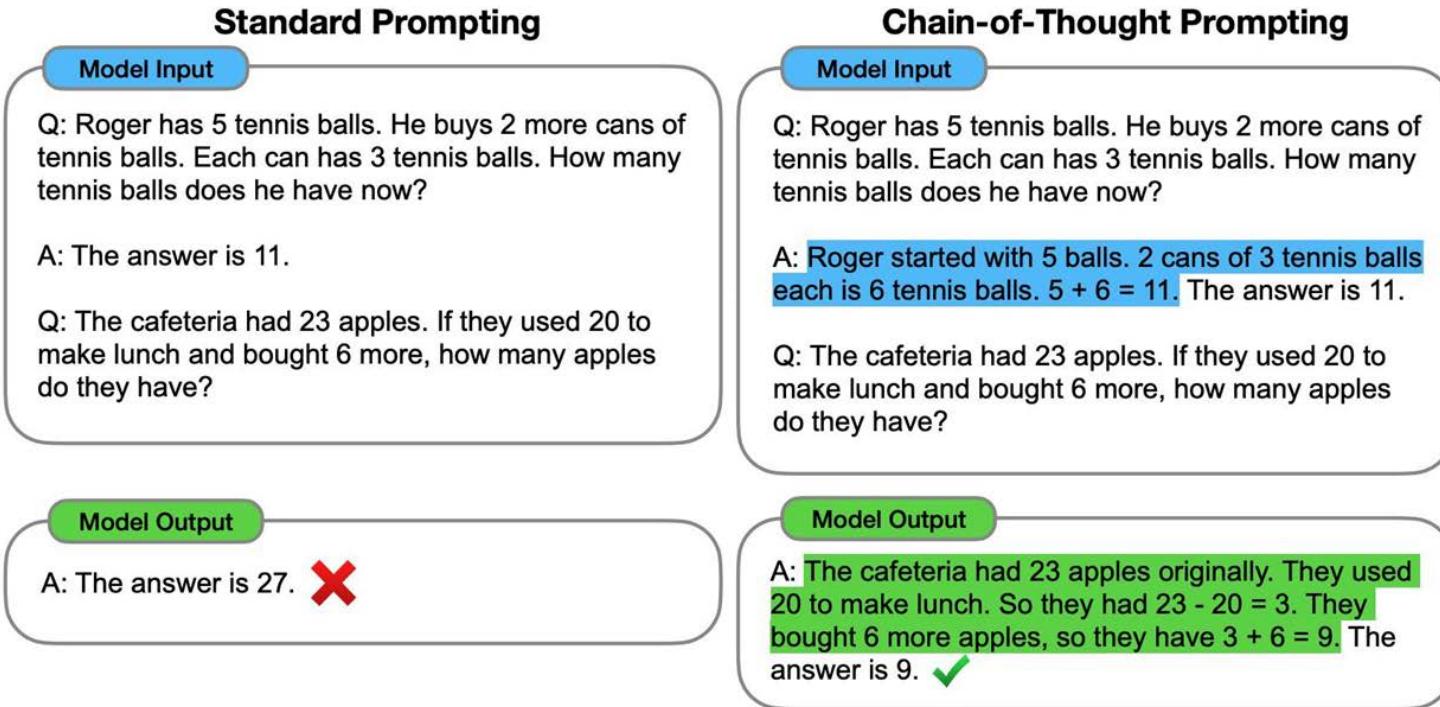
What types of NLP tasks (input and output formats)?

How many annotated examples do you have?

# Pretraining + Prompting Paradigm

- Fine-tuning (FT)
    - + Strongest performance
    - - Need curated and labeled dataset for each new task (typically 1k-100k ex.)
    - - Poor generalization, spurious feature exploitation
  - Few-shot (FS)
    - + Much less task-specific data needed
    - + No spurious feature exploitation
    - - Challenging
  - One-shot (1S)
    - + "Most natural," e.g. giving humans instructions
    - - Challenging
  - Zero-shot (OS)
    - + Most convenient
    - - Challenging, can be ambiguous
- Stronger  
task-specific  
performance**
- 
- More convenient,  
general, less data**

# Chain of Thoughts Prompting



# Zero-Shot CoT Prompting

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

*(Output) The answer is 8. X*

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

*(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓*

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

*(Output) 8 X*

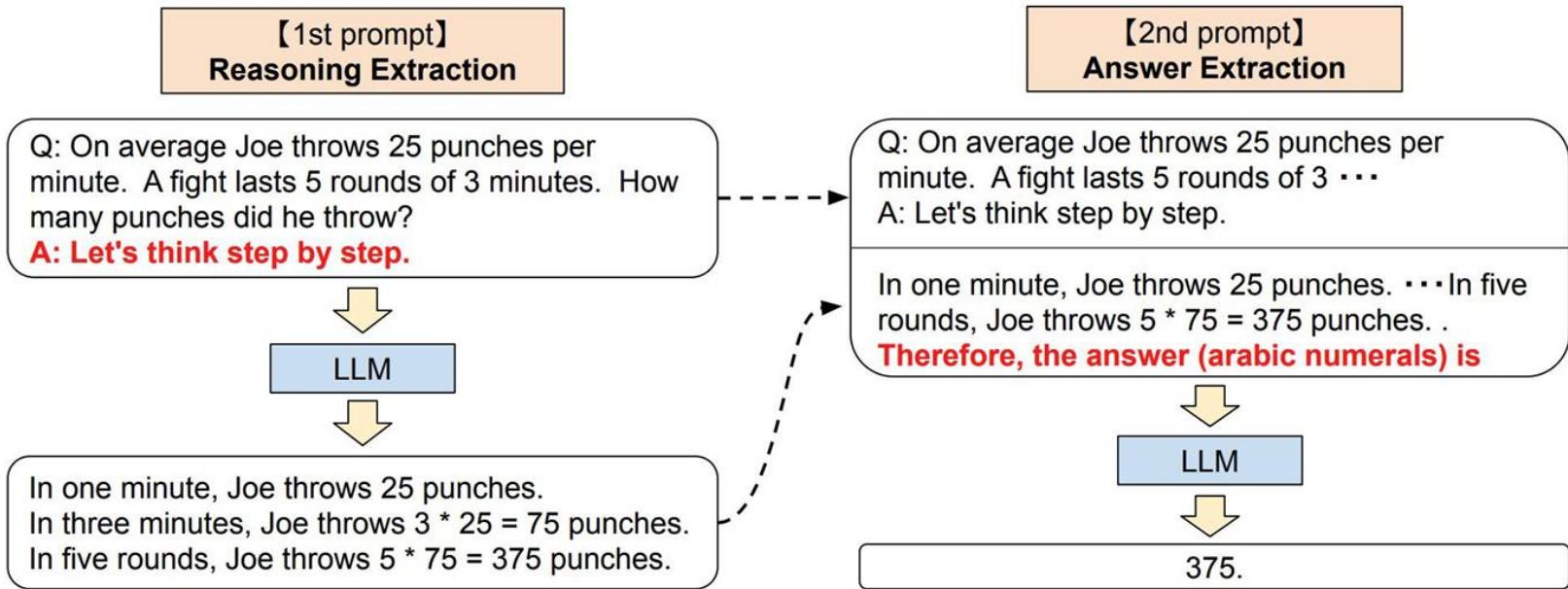
(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

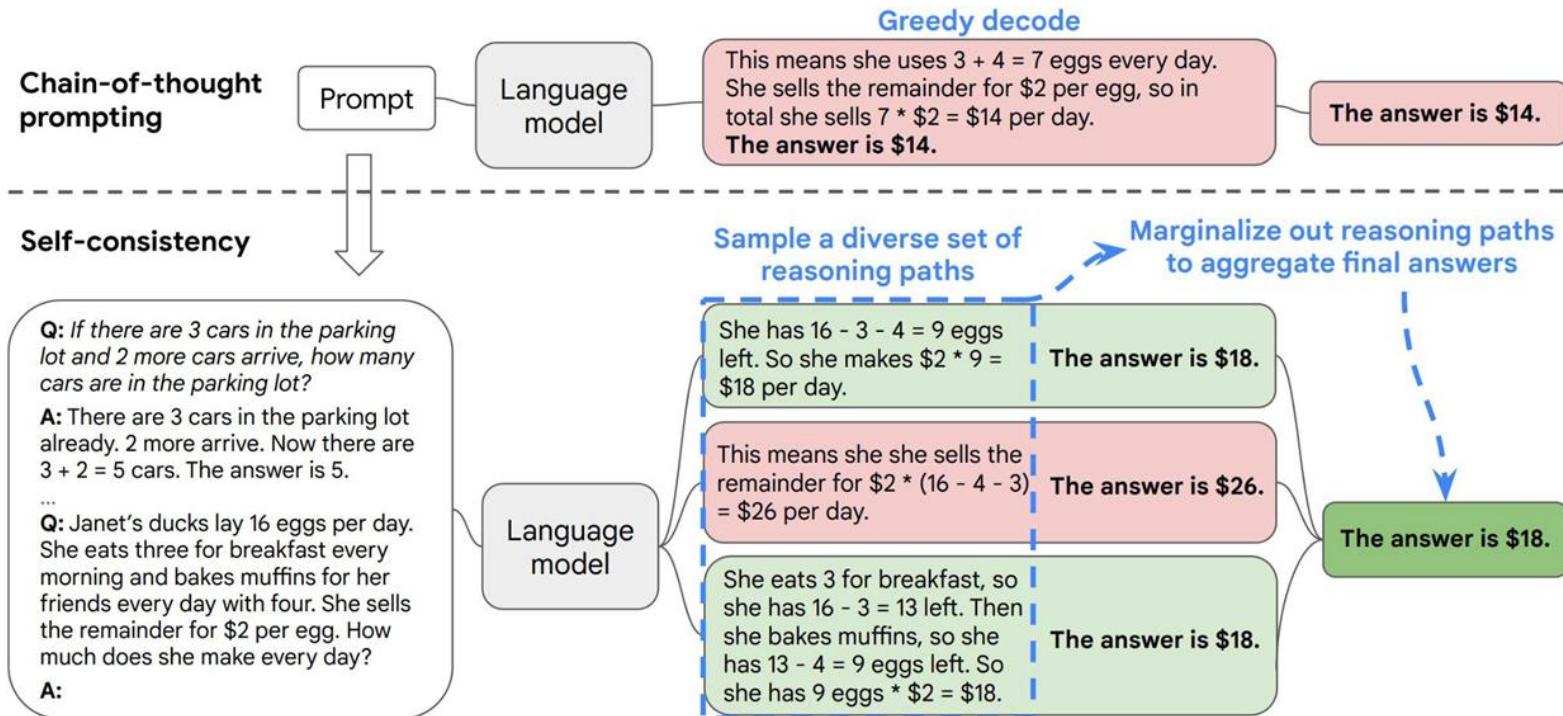
A: **Let's think step by step.**

*(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

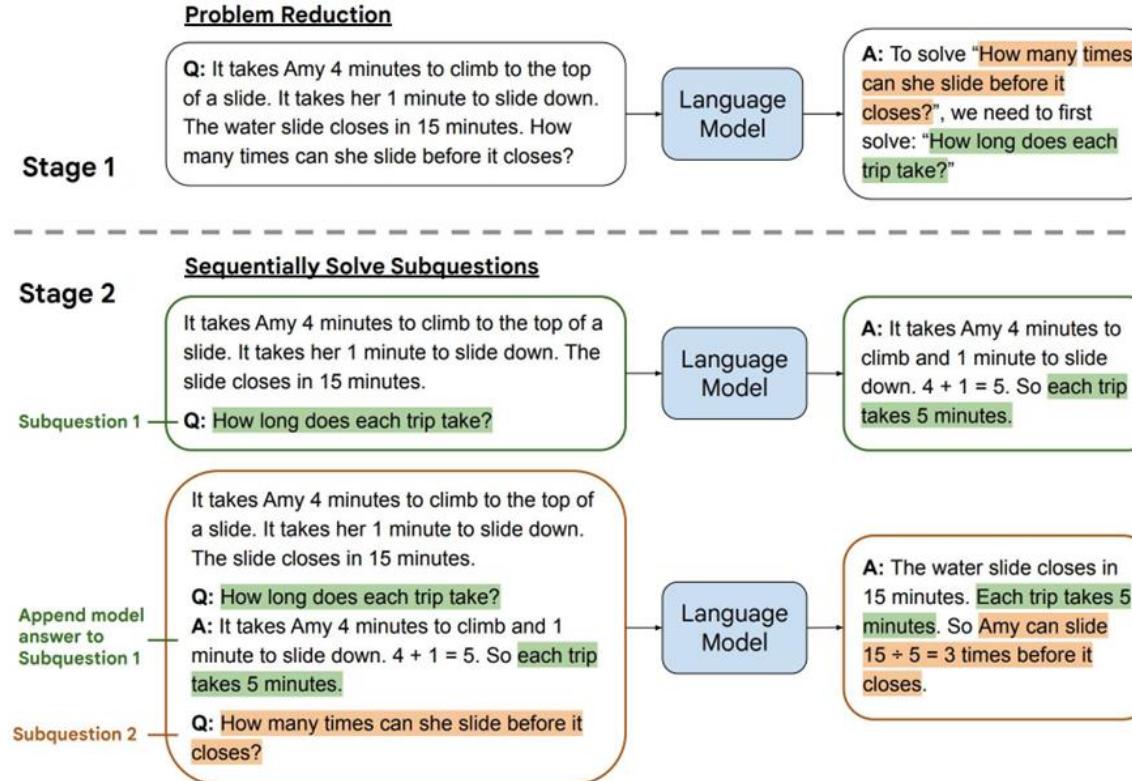
# Zero-Shot CoT Prompting



# Self-Consistency Prompting



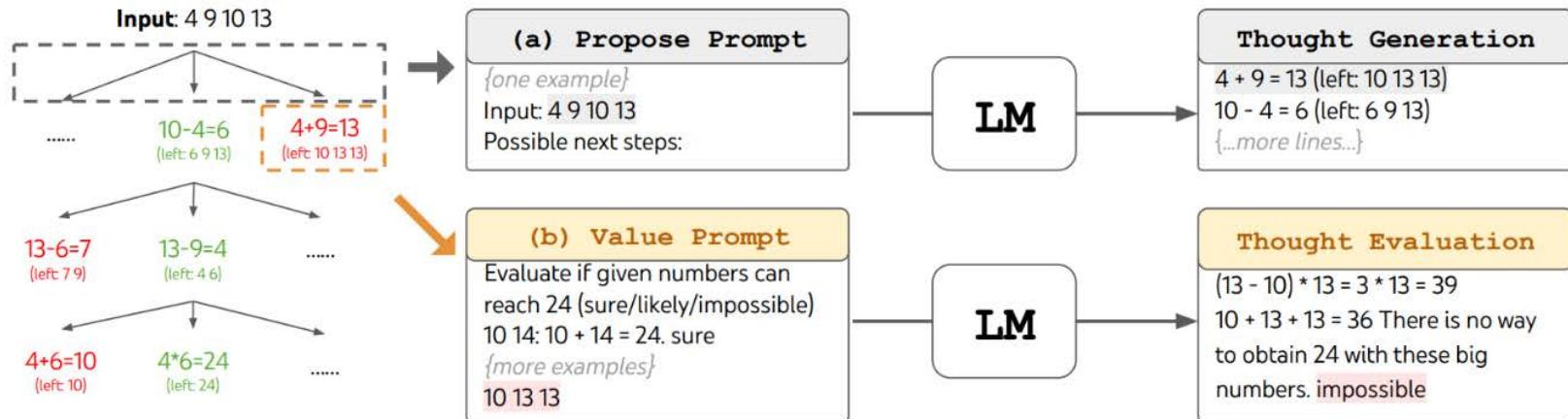
# Least-to-Most Prompting



# Tree of Thought

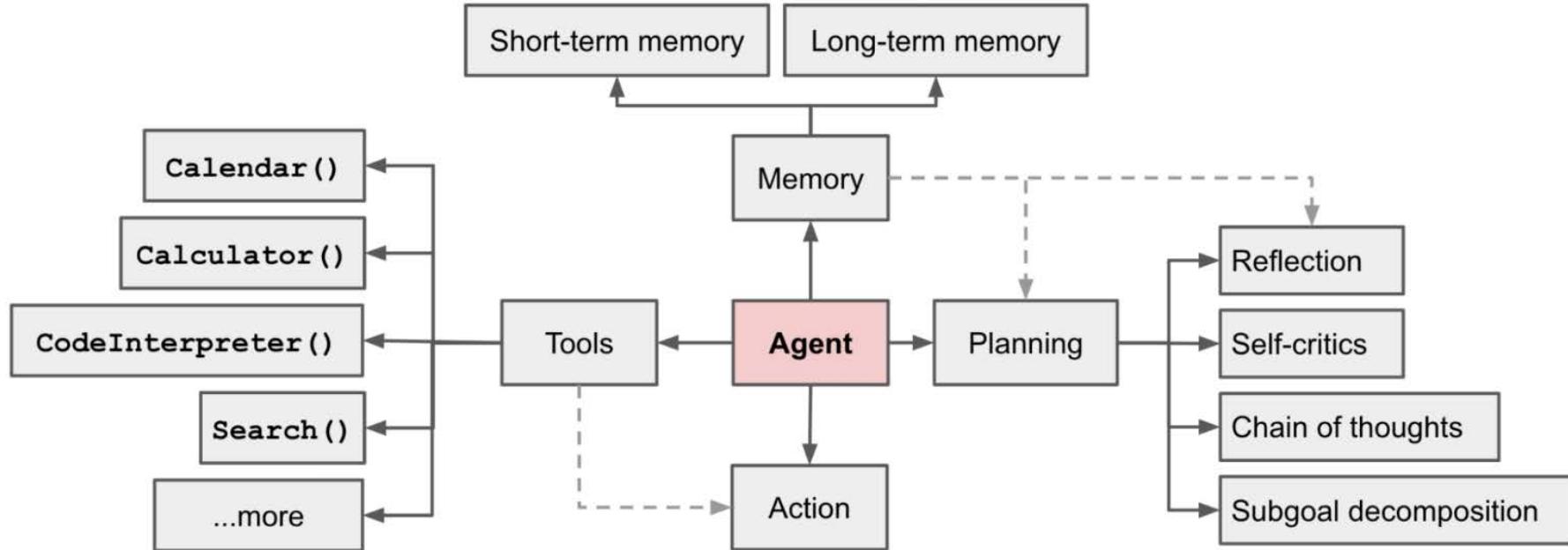
## 4.1 Game of 24

Game of 24 is a mathematical reasoning challenge, where the goal is to use 4 numbers and basic arithmetic operations (+-\*%) to obtain 24. For example, given input “4 9 10 13”, a solution output could be “ $(10 - 4) * (13 - 9) = 24$ ”.



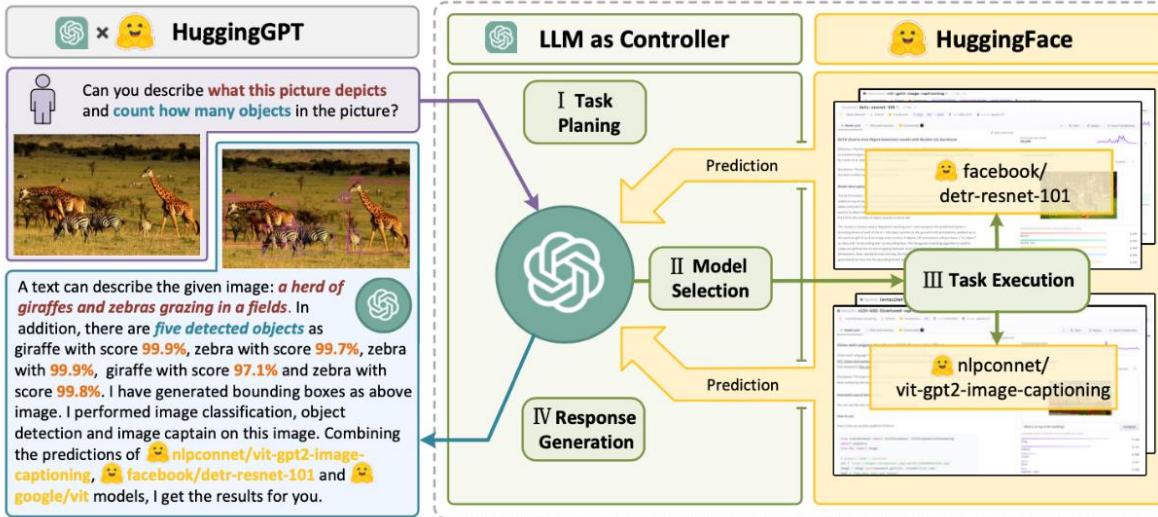
# Agent

LLM acts as a Decision Center (Reasoning) and Human Interaction Front end (Chat)



# Agent: Tool use

The biggest difference between humans and animals is the ability to use tools



**HuggingGPT** (Shen et al. 2023) is a framework to use ChatGPT as the task planner to select models available in HuggingFace platform according to the model descriptions and summarize the response based on the execution results.

---

## Algorithm 1 API call process

```
1: Input:  $us \leftarrow UserStatement$ 
2: if API Call is needed then
3:   while API not found do
4:      $keywords \leftarrow summarize(us)$ 
5:      $api \leftarrow search(keywords)$ 
6:     if Give Up then
7:       break
8:     end if
9:   end while
10:  if API found then
11:     $api\_doc \leftarrow api.documentation$ 
12:    while Response not satisfied do
13:       $api\_call \leftarrow gen\_api\_call(api\_doc, us)$ 
14:       $api\_re \leftarrow execute\_api\_call(api\_call)$ 
15:      if Give Up then
16:        break
17:      end if
18:    end while
19:  end if
20: end if
21: if response then
22:    $re \leftarrow generate\_response(api\_re)$ 
23: else
24:    $re \leftarrow generate\_response()$ 
25: end if
26: Output:  $ResponseToUser$ 
```

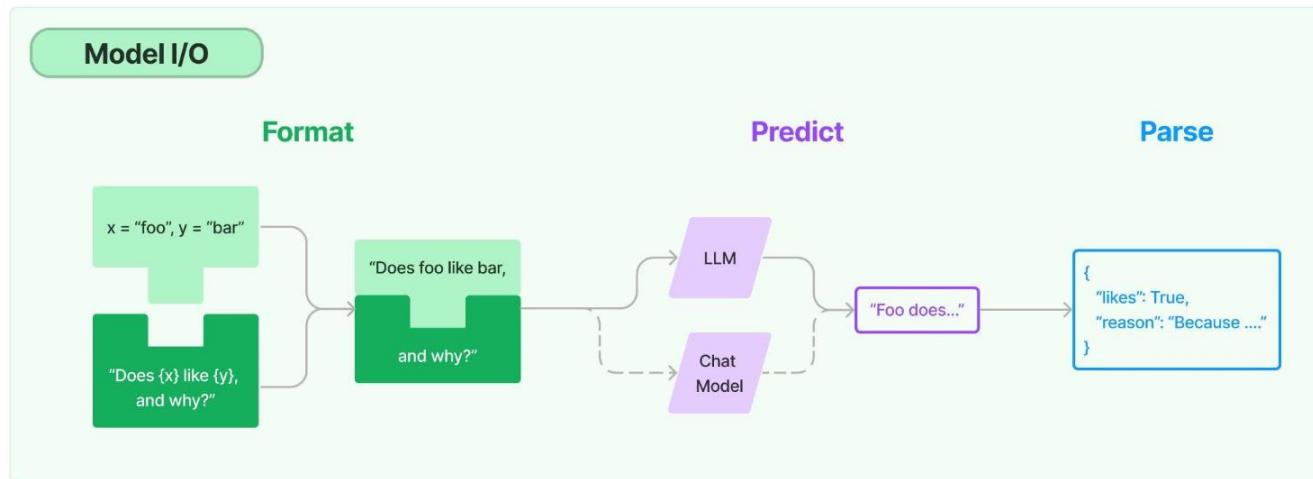
Pseudo code of how LLM makes an API call in API-Bank.

**API-Bank** (Li et al. 2023) : A benchmark for evaluating the performance of tool-augmented LLMs. It contains 53 commonly used API tools, a complete tool-augmented LLM workflow, and 264 annotated dialogues that involve 568 API calls.

# Langchain



- ❖ LangChain is a framework for developing applications powered by language models.
- ❖ The core building block of LangChain applications is the LLMChain. This combines three things:
  - LLM: The language model is the core reasoning engine here. In order to work with LangChain, you need to understand the different types of language models and how to work with them.
  - Prompt Templates: This provides instructions to the language model. This controls what the language model outputs, so understanding how to construct prompts and different prompting strategies is crucial.
  - Output Parsers: These translate the raw response from the LLM to a more workable format, making it easy to use the output downstream.



# Contents

- Philosophy of this course
- Large language models
- **Introduction to ChatGPT**

# ChatGPT

- ▶ Reaching 1M users in five days; research 100M users in two months
- ▶ Everyone discusses ChatGPT, its spreading speed is faster than COVID 19
- ▶ Red alarms in Google
- ▶ Google released Bard very soon, but it performs worse, stock valued reduced by 8%
- ▶ Microsoft invests 10B dollars to OpenAI
- ▶ New Bing and Office used ChatGPT
- ▶ 百模大战 in China

## 用户数突破100万用时

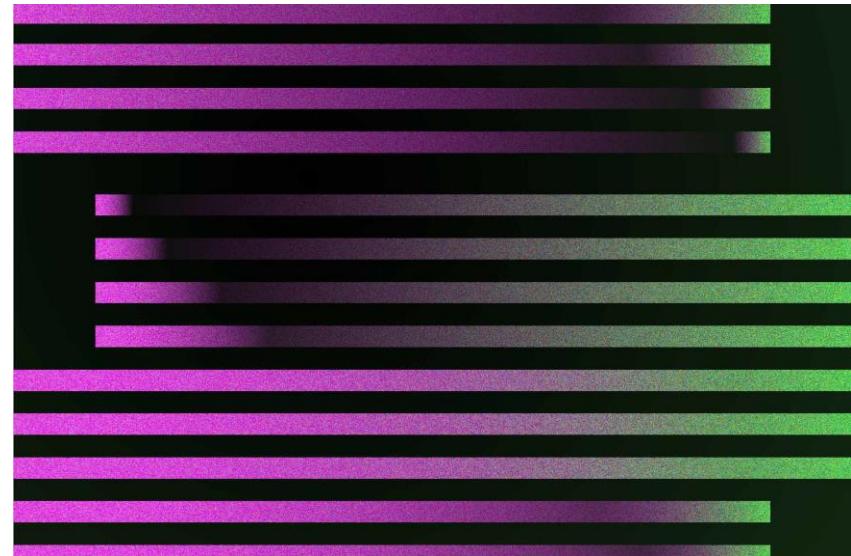
- GPT-3: 24个月
- Copilot: 6个月
- DALL-E: 2.5个月
- **ChatGPT: 5天**
- Netflix - 41个月
- Twitter - 24个月
- Facebook - 10个月
- Instagram - 2.5个月

# ChatGPT

## ChatGPT: Optimizing Language Models for Dialogue

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

November 30, 2022  
13 minute read



We are excited to introduce ChatGPT to get users' feedback and learn about its strengths and weaknesses. During the research preview, usage of ChatGPT is free. Try it now at [chat.openai.com](https://chat.openai.com).

ChatGPT Blog: <https://openai.com/blog/chatgpt/>

# ChatGPT

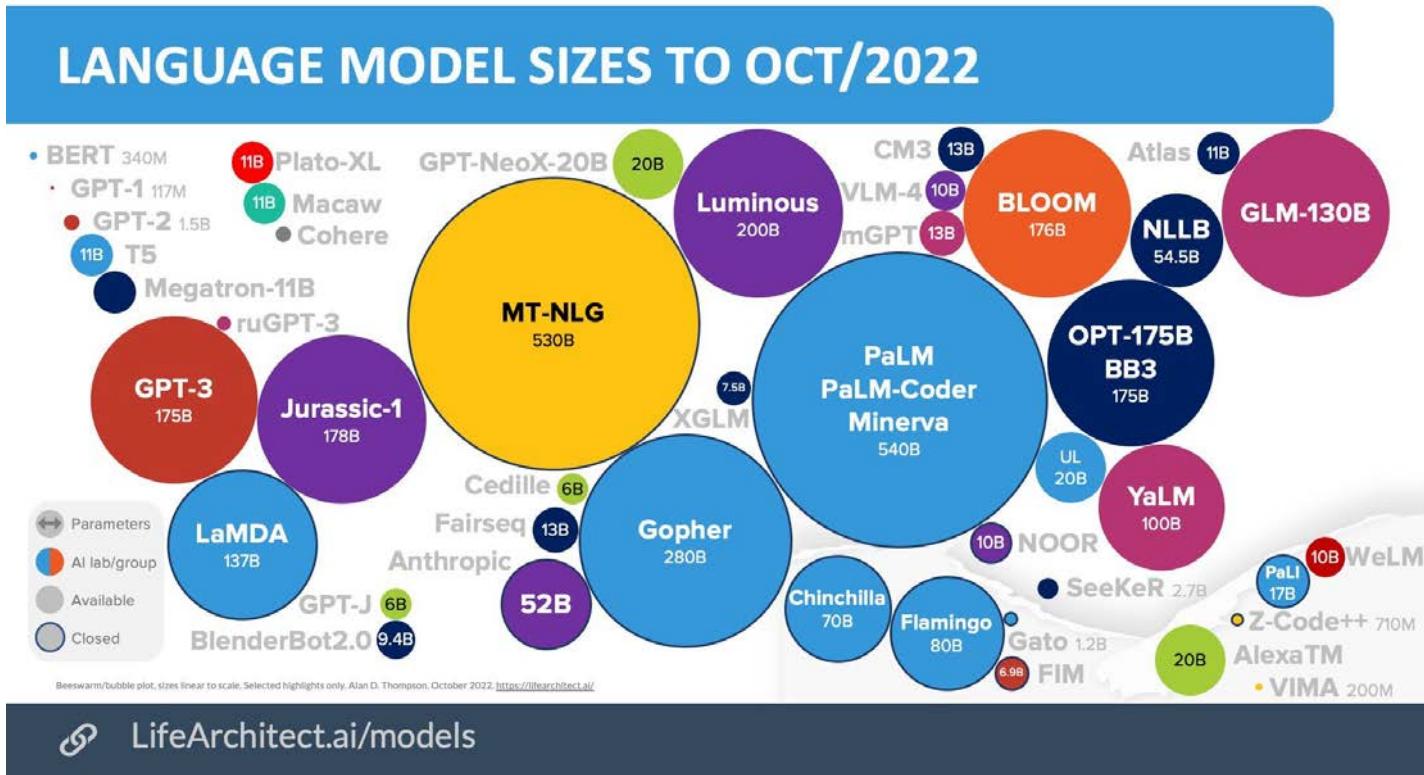
The main features of ChatGPT highlighted in the official blog:

- ▶ answer followup questions
- ▶ admit its mistakes
- ▶ challenge incorrect premises
- ▶ reject inappropriate requests

ChatGPT Blog: <https://openai.com/blog/chatgpt/>

# The Size of ChatGPT

ChatGPT is based on Davinci-3



# Size of ChatGPT

Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 \times 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 \times 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 \times 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 \times 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2 \times 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 \times 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6 \times 10^{-4}$

## Four models released by OpenAI:

Language models

Base models

Ada Fastest

\$0.0004 /1K tokens

Babbage

\$0.0005 /1K tokens

Curie

\$0.0020 /1K tokens

Davinci Most powerful

\$0.0200 /1K tokens

Multiple models, each with different capabilities and price points.  
Ada is the fastest model, while Davinci is the most powerful.

# Size of ChatGPT

The size of Davinci (GPT 3) could be 175B

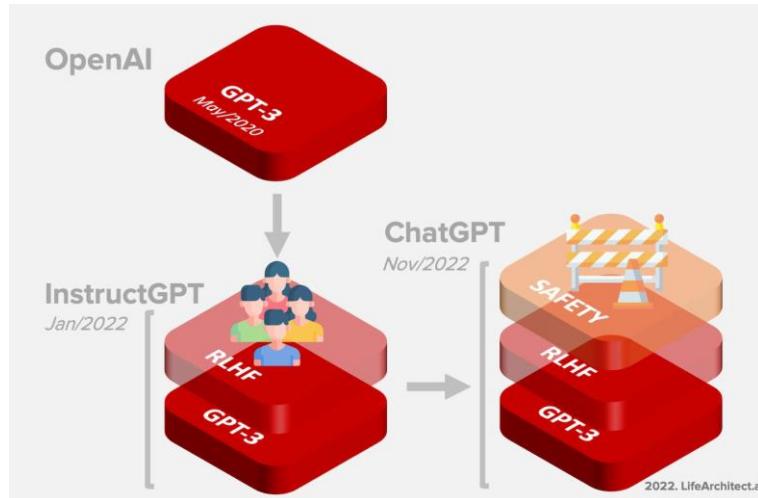
Model	LAMBADA ppi ↓	LAMBADA acc ↑	Winogrande ↑	Hellaswag ↑	PIQA ↑
GPT-3-124M	18.6	42.7%	52.0%	33.7%	64.6%
GPT-3-350M	9.09	54.3%	52.1%	43.6%	70.2%
Ada	9.95	51.6%	52.9%	43.4%	70.5%
GPT-3-760M	6.53	60.4%	57.4%	51.0%	72.9%
GPT-3-1.3B	5.44	63.6%	58.7%	54.7%	75.1%
Babbage	5.58	62.4%	59.0%	54.5%	75.5%
GPT-3-2.7B	4.60	67.1%	62.3%	62.8%	75.6%
GPT-3-6.7B	4.00	70.3%	64.5%	67.4%	78.0%
Curie	4.00	68.5%	65.6%	68.5%	77.9%
GPT-3-13B	3.56	72.5%	67.9%	70.9%	78.5%
GPT-3-175B	3.00	76.2%	70.2%	78.9%	81.0%
Davinci	2.97	74.8%	70.2%	78.1%	80.4%

All GPT-3 figures are from the [GPT-3 paper](#); all API figures are computed using eval harness

Ada, Babbage, Curie and Davinci line up closely with 350M, 1.3B, 6.7B, and 175B respectively.  
Obviously this isn't ironclad evidence that the models *are* those sizes, but it's pretty suggestive.

Leo Gao, On the Sizes of OpenAI API Models, <https://blog.eleuther.ai/gpt3-model-sizes/>

# ChatGPT timeline



## Timeline to ChatGPT

Date	Milestone
11/Jun/2018	GPT-1 announced on the OpenAI blog.
14/Feb/2019	GPT-2 announced on the OpenAI blog.
28/May/2020	Initial GPT-3 preprint paper published to arXiv.
11/Jun/2020	GPT-3 API private beta.
22/Sep/2020	GPT-3 licensed to Microsoft.
18/Nov/2021	GPT-3 API opened to the public.
27/Jan/2022	InstructGPT released, now known as GPT-3.5. InstructGPT preprint paper Mar/2022.
28/Jul/2022	Exploring data-optimal models with FIM, paper on arXiv.
1/Sep/2022	GPT-3 model pricing cut by 66% for davinci model.
21/Sep/2022	Whisper (speech recognition) announced on the OpenAI blog.
28/Nov/2022	GPT-3.5 expanded to text-davinci-003, announced via email: <ol style="list-style-type: none"><li>1. Higher quality writing.</li><li>2. Handles more complex instructions.</li><li>3. Better at longer form content generation.</li></ol>
30/Nov/2022	ChatGPT announced on the OpenAI blog.

Next...

GPT-4...

Alan D. Thompson, GPT-3.5 + ChatGPT: An illustrated overview, <https://lifearchitect.ai/chatgpt/>

GPT 5

**Stronger Reasoning  
More efficient**

Coming from Oct. to Dec.

# What is language modeling?

A **language model** assigns a probability to a N-gram

$$f: V^n \rightarrow R^+$$

A **conditional language model** assigns a probability of a word given some conditioning context

$$g: (V^{n-1}, V) \rightarrow R^+$$

And       $p(w_n | w_1 \dots w_{n-1}) = g(w_1 \dots w_{n-1}, w) = \frac{f(w_1 \dots w_n)}{f(w_1 \dots w_{n-1})}$

$p(w_n | w_1 \dots w_{n-1})$  is the foundation of **modern large language models** (GPT, ChatGPT, etc.)

# Language models: Narrow Sense

A probabilistic model that assigns a probability to every finite sequence (grammatical or not)

Sentence: “the cat sat on the mat”

$$\begin{aligned} P(\text{the cat sat on the mat}) &= P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat}) \\ &\quad * P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on}) \\ &\quad * P(\text{mat}|\text{the cat sat on the}) \end{aligned}$$

Implicit order

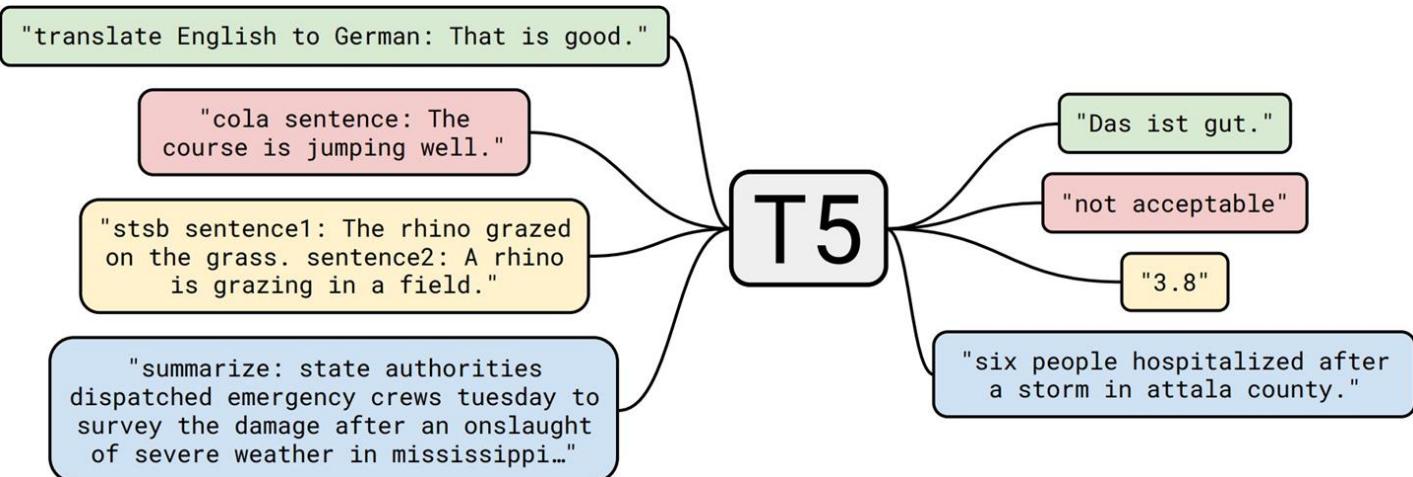
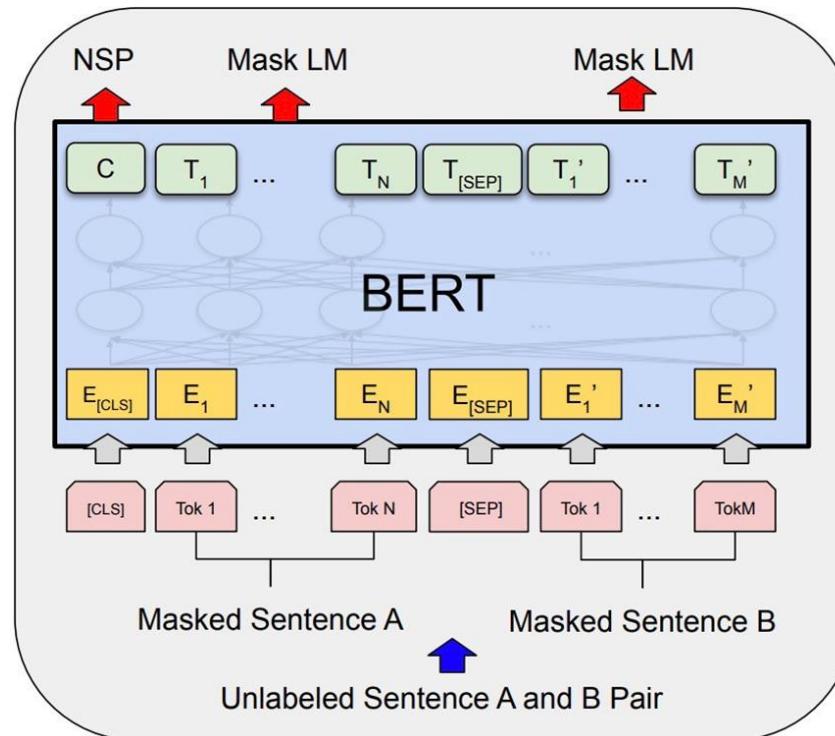


GPT-3 still acts in this way but the model is implemented as a very large neural network of 175-billion parameters!

# Language models:Broad Sense

- ❖ Decoder-only models (GPT-x models)
- ❖ Encoder-only models (BERT, RoBERTa, ELECTRA)
- ❖ Encoder-decoder models (T5, BART)

The latter two usually involve a different **pre-training** objective.

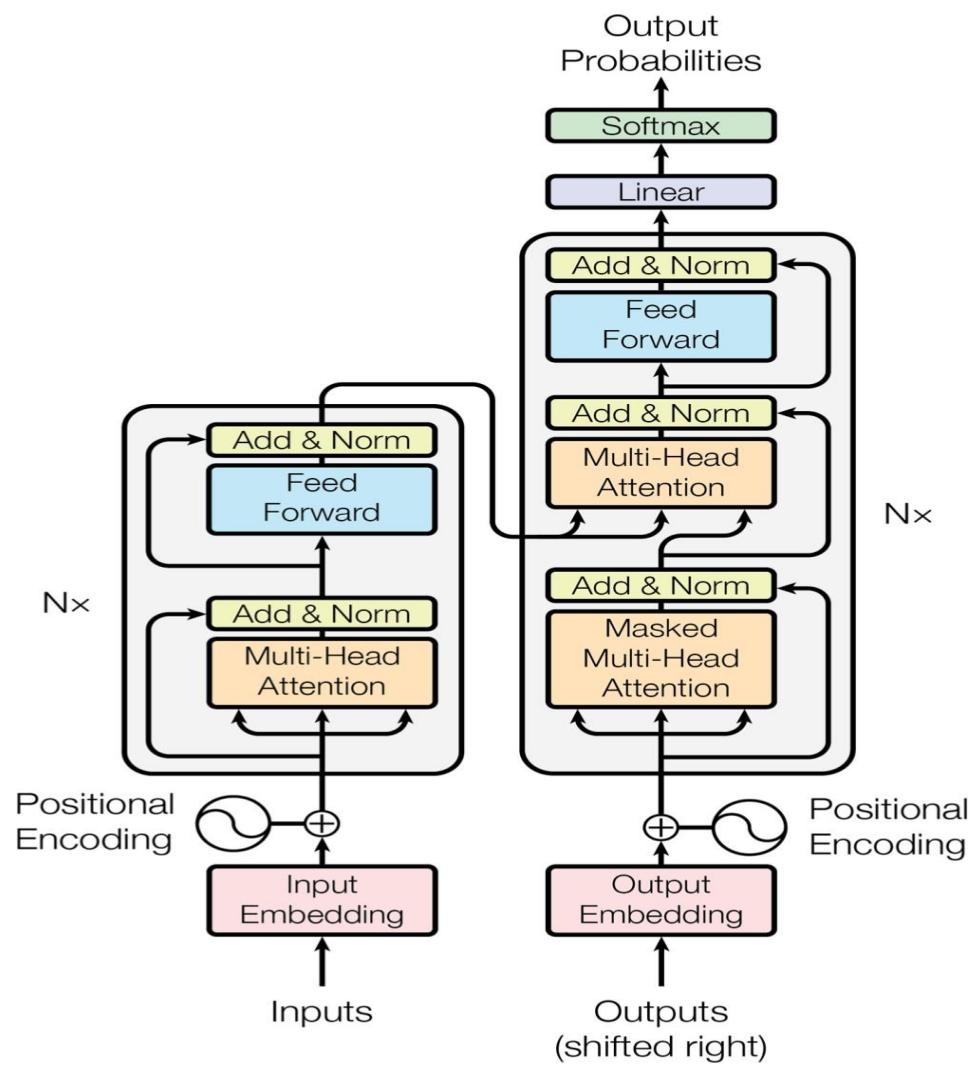


# Transformer

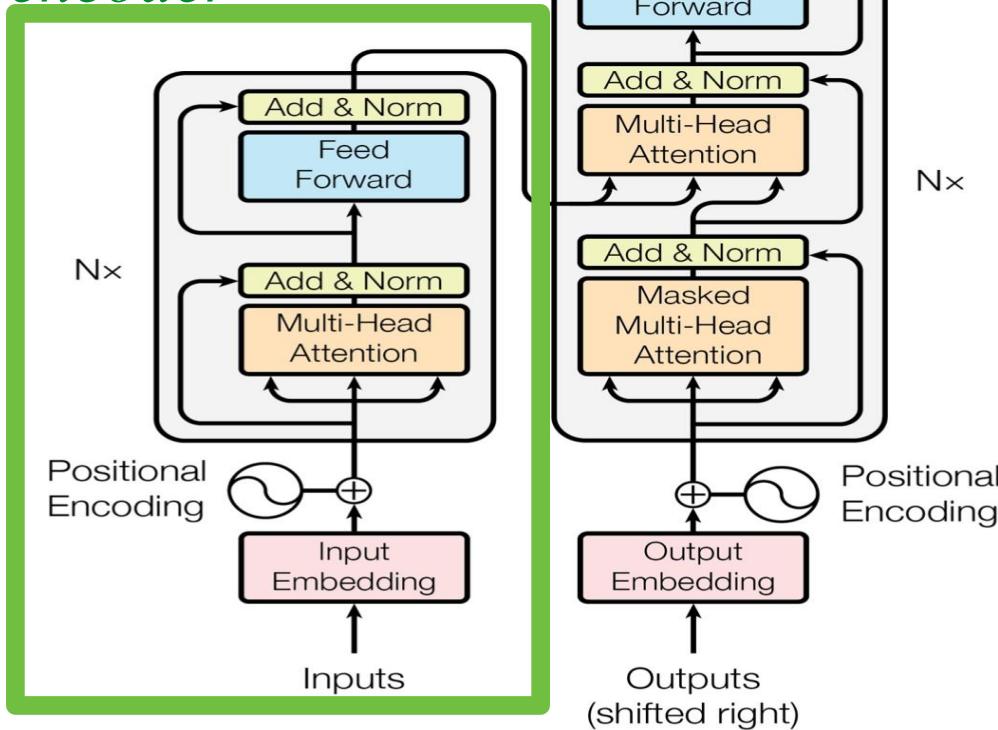
- Encoder
- Decoder
- Self-attention
- Multi-head self-attention
- Positional Encoding

# Today's lecture

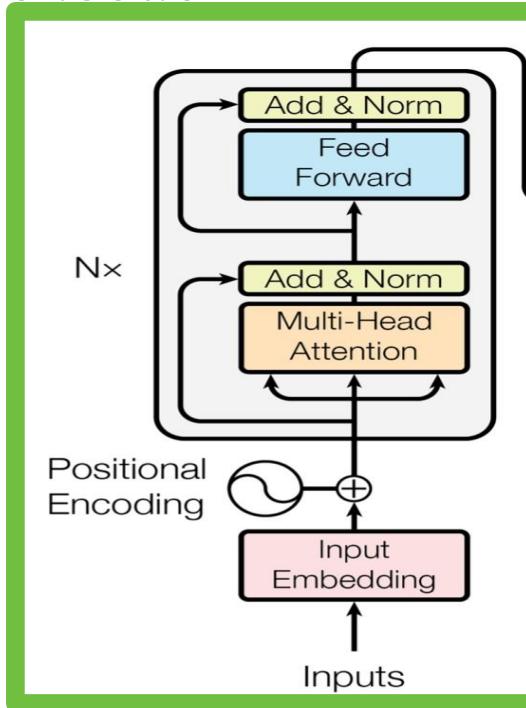
- MLP
  - +: Strongest inductive bias: if all words are concated
  - +: Weakest inductive bias: if all words are averaged
  - : The interaction at the token-level is too weak
- CNN & RNN
  - +: The interaction at the token-level is slightly better.
    - CNN: Bringing the global token-level interaction to the window-level
      - : Make simplifications, its global dependencies are limited
    - RNN: An ideal method for processing token sequences
      - : Its recursive nature has the problem of disaster forgetting.
- Transformer
  - +: Achieve **global dependence** at the **token-level** by **decoupling** token-level interaction and feature-level abstraction into two components, in **SAN** and **FNN**.
- Scaling law and emergent ability



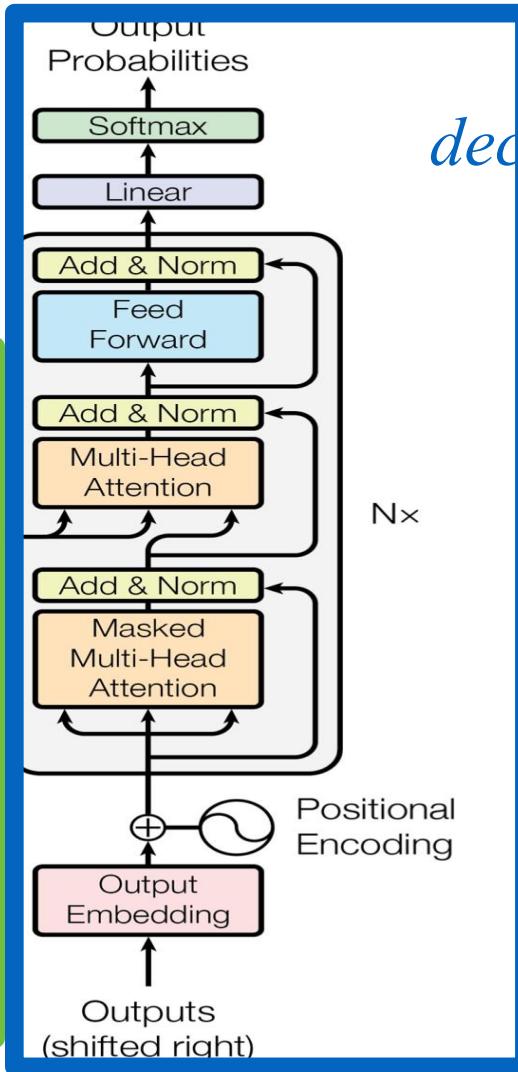
## *encoder*



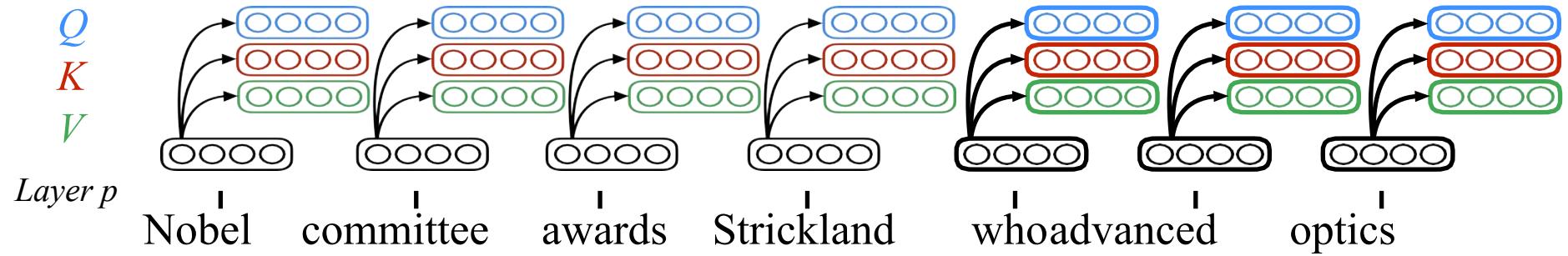
*encoder*



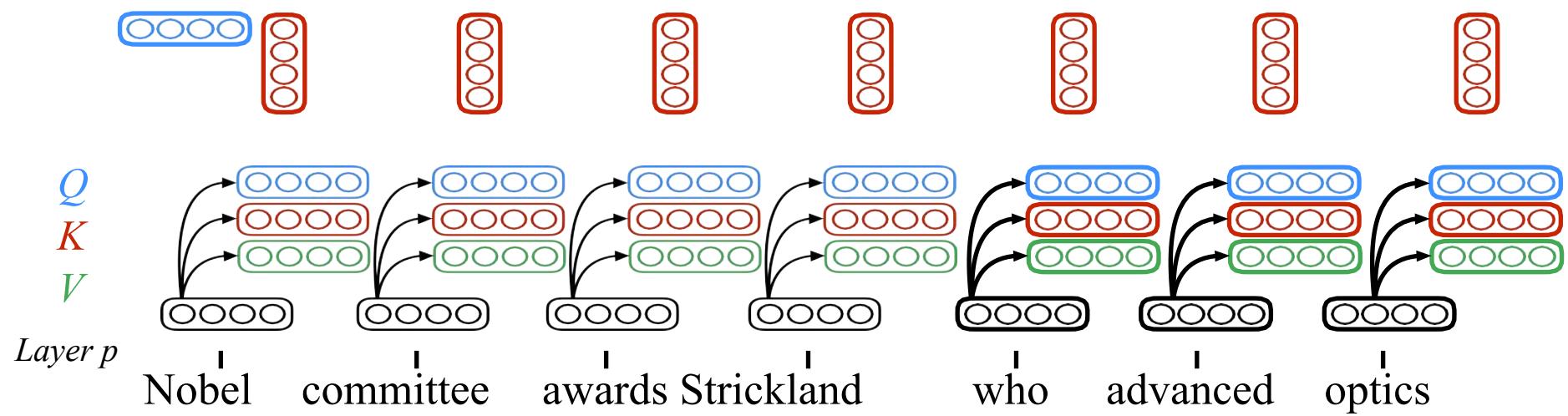
*decoder*



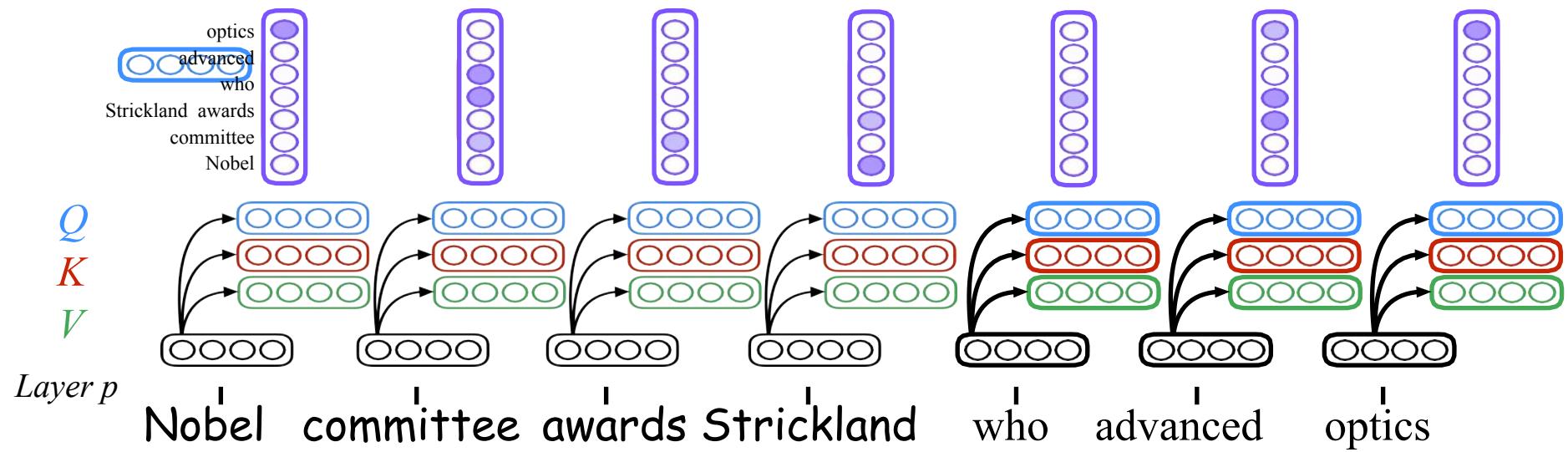
# Self-attention (in encoder)



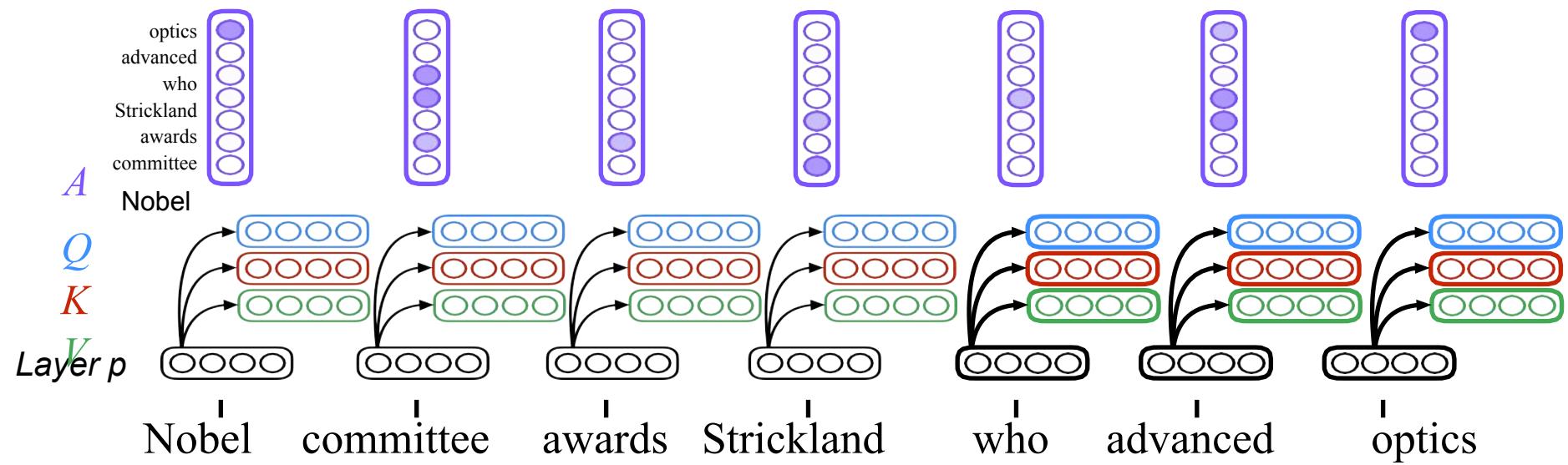
# Self-attention (in encoder)



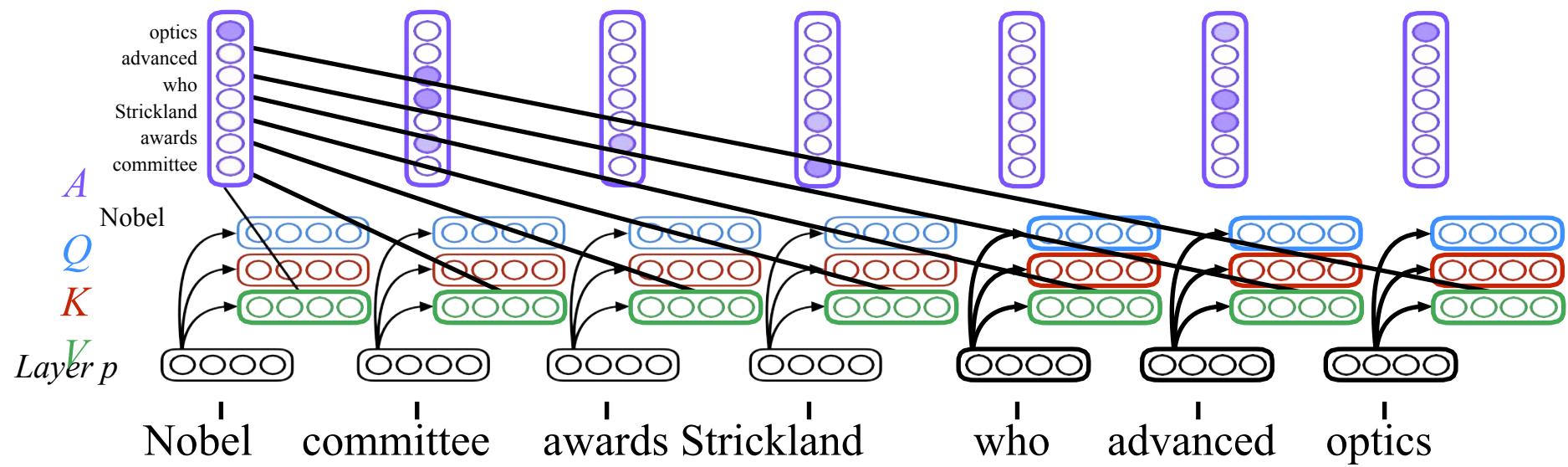
# Self-attention (in encoder)



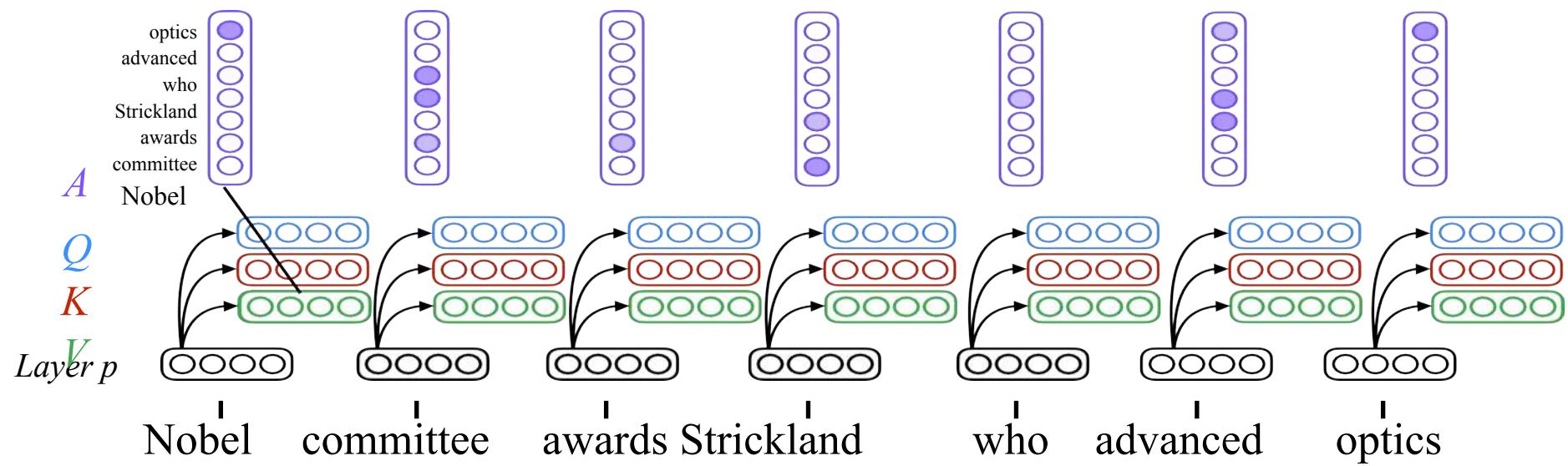
# Self-attention (in encoder)



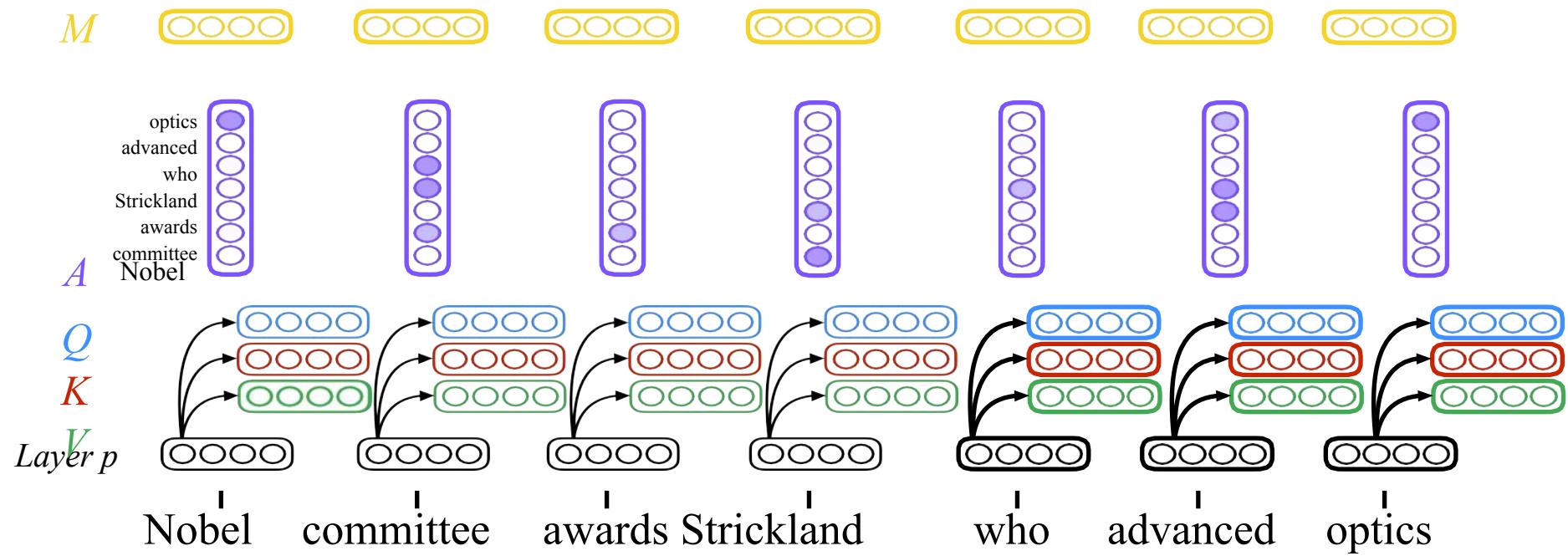
# Self-attention (in encoder)



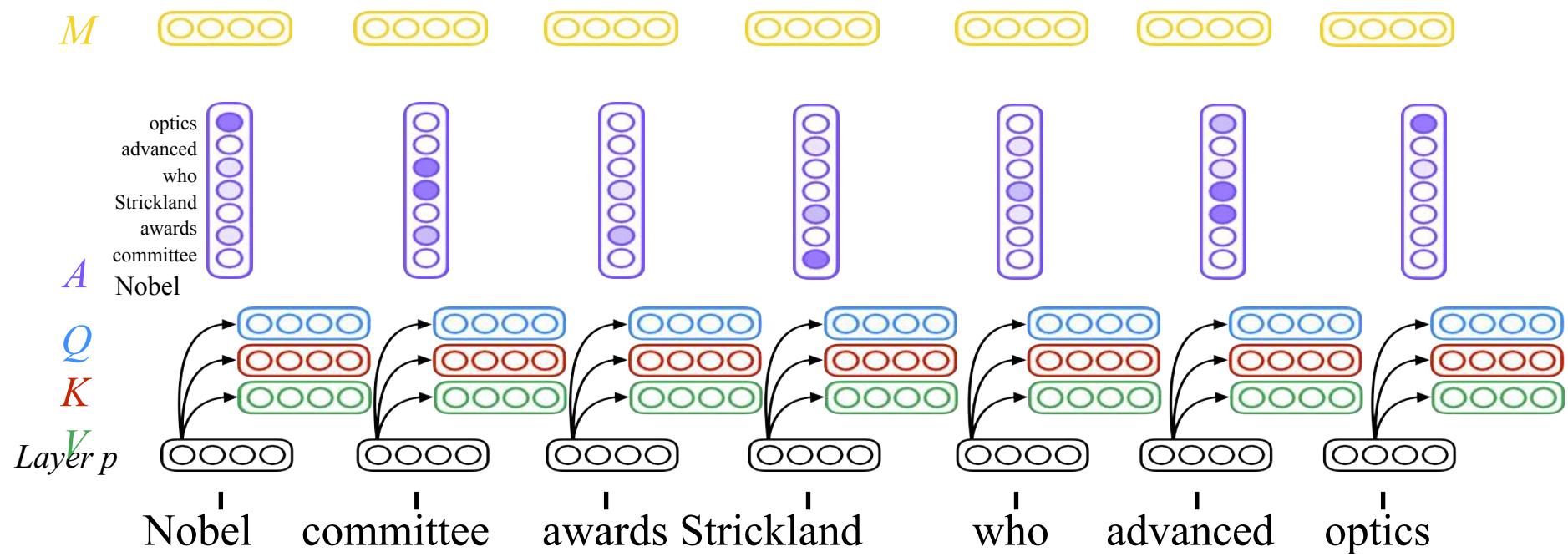
# Self-attention (in encoder)



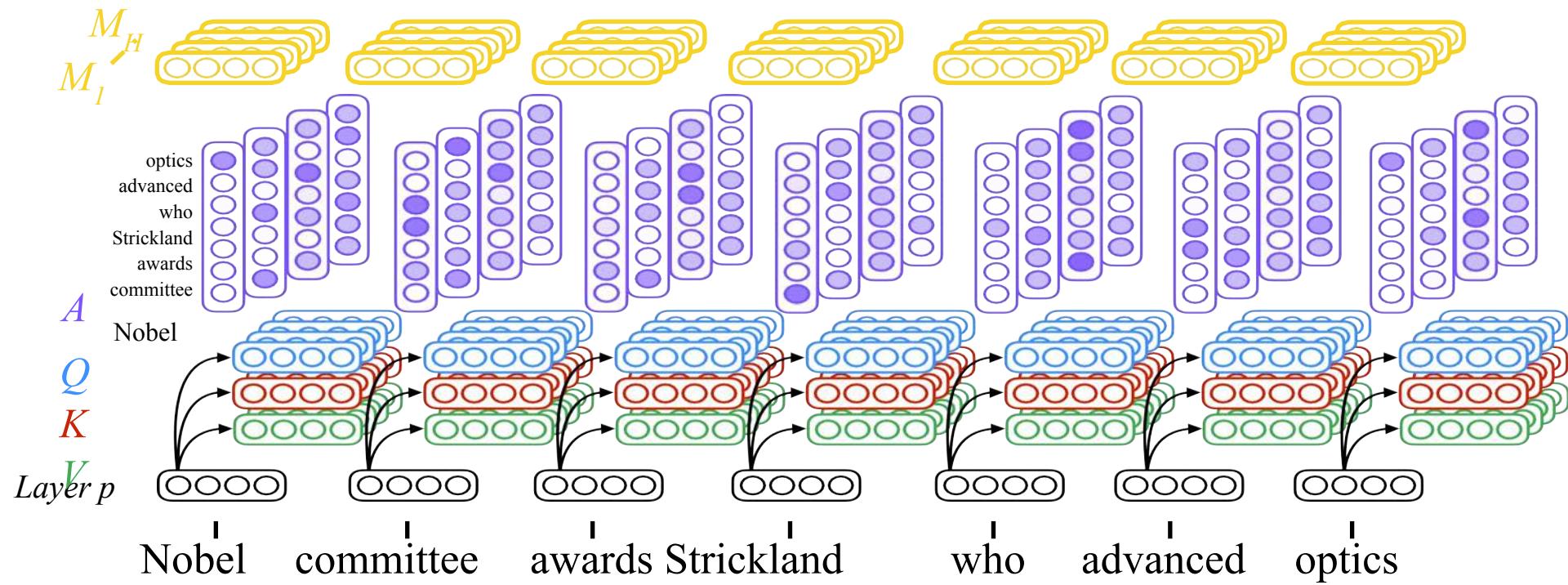
# Self-attention (in encoder)



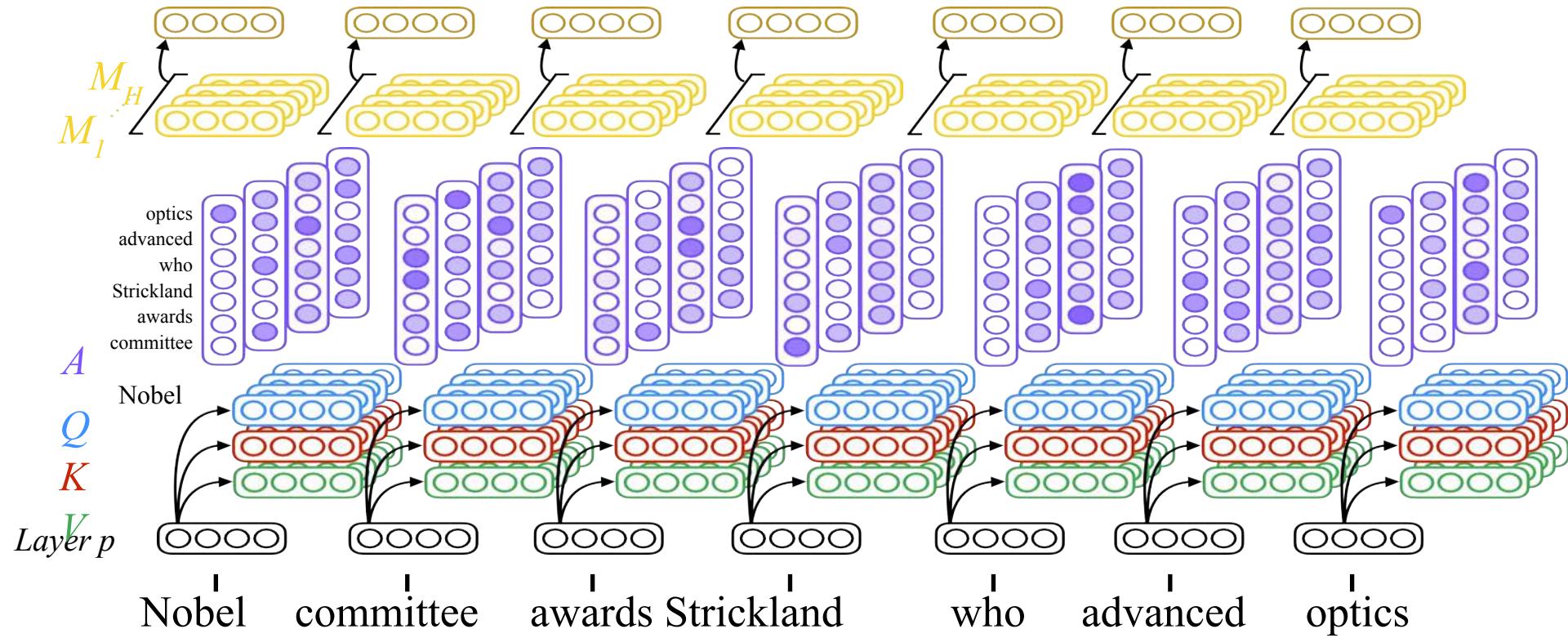
# Self-attention (in encoder)



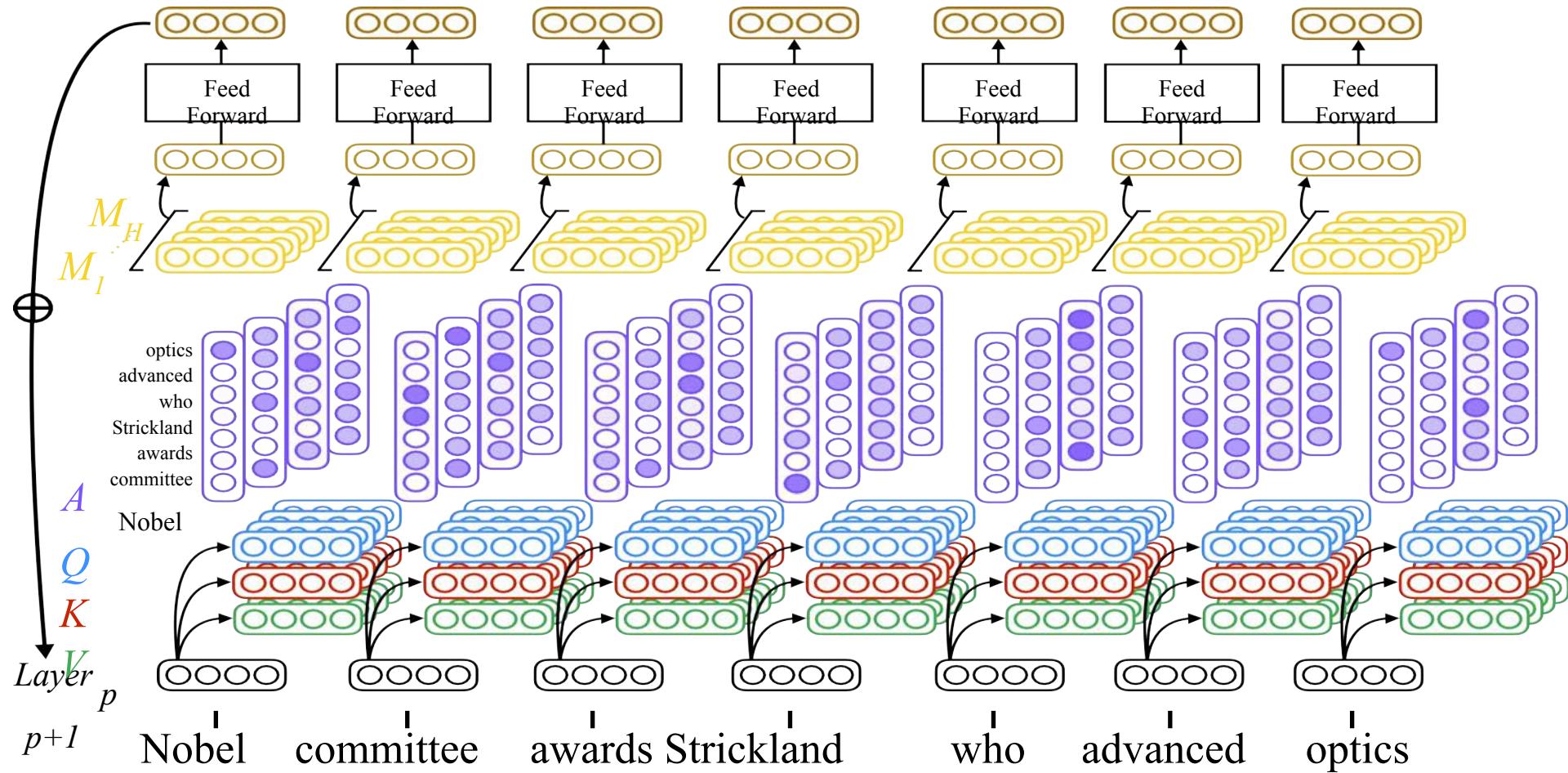
# Multi-head self-attention



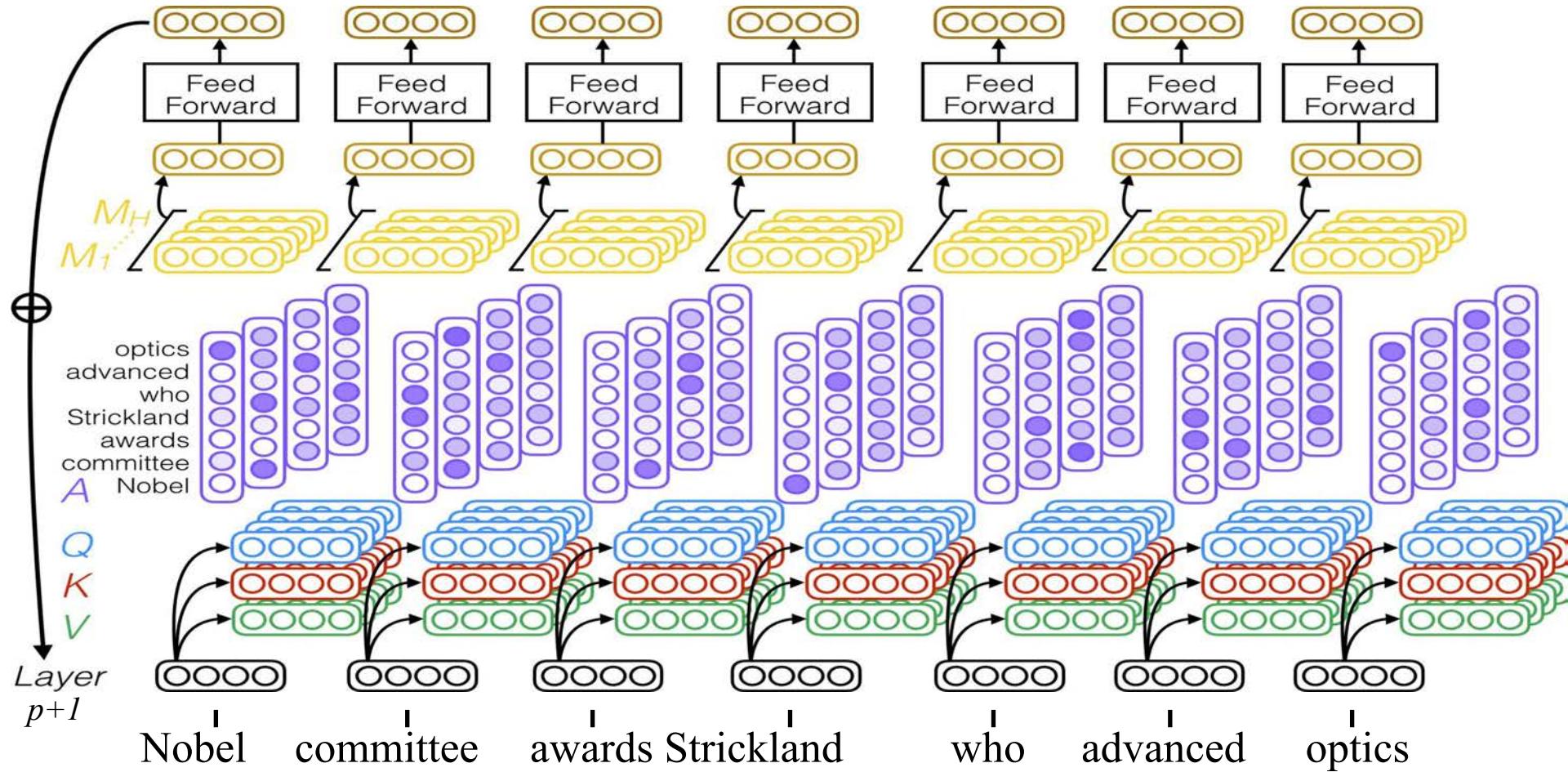
# Multi-head self-attention



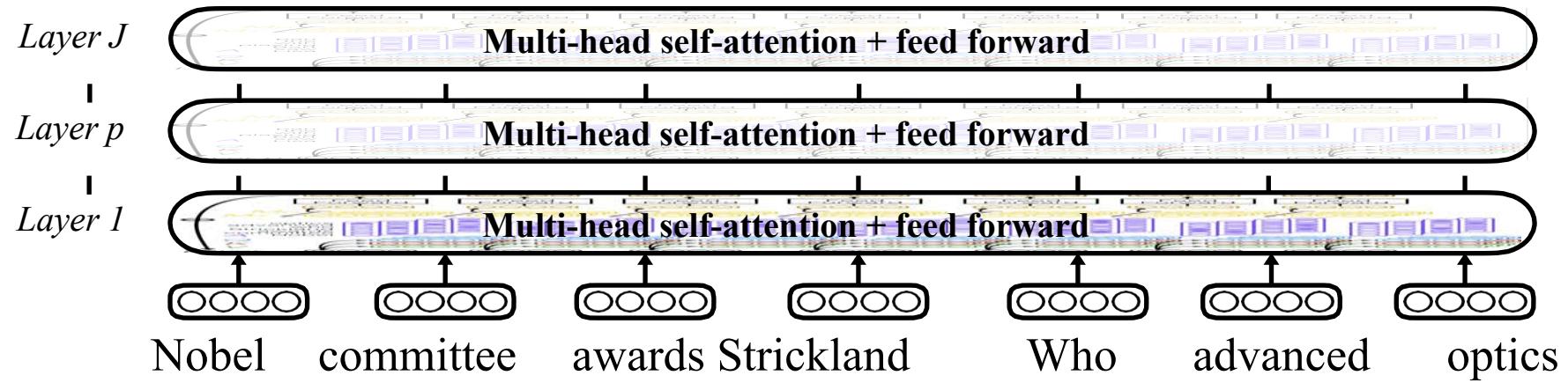
# Multi-head self-attention



# Multi-head self-attention

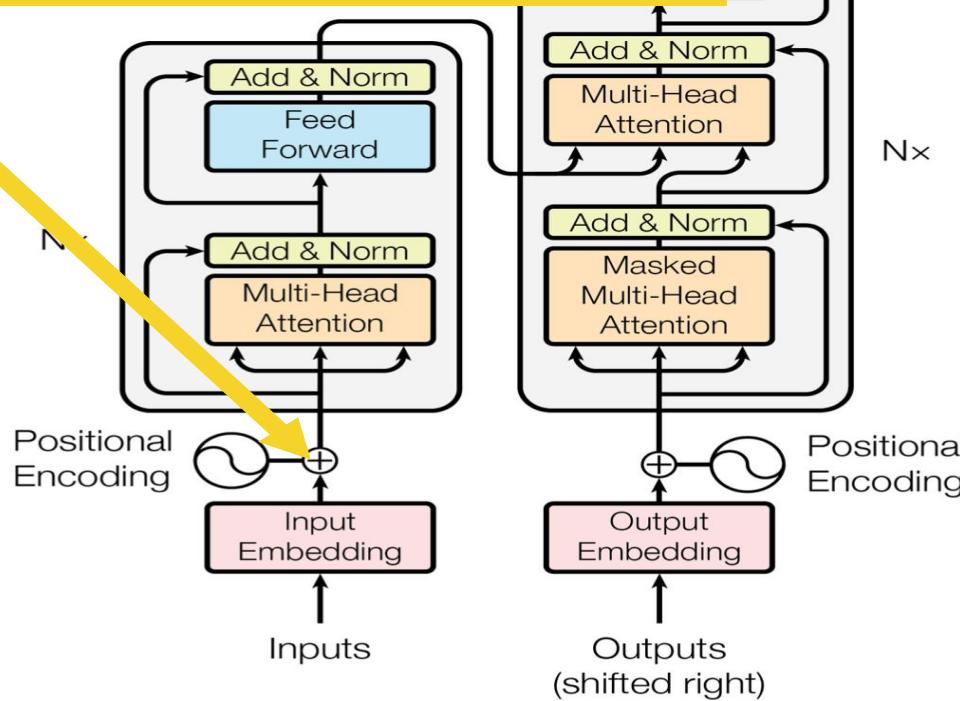


# Multi-head self-attention

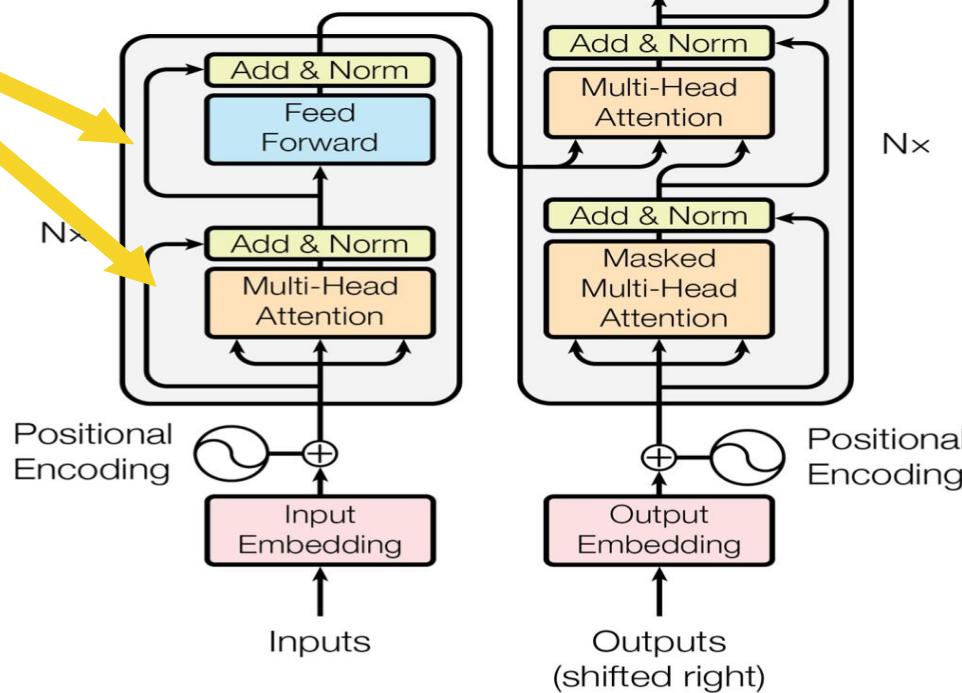


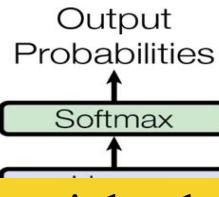
Position embeddings are *added* to each word embedding.

Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!

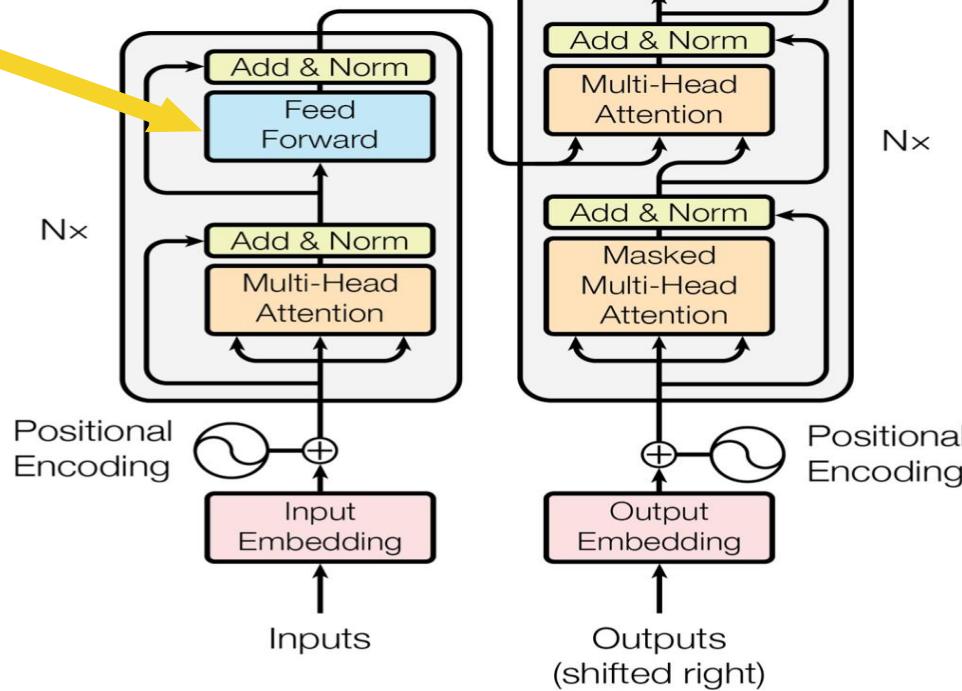


*Residual connections*, which mean that we add the input to a particular block to its output, help improve gradient flow

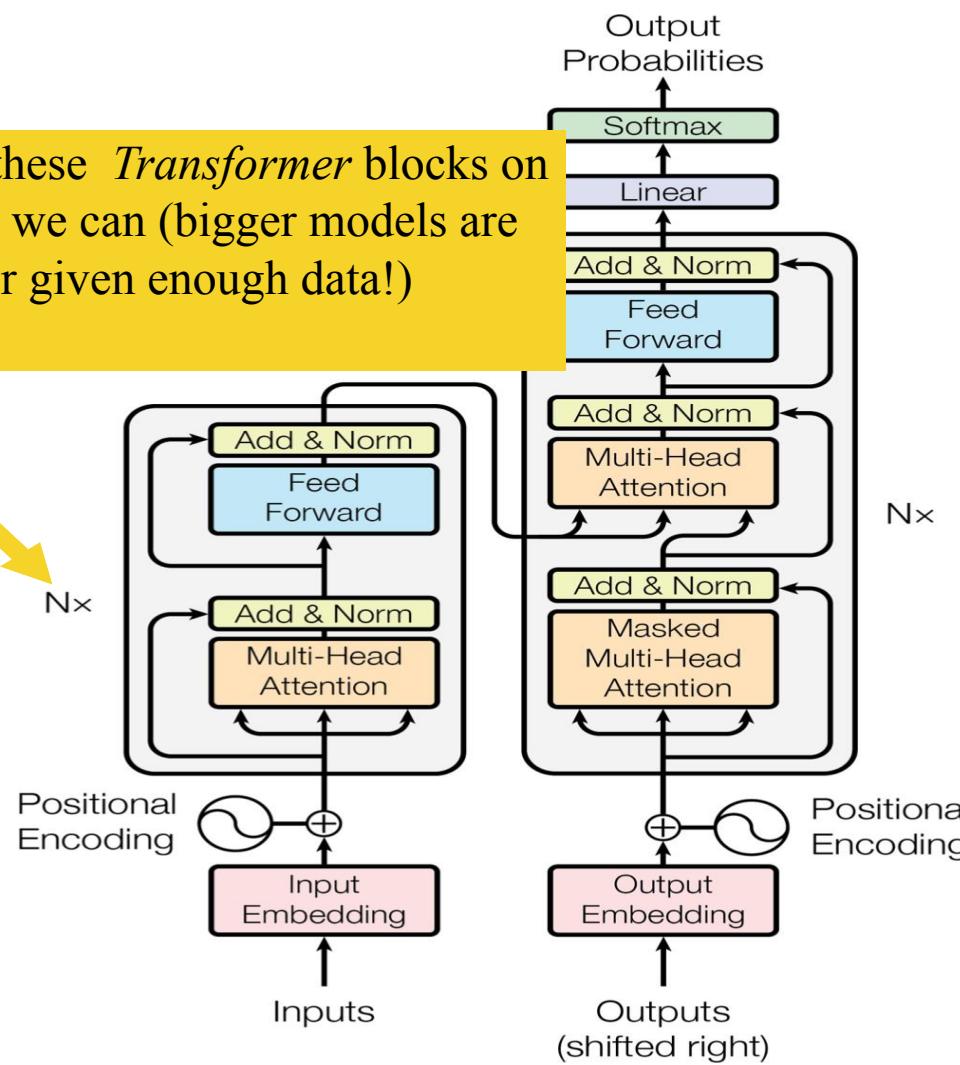




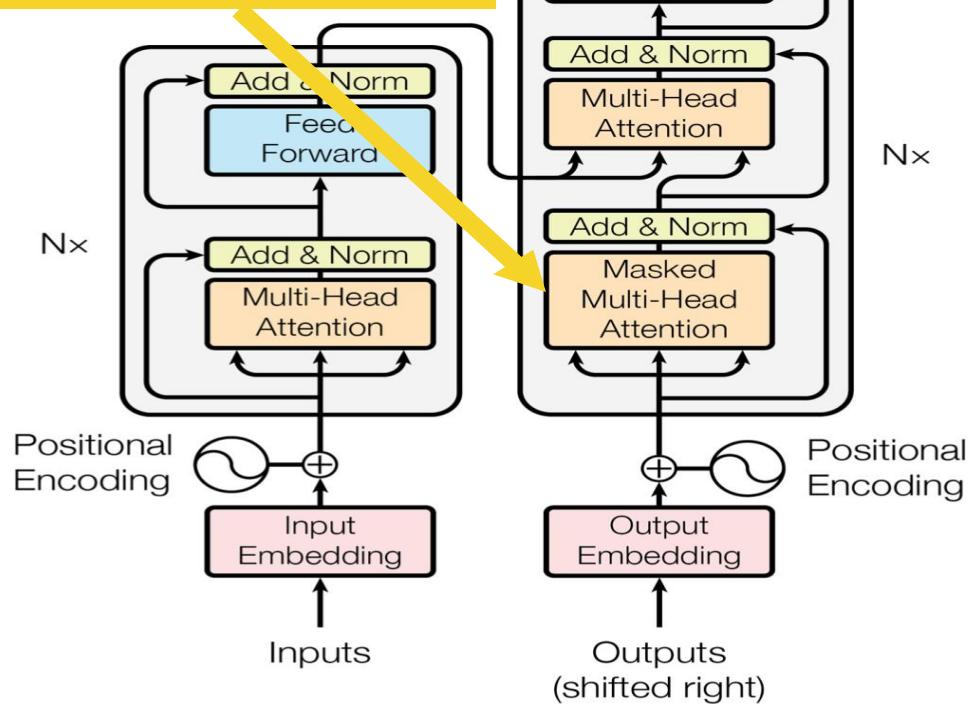
A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity



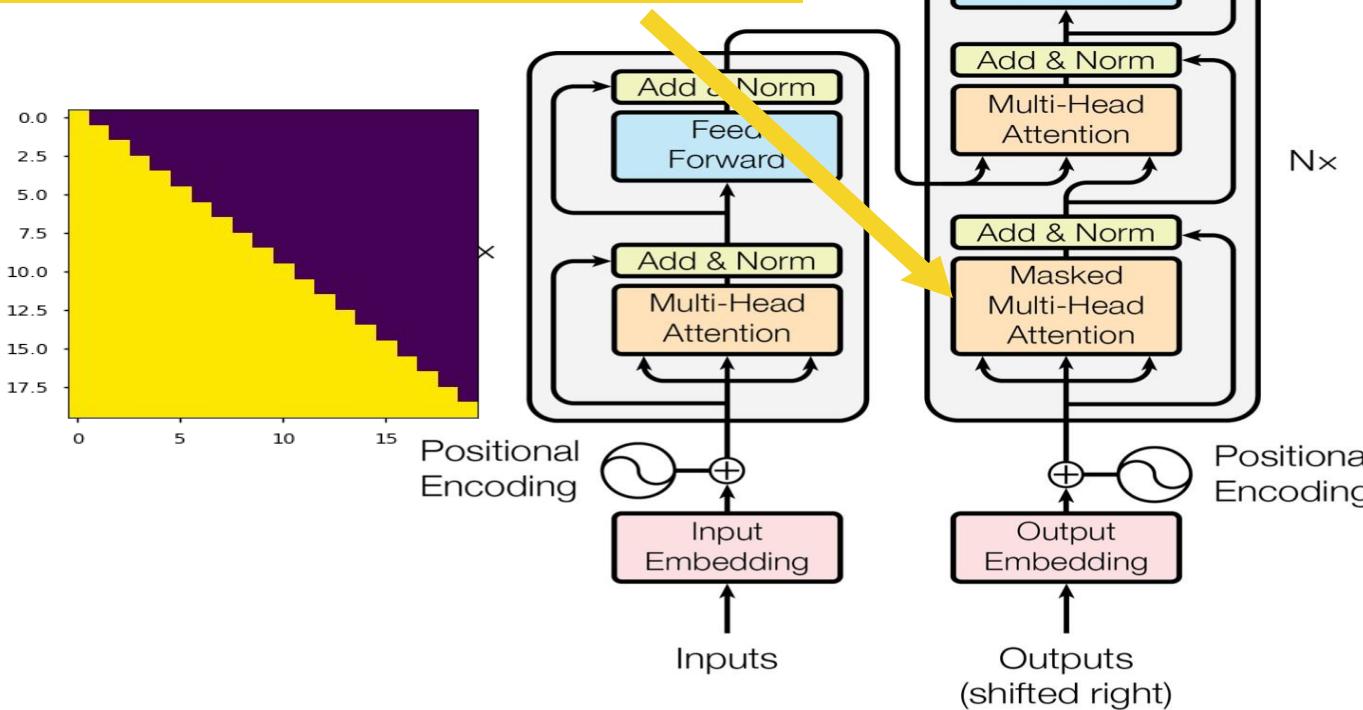
We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!)



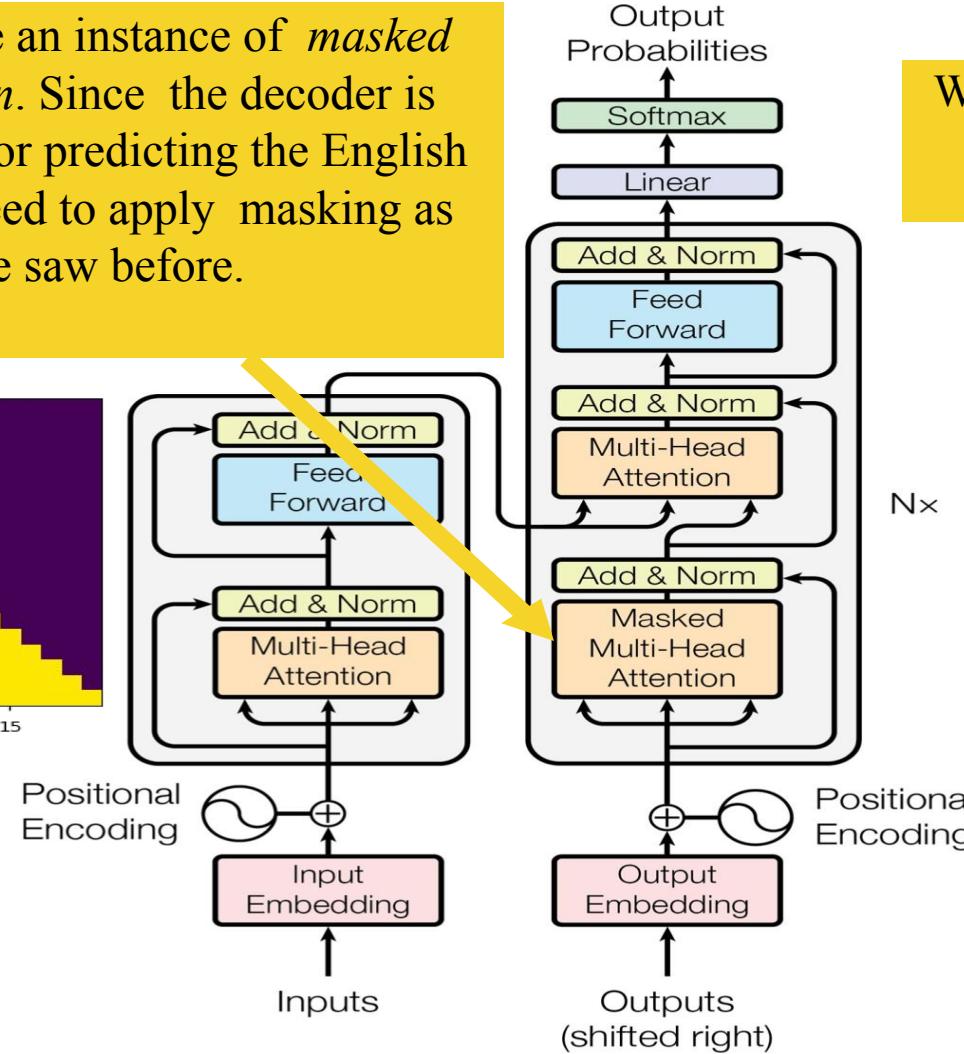
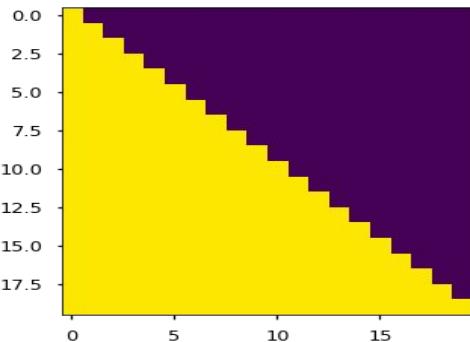
Moving onto the decoder, which takes in English sequences that have been shifted to the right (e.g., *<START> schools opened their*)



We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

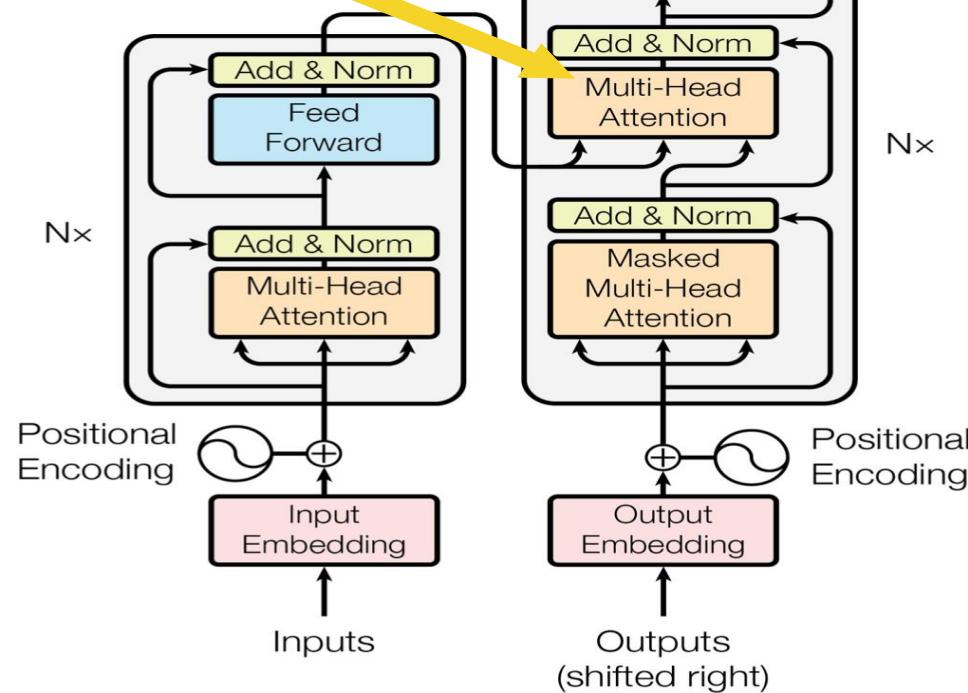


We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

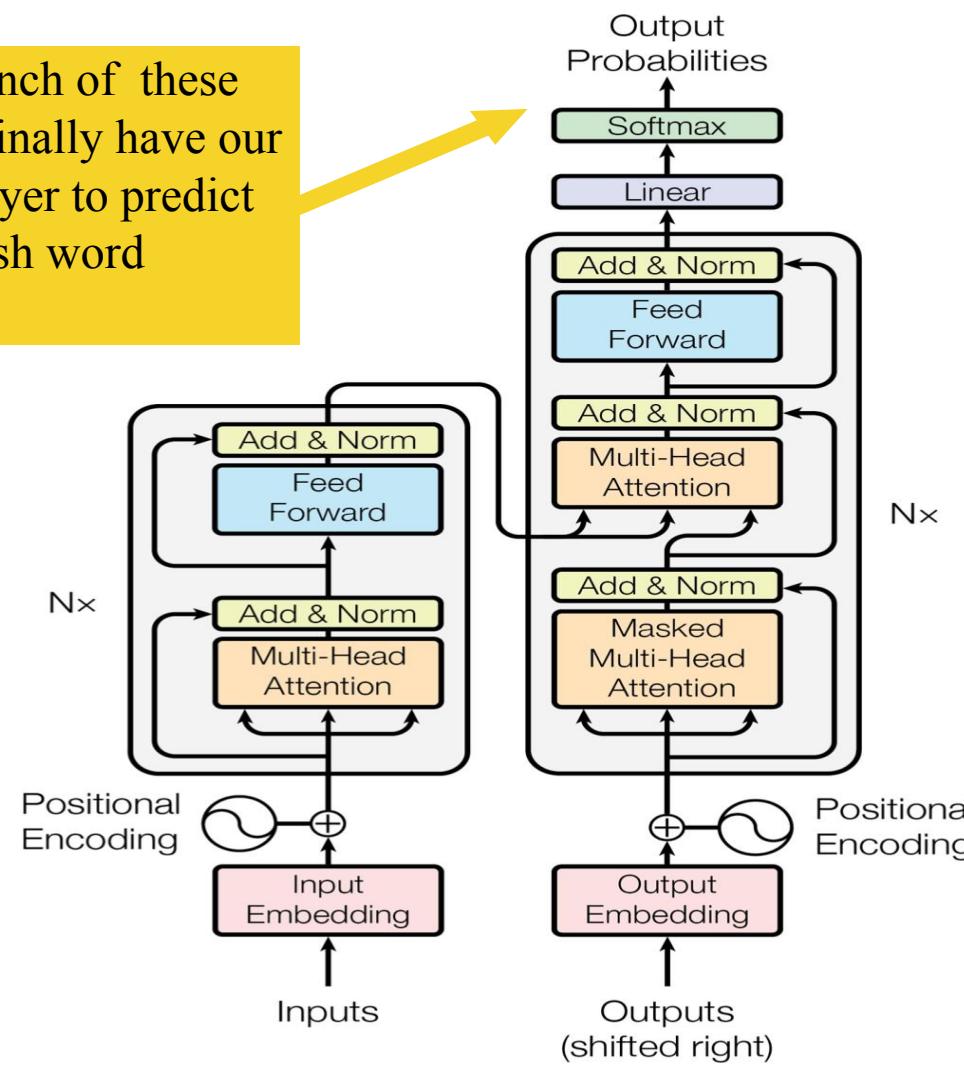


Why don't we do masked self-attention in the encoder?

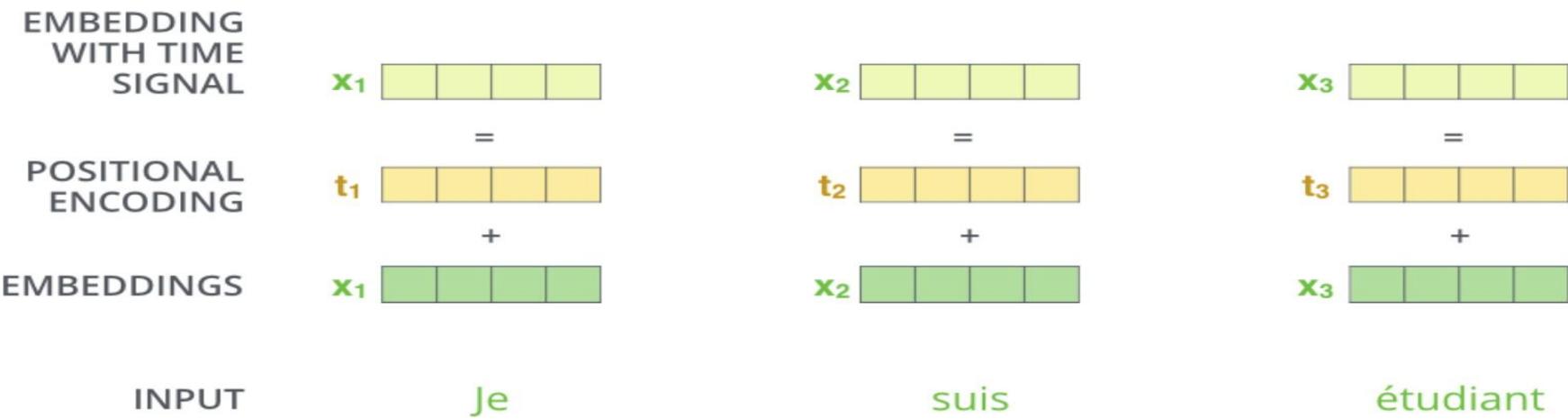
Now, we have *cross attention*, which connects the decoder to the encoder by enabling it to attend over the encoder's final hidden states.



After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word



# Positional encoding



# Intuitive example

0 :	○	○	○	○	8 :	1	○	○	○
1 :	○	○	○	1	9 :	1	○	○	1
2 :	○	○	1	○	10 :	1	○	1	○
3 :	○	○	1	1	11 :	1	○	1	1
4 :	○	1	○	○	12 :	1	1	○	○
5 :	○	1	○	1	13 :	1	1	○	1
6 :	○	1	1	○	14 :	1	1	1	○
7 :	○	1	1	1	15 :	1	1	1	1

# Transformer positional encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Positional encoding is a 512d vector

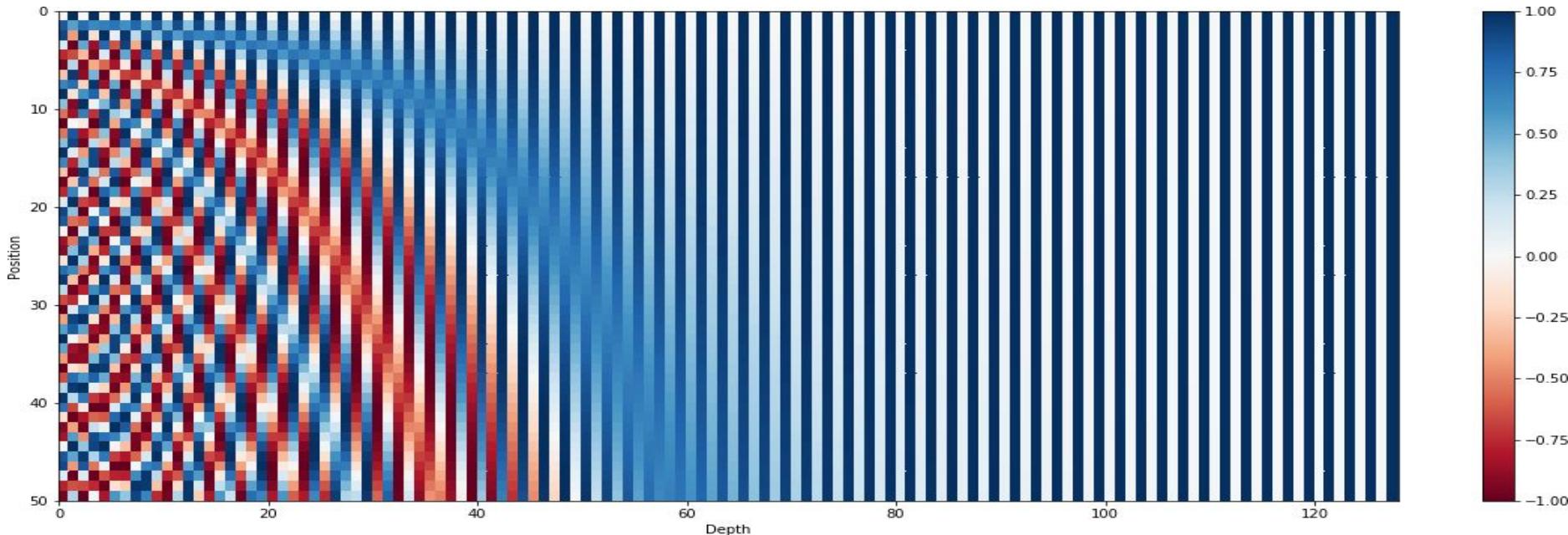
$i$  = a particular dimension of this vector

$pos$  = dimension of the word

$d_{model} = 512$

# What does this look like?

*(each row is the pos. emb. of a 50-word sentence)*

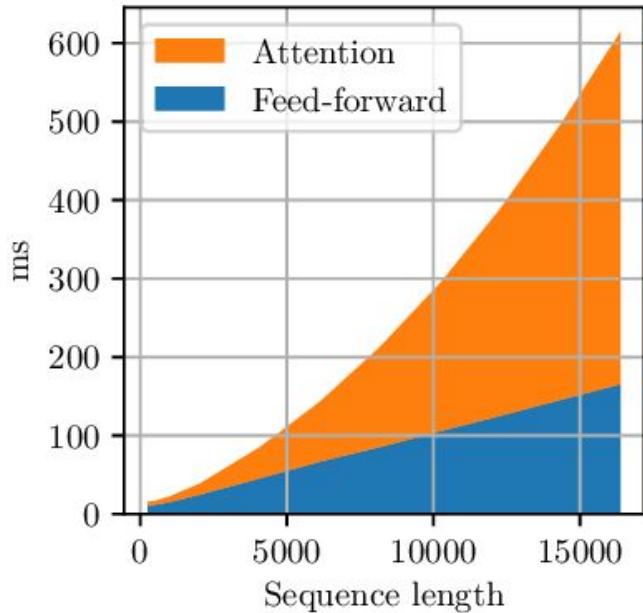


# More on new-Transformer

What would we like to fix about the Transformer?

### Quadratic compute in self-attention (today):

- Computing all pairs of interactions means our computation grows quadratically with the sequence length!
- For recurrent models, it only grew linearly!



# Quadratic computation as a function of sequence length

- One of the benefits of self-attention over recurrence was that it's highly parallelizable.
- However, its total number of operations grows as  $O(n^2d)$ , where  $n$  is the sequence length, and  $d$  is the dimensionality.

$$XQ \begin{matrix} \\ K^\top X^\top \end{matrix} = XQK^\top X^\top \in \mathbb{R}^{n \times n}$$

Need to compute all pairs of interactions!  
 $O(n^2d)$

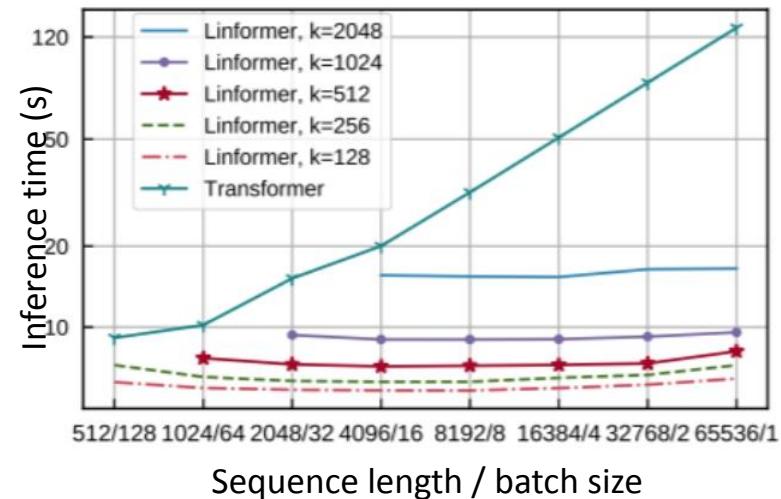
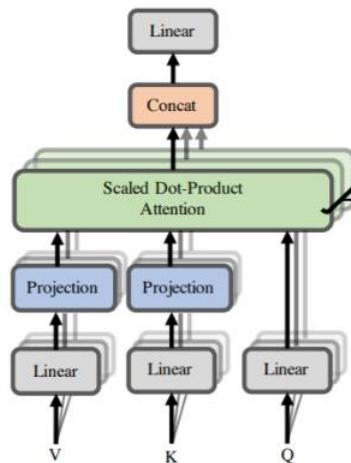
- Think of  $d$  as around **1,000** (though for large language models it's much larger!).
  - So, for a single (shortish) sentence,  $n \leq 30$ ;  $n^2 \leq 900$ .
  - In practice, we set a bound like  $n = 512$ .
  - **But what if we'd like  $n \geq 50,000$ ?** For example, to work on long documents?

# Work on improving on quadratic self-attention cost

Considerable recent work has gone into the question, *Can we build models like Transformers without paying the all-pairs self-attention cost?*  
For example, Linformer [Wang et al., 2020]

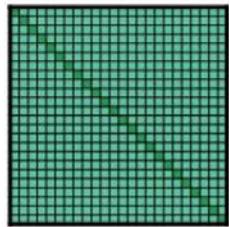
## Key Idea:

- Linformer introduces a novel concept called "compressed" or "linearized" self-attention.
- Instead of computing attention scores for all pairs of input elements, it employs linear projections to reduce the complexity.

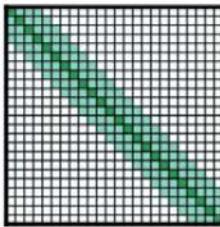


# Example: Longformer / Big Bird

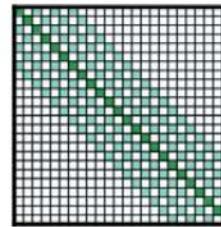
Key idea: use sparse attention patterns!



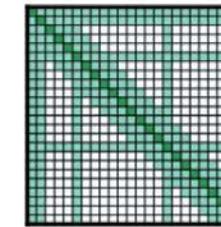
(a) Full  $n^2$  attention



(b) Sliding window attention

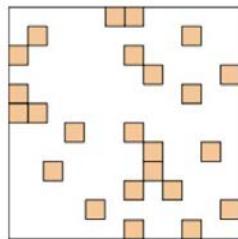


(c) Dilated sliding window

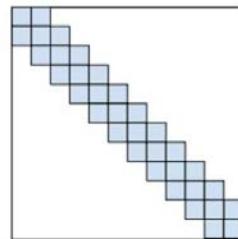


(d) Global+sliding window

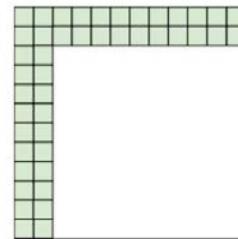
(Beltagy et al., 2020): Longformer: The Long-Document Transformer



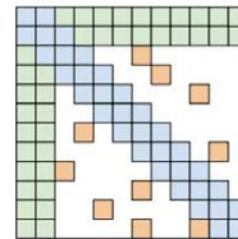
(a) Random attention



(b) Window attention



(c) Global Attention



(d) BIGBIRD

(Zaheer et al., 2021): Big Bird: Transformers for Longer

# Do we even need to remove the quadratic cost of attention?

- **As Transformers Scale Up:** When Transformers are scaled to larger sizes, an increasingly significant portion of computational resources is allocated to tasks outside of the self-attention mechanism, despite its quadratic computational cost.
- **Current Practice:** In practice, nearly all large Transformer-based language models continue to rely on the traditional quadratic-cost attention mechanism that has been presented.
- **Challenges with Cost-Efficiency:** Alternative, more computationally efficient methods often do not perform as effectively when applied at a large scale.
- **Exploring Cheaper Alternatives:** Is there value in exploring cost-efficient alternatives to self-attention, or could we unlock the potential for significantly improved models with much longer contextual information (e.g., >100k tokens) if we find the right approach?

# Do Transformer Modifications Transfer?

- "Surprisingly, we find that most modifications do not meaningfully improve performance."

Model	Params	Ops	Step/s	Early loss	Final loss	SGLUE	XSum	WebQ	WMT EnDe
Vanilla Transformer	223M	11.3T	3.50	2.183 ± 0.005	1.838	71.66	17.78	33.02	26.62
GeLU	223M	11.3T	3.58	2.179 ± 0.003	1.838	<b>75.79</b>	<b>17.86</b>	<b>25.13</b>	36.47
Swish	223M	11.3T	3.62	2.186 ± 0.003	1.847	73.77	17.74	<b>24.84</b>	26.75
ELU	223M	11.3T	3.56	2.379 ± 0.007	1.932	67.83	16.73	33.02	36.06
GLU	223M	11.3T	3.59	2.174 ± 0.003	1.814	74.20	17.42	24.34	37.12
GeGLU	223M	11.3T	3.58	2.183 ± 0.003	1.792	75.96	18.27	<b>24.87</b>	36.47
ReLU	223M	11.3T	3.57	2.183 ± 0.004	1.808	76.17	17.50	24.87	37.02
SeLU	223M	11.3T	3.55	2.313 ± 0.004	1.948	68.76	16.76	22.75	25.89
SwGLU	223M	11.3T	3.53	2.127 ± 0.003	1.788	76.00	<b>18.20</b>	<b>24.84</b>	<b>27.02</b>
LiGLU	223M	11.3T	3.58	3.149 ± 0.005	1.798	<b>75.34</b>	17.97	<b>24.84</b>	36.53
Sigmoid	223M	11.3T	3.63	2.291 ± 0.019	1.867	74.31	17.51	23.93	36.30
Softplus	223M	11.3T	3.47	2.307 ± 0.011	1.800	72.45	17.65	<b>24.84</b>	<b>26.89</b>
HuB Norm	223M	11.3T	4.66	2.187 ± 0.003	1.838	<b>75.45</b>	<b>17.86</b>	<b>24.07</b>	37.14
ResNet	223M	11.3T	3.51	2.182 ± 0.003	1.809	61.59	16.64	26.15	36.17
Rosero + LayerNorm	223M	11.3T	3.36	2.223 ± 0.003	1.854	70.42	17.58	23.02	36.29
Rosero + HuB Norm	223M	11.3T	3.34	2.221 ± 0.003	1.875	70.33	17.32	23.02	36.19
Fixup	223M	11.3T	2.95	2.382 ± 0.012	2.067	58.96	14.42	23.02	36.31
24 layers, $d_h = 1236$ , $H = 6$	224M	11.3T	3.33	2.200 ± 0.007	1.843	<b>74.89</b>	17.75	<b>25.13</b>	<b>26.89</b>
18 layers, $d_h = 2048$ , $H = 8$	223M	11.3T	3.38	2.185 ± 0.005	1.833	<b>76.45</b>	16.83	<b>24.84</b>	<b>27.10</b>
8 layers, $d_h = 4096$ , $H = 18$	223M	11.3T	3.69	2.190 ± 0.005	1.847	<b>74.58</b>	17.69	<b>23.28</b>	<b>26.85</b>
4 layers, $d_h = 8144$ , $H = 24$	223M	11.3T	3.70	2.301 ± 0.010	1.857	<b>73.35</b>	17.39	<b>24.80</b>	<b>26.76</b>
Block sharding	65M	11.3T	3.91	2.497 ± 0.007	2.164	64.90	14.53	21.96	25.48
+ Factorized embeddings	65M	8.4T	4.21	2.631 ± 0.305	2.183	60.84	14.09	19.84	25.27
+ Factorized & shared embeddings	263M	8.1T	4.37	2.307 ± 0.313	2.383	53.95	11.37	19.84	25.19
Encoder only block sharding	170M	11.3T	3.68	2.298 ± 0.023	1.929	69.60	16.33	23.02	36.23
Decoder only block sharding	144M	11.3T	3.70	2.352 ± 0.029	2.082	67.93	16.13	<b>23.81</b>	36.08
Factorized Embedding	227M	9.4T	3.80	2.208 ± 0.006	1.855	70.41	15.92	22.75	36.50
Factorized & shared embedding	202M	9.1T	3.92	2.320 ± 0.010	1.952	68.69	16.33	22.22	36.44
Text encoder/decoder input embeddings	348M	11.3T	3.55	2.192 ± 0.003	1.840	<b>71.70</b>	17.72	<b>24.84</b>	36.49
Text decoder input and output embeddings	348M	11.3T	3.57	2.187 ± 0.007	<b>1.827</b>	<b>74.86</b>	17.74	<b>24.87</b>	<b>26.67</b>
Adaptive embeddings	273M	11.3T	3.53	2.193 ± 0.005	<b>1.834</b>	<b>72.99</b>	17.58	<b>23.28</b>	36.48
Unified embeddings	304M	9.2T	3.55	2.250 ± 0.002	1.890	66.57	16.21	<b>24.07</b>	<b>26.66</b>
Adaptive softmax	304M	9.2T	3.60	2.364 ± 0.005	1.982	<b>72.91</b>	16.67	21.16	35.36
Adaptive softmax without projections	223M	10.8T	3.43	2.229 ± 0.009	1.914	<b>71.82</b>	17.10	23.02	35.72
Mixture of softmaxes	332M	16.3T	2.24	2.227 ± 0.017	<b>1.821</b>	<b>70.77</b>	17.62	22.75	<b>26.82</b>
Transparent attention	223M	11.3T	3.33	2.181 ± 0.014	1.874	54.31	10.49	21.18	<b>26.80</b>
Dynamic convolution	237M	11.3T	2.65	2.413 ± 0.009	2.047	58.30	12.67	21.16	17.03
Lightweight convolution	224M	10.6T	4.07	2.370 ± 0.010	1.988	63.07	14.86	23.02	24.73
Enforced convolution	217M	9.0T	3.09	2.220 ± 0.003	1.863	<b>73.67</b>	10.76	<b>24.07</b>	36.56
Systemthesizer (dense)	224M	11.3T	3.47	2.314 ± 0.017	1.962	61.03	14.27	16.14	<b>26.43</b>
Systemthesizer (dense plus alpha)	238M	11.3T	3.49	2.310 ± 0.009	1.940	74.89	16.95	23.02	<b>26.41</b>
Systemthesizer (dense plus alpha plus alpha)	243M	12.6T	3.01	2.196 ± 0.007	<b>1.828</b>	74.25	17.02	<b>23.28</b>	36.61
Systemthesizer (factorized)	207M	10.3T	3.94	2.341 ± 0.017	1.968	62.78	15.39	<b>23.55</b>	26.42
Systemthesizer (random)	254M	10.3T	4.08	2.326 ± 0.012	2.009	54.27	10.35	18.56	26.44
Systemthesizer (random plus alpha)	292M	12.6T	3.63	2.189 ± 0.008	1.847	<b>73.32</b>	17.04	<b>24.87</b>	36.43
Systemthesizer (random plus alpha plus alpha)	292M	12.6T	3.42	2.186 ± 0.017	<b>1.828</b>	75.24	17.08	<b>24.68</b>	36.39
Universal Transformer	84M	40.6T	0.88	2.406 ± 0.036	2.053	70.13	14.09	19.05	23.91
Mixture of experts	648M	11.3T	3.20	2.149 ± 0.006	1.788	<b>74.55</b>	<b>18.13</b>	<b>24.08</b>	<b>26.94</b>
Switch Transformer	1100M	11.3T	2.18	2.133 ± 0.006	1.758	<b>75.38</b>	<b>18.02</b>	<b>26.19</b>	<b>26.81</b>
Feature Transformer	223M	11.3T	3.20	2.149 ± 0.009	1.808	67.34	17.50	21.36	25.36
Weighted Transformer	280M	71.0T	0.59	2.378 ± 0.021	1.989	60.04	16.39	23.02	36.30
Product key memory	421M	306.8T	0.25	2.155 ± 0.003	<b>1.798</b>	<b>75.16</b>	17.04	<b>23.55</b>	<b>26.73</b>

## Do Transformer Modifications Transfer Across Implementations and Applications?

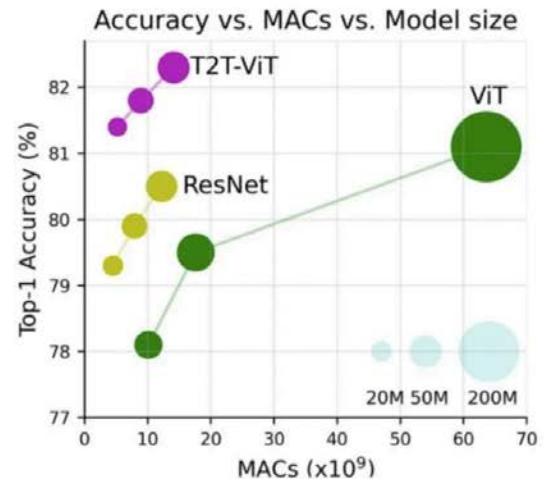
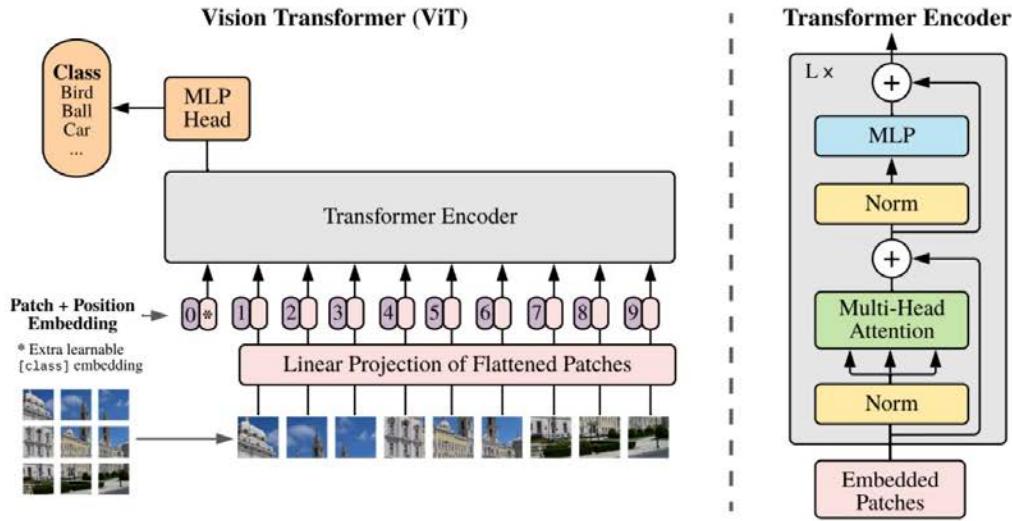
Sharan Narang\* Hyung Won Chung Yi Tay William Fedus

Thibault Fevry† Michael Matena† Karishma Malkhan† Noah Fiedel

Noam Shazeer Zhenzhong Lan† Yanqi Zhou Wei Li

Nan Ding Jake Marcus Adam Roberts Colin Raffel†

# Vision Transformer (ViT)



(Dosovitskiy et al., 2021): An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale