

Overview of LLMs and Transformers

Eric Wallace



Berkeley NLP



BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Berkeley AI Research



Lecture Overview

- State of the field
- Why the task of language modeling?
- Transformer architecture and tokenization
- Large-scale pre-training
- Instruction-tuning and RLHF



Today's NLP Recipe



Today's NLP Recipe



Curate massive pre-training data



Today's NLP Recipe



Curate massive pre-training data



Create fine-tuning data



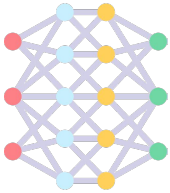
Today's NLP Recipe



Curate massive pre-training data



Create fine-tuning data



Train massive model



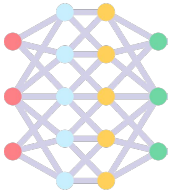
Today's NLP Recipe



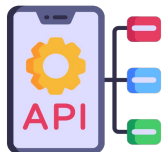
Curate massive pre-training data



Create fine-tuning data



Train massive model



Deploy model widely



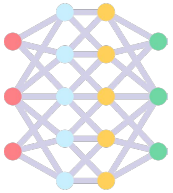
Today's NLP Recipe



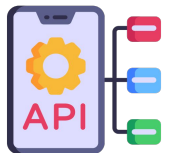
Curate massive pre-training data



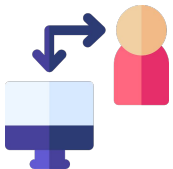
Create fine-tuning data



Train massive model



Deploy model widely



Update using user interactions



Language Modeling

$$p(x_1, x_2, \dots, x_L)$$



Language Modeling

$$p(x_1, x_2, \dots, x_L)$$

$$p(\text{the, mouse, ate, the, cheese}) = 0.001$$

$$p(\text{mouse, the, cheese, ate, the}) = 10^{-20}$$



Language Modeling

$$p(x_1, x_2, \dots, x_L)$$

=

$$\prod_{i=1}^L p(x_i \mid x_{1:i-1})$$



Neural Language Models

$$\prod_{i=1}^L p(x_i \mid x_{1:i-1})$$



Neural Language Models

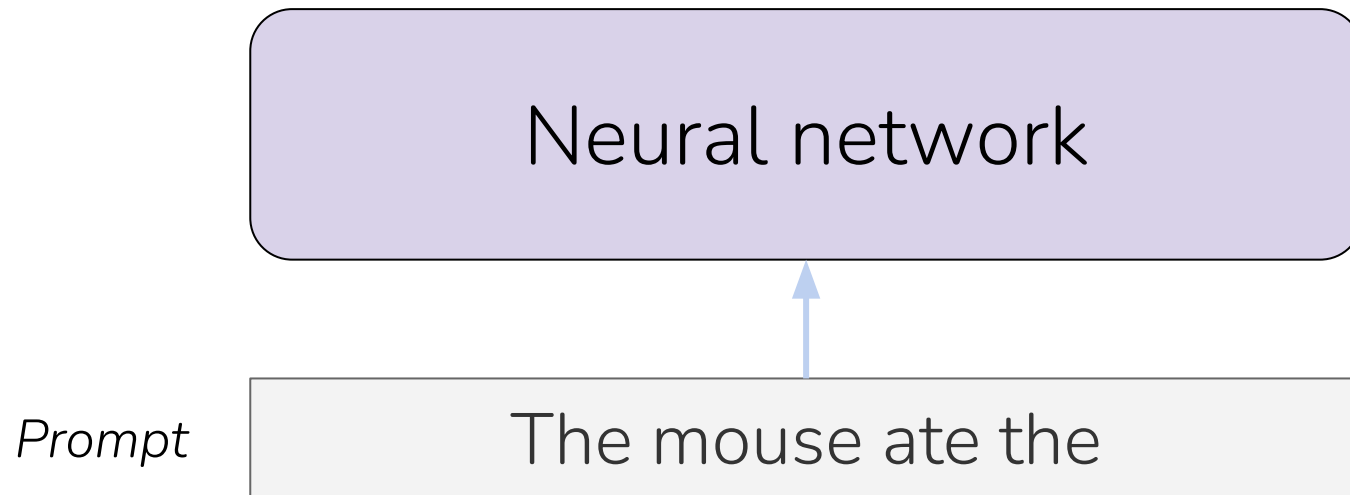
Prompt

The mouse ate the

$$\prod_{i=1}^L p(x_i \mid x_{1:i-1})$$



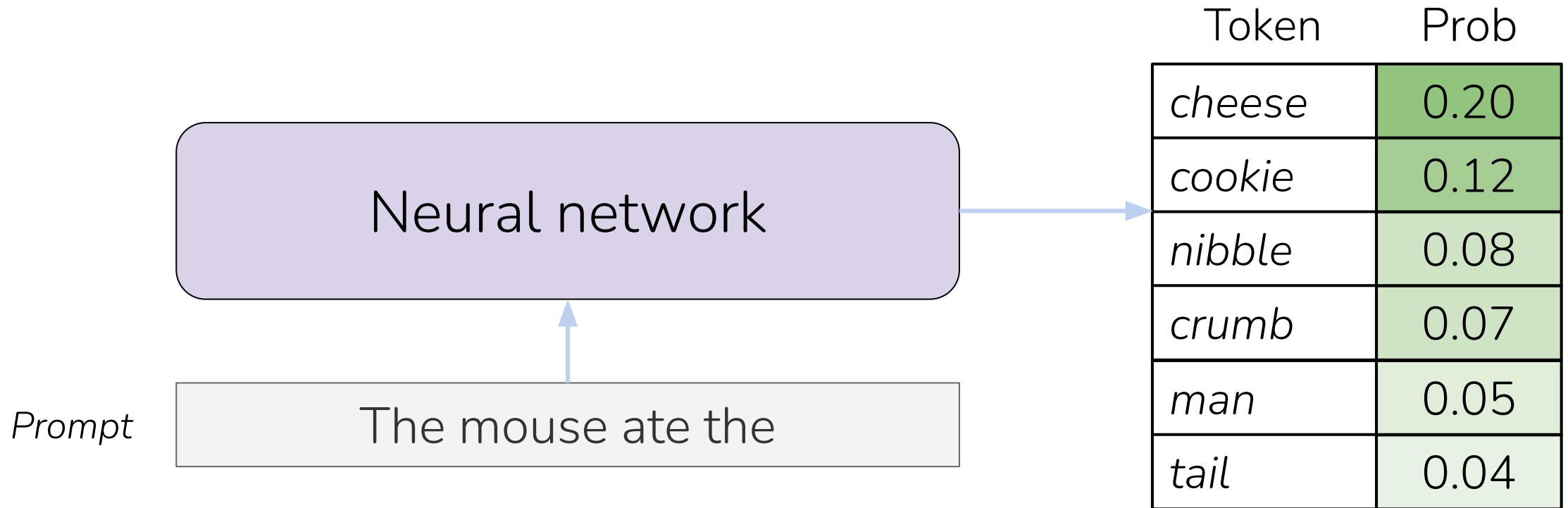
Neural Language Models



$$\prod_{i=1}^L p(x_i | x_{1:i-1})$$



Neural Language Models



$$\prod_{i=1}^L p(x_i | x_{1:i-1})$$



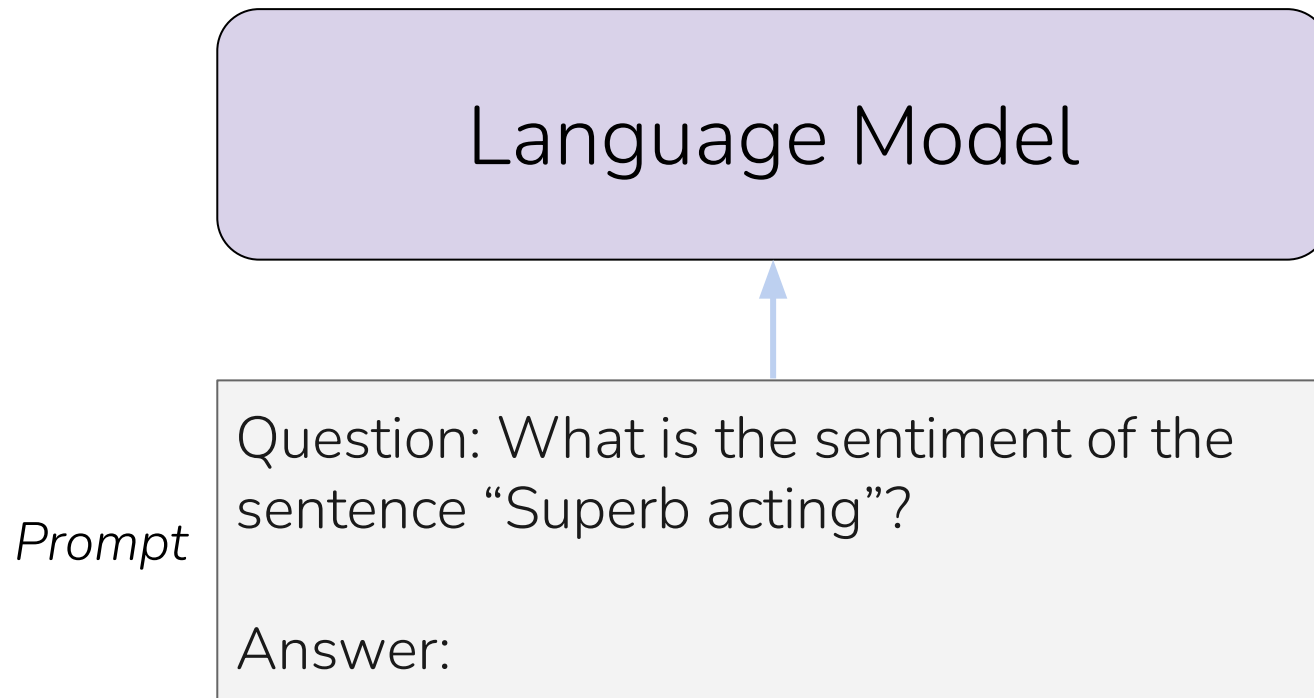
Why Language Modeling?

- Zero- and few-shot learning with language models



Why Language Modeling?

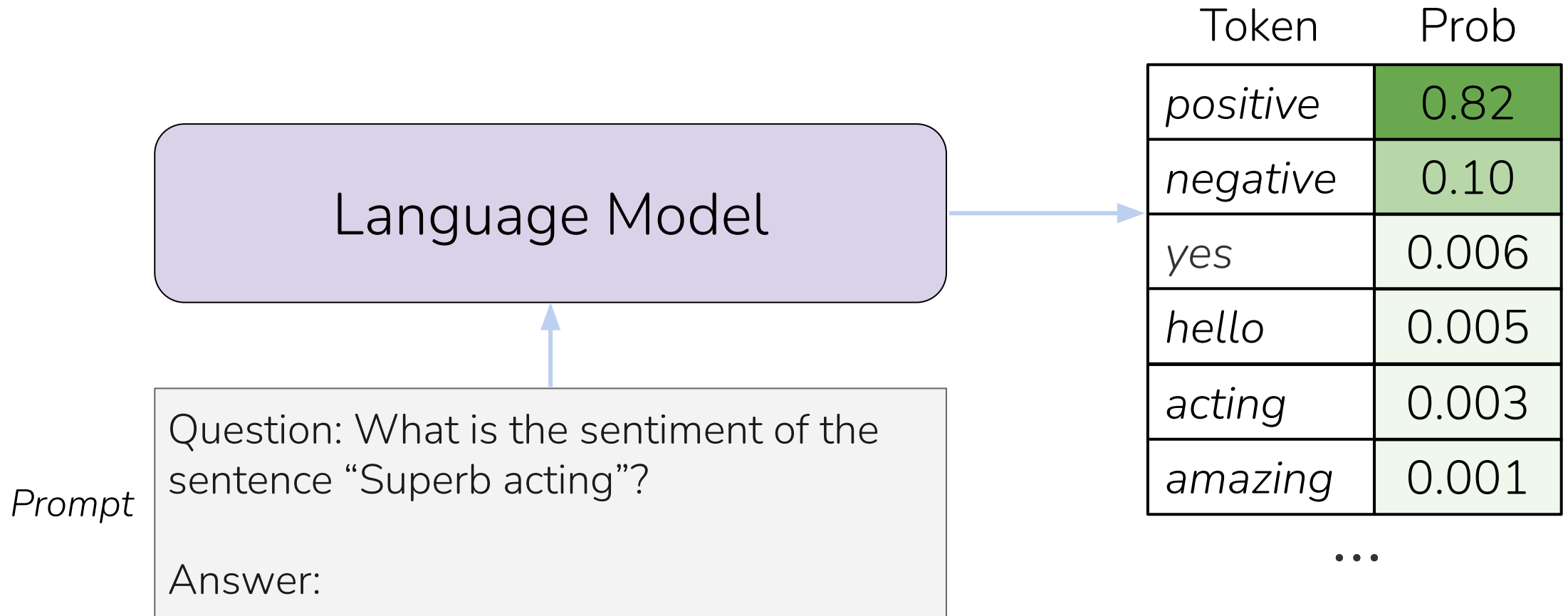
- Zero- and few-shot learning with language models





Why Language Modeling?

- Zero- and few-shot learning with language models





Why Language Modeling?

- “Oracle” LM can perform any task



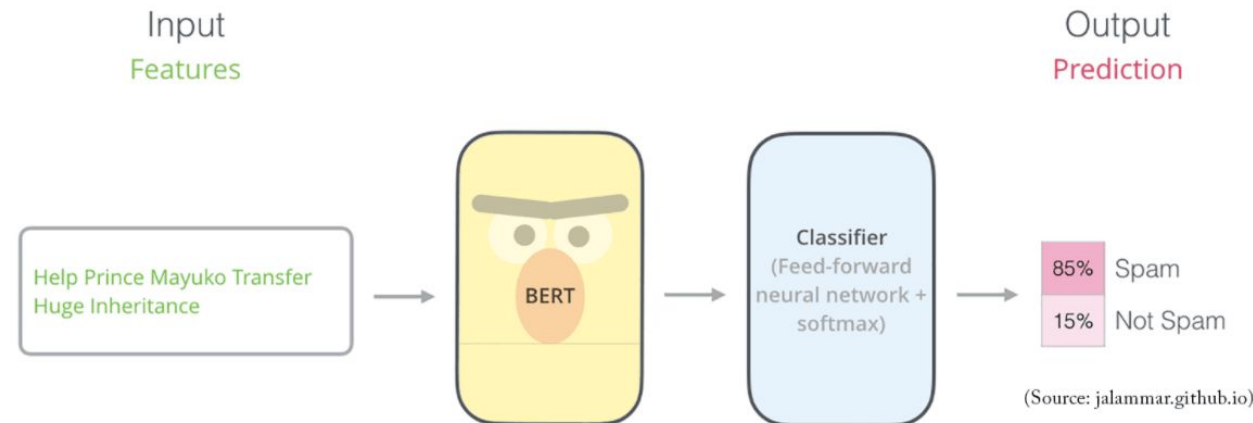
Why Language Modeling?

- “Oracle” LM can perform any task
- Language modeling leads to rich representations
 - George Washington was born in the year _____
 - If it is raining, you may need an _____
 - Using the power rule, the derivative of $3x^5$ is _____



Why Language Modeling?

- “Oracle” LM can perform any task
- Language modeling leads to rich representations
 - George Washington was born in the year _____
 - If it is raining, you may need an _____
 - Using the power rule, the derivative of $3x^5$ is _____





Why Language Modeling?

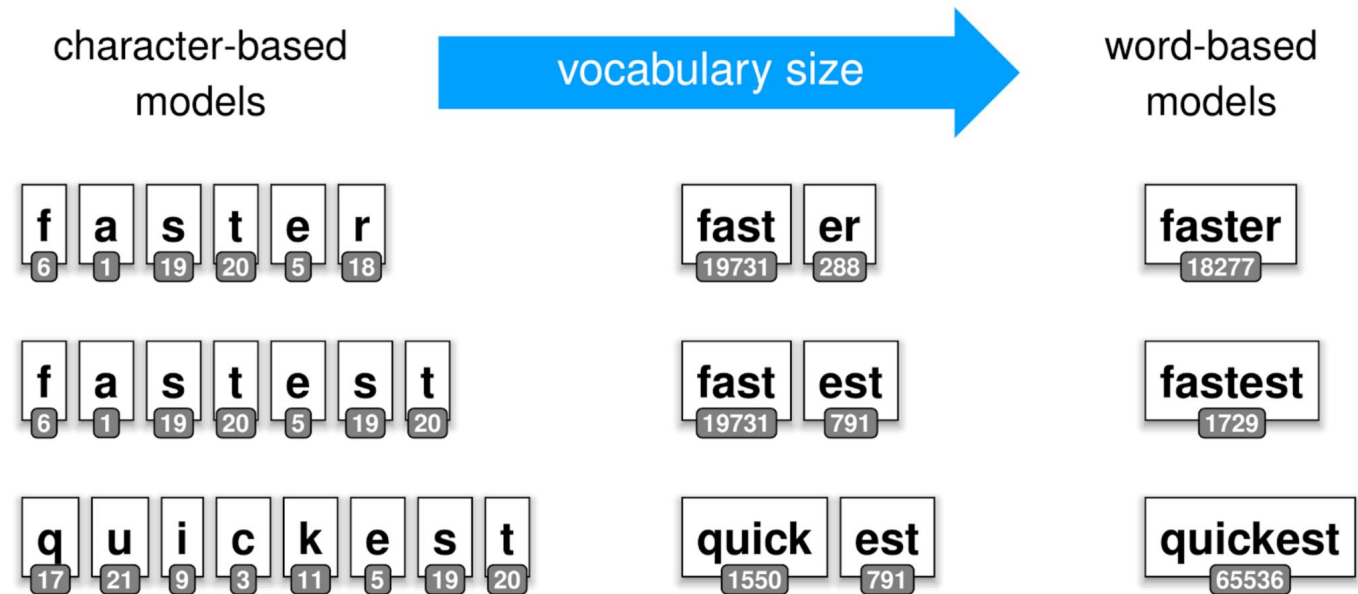
- “Oracle” LM can perform any task
- Language modeling leads to rich representations
 - George Washington was born in the year _____
 - If it is raining, you may need an _____
 - Using the power rule, the derivative of $3x^5$ is _____
- There is effectively “unlimited” data for language modeling
 - enables powerful function approximators (large transformers)

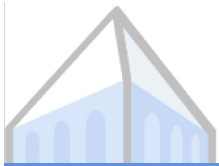


Neural LMs from Scratch

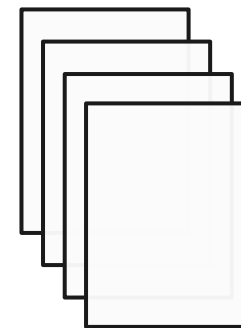
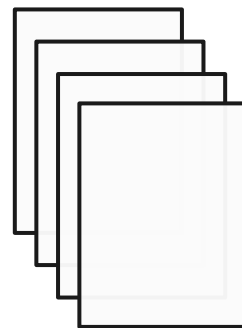
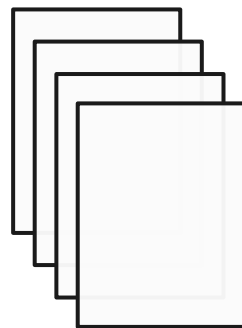
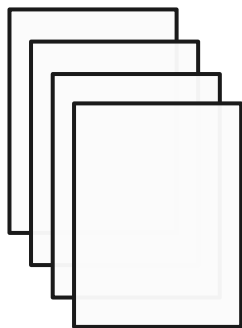
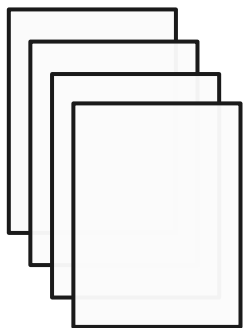


Tokenization



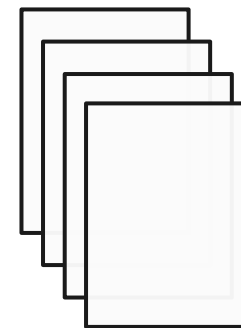
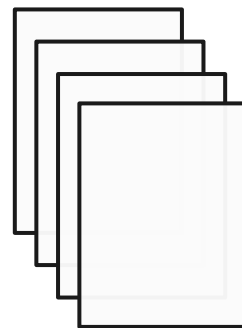
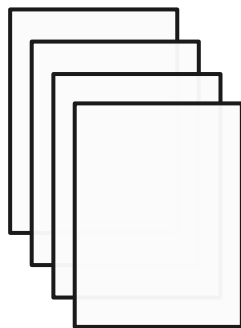
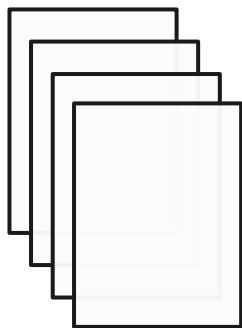
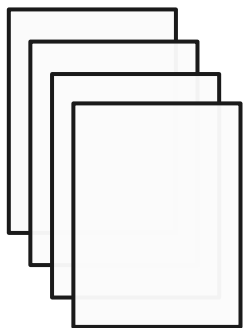


Data for Pre-training





Data for Pre-training

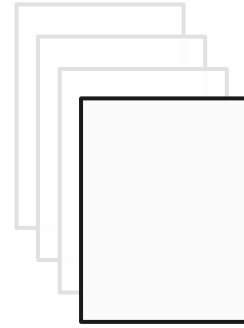
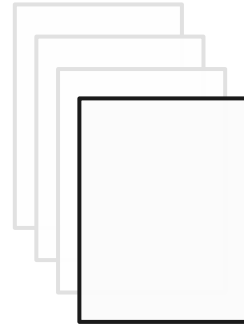
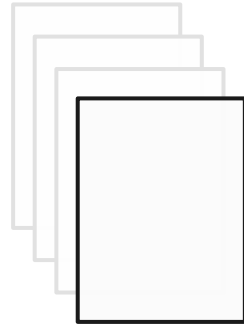
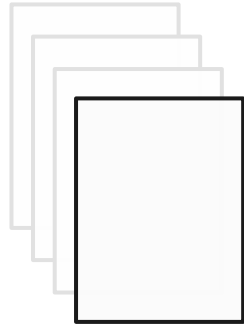
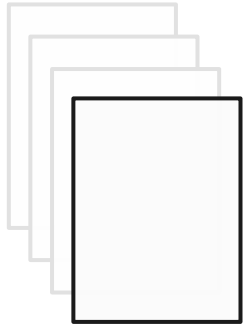


Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

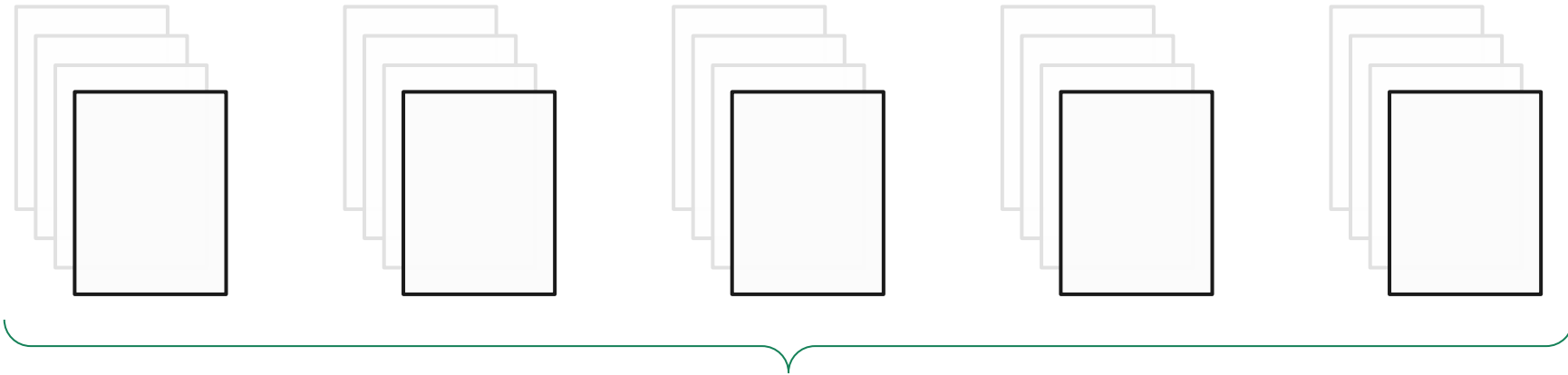


Data for Pre-training





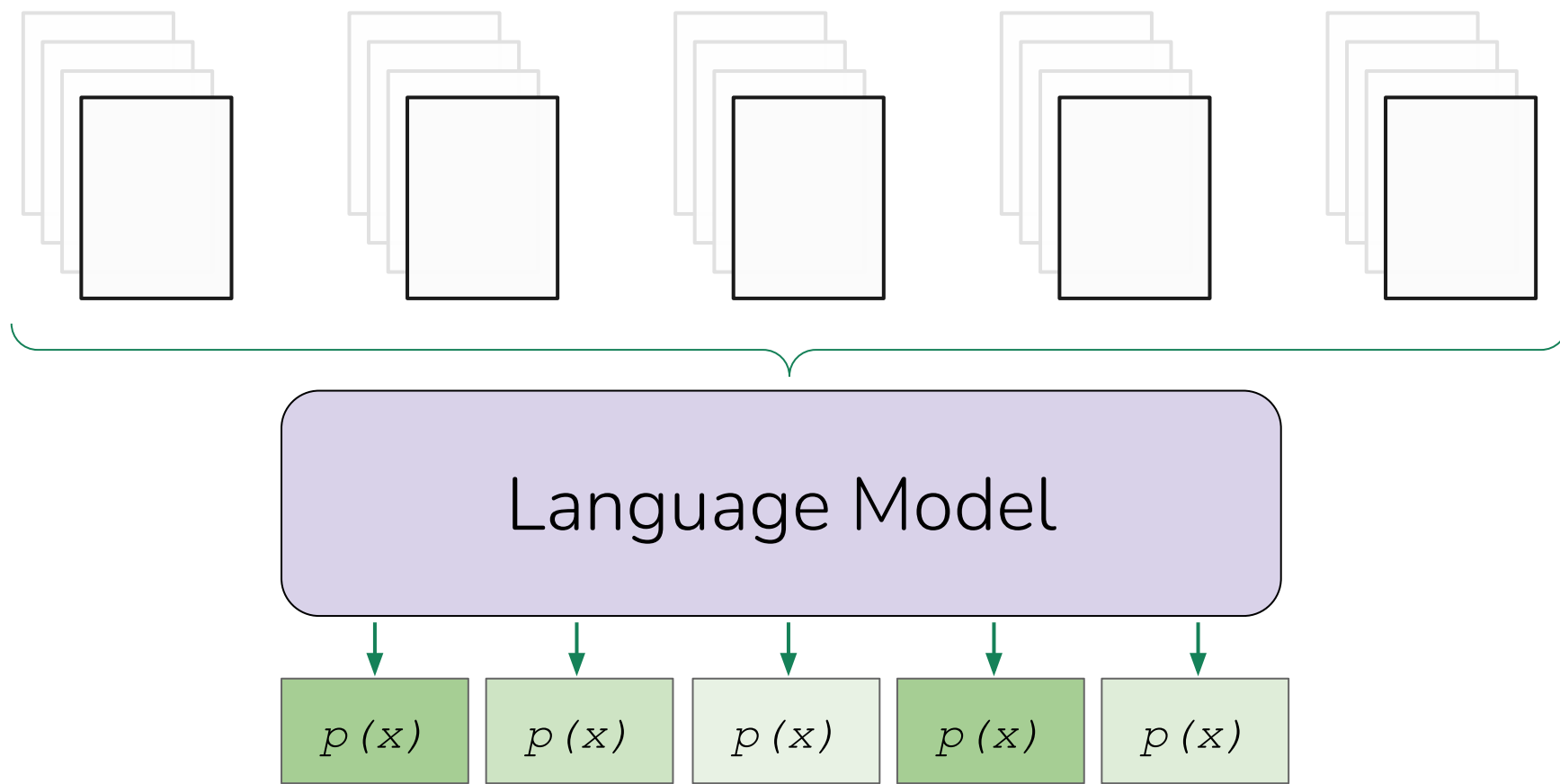
Data for Pre-training



Language Model

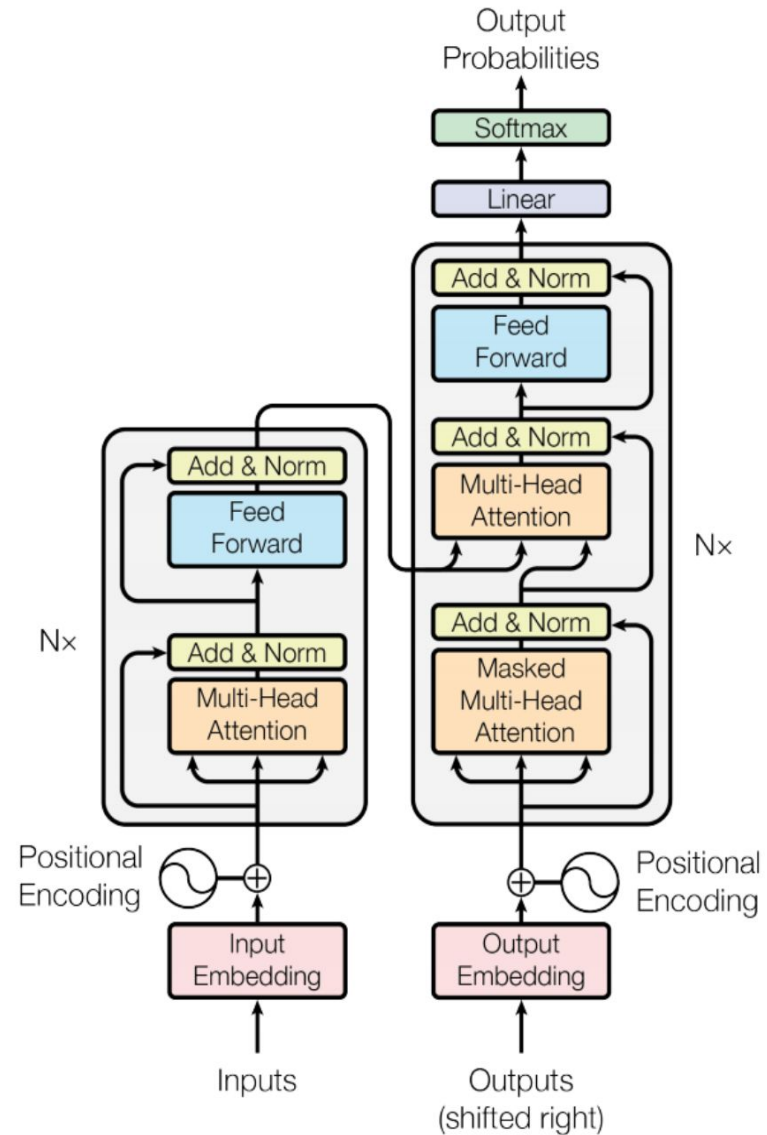


Data for Pre-training



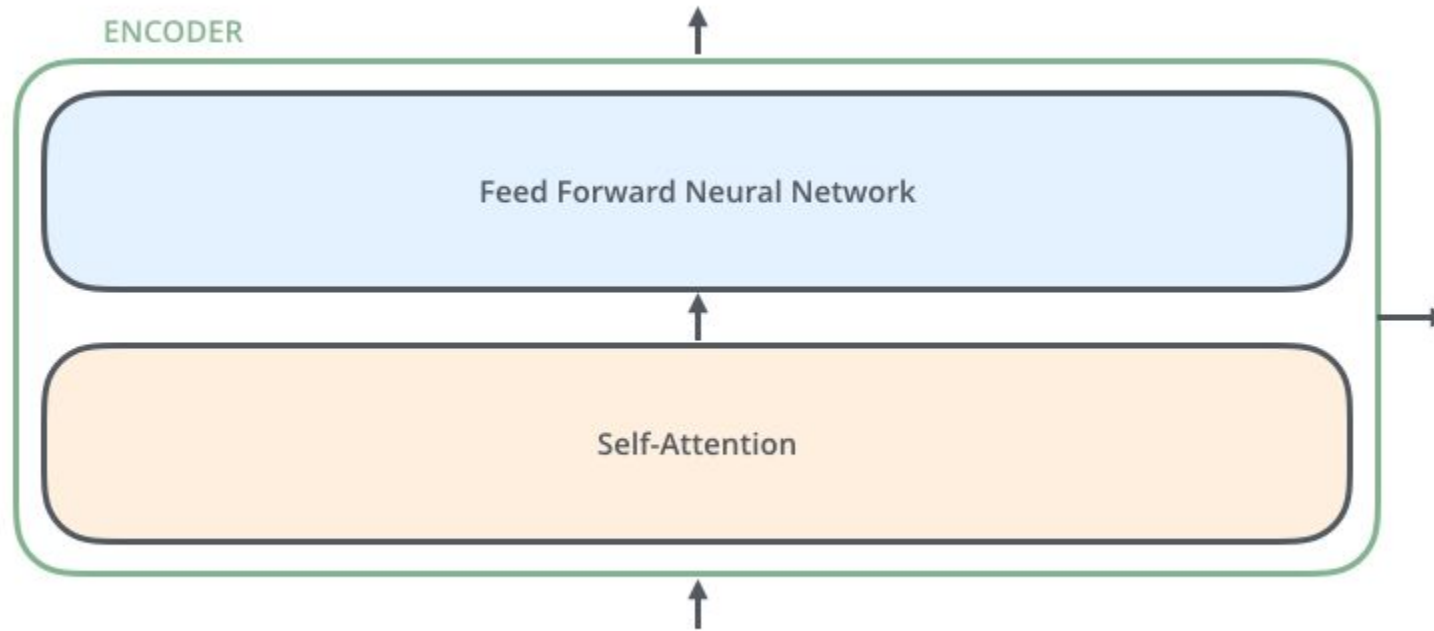


Transformer Architecture



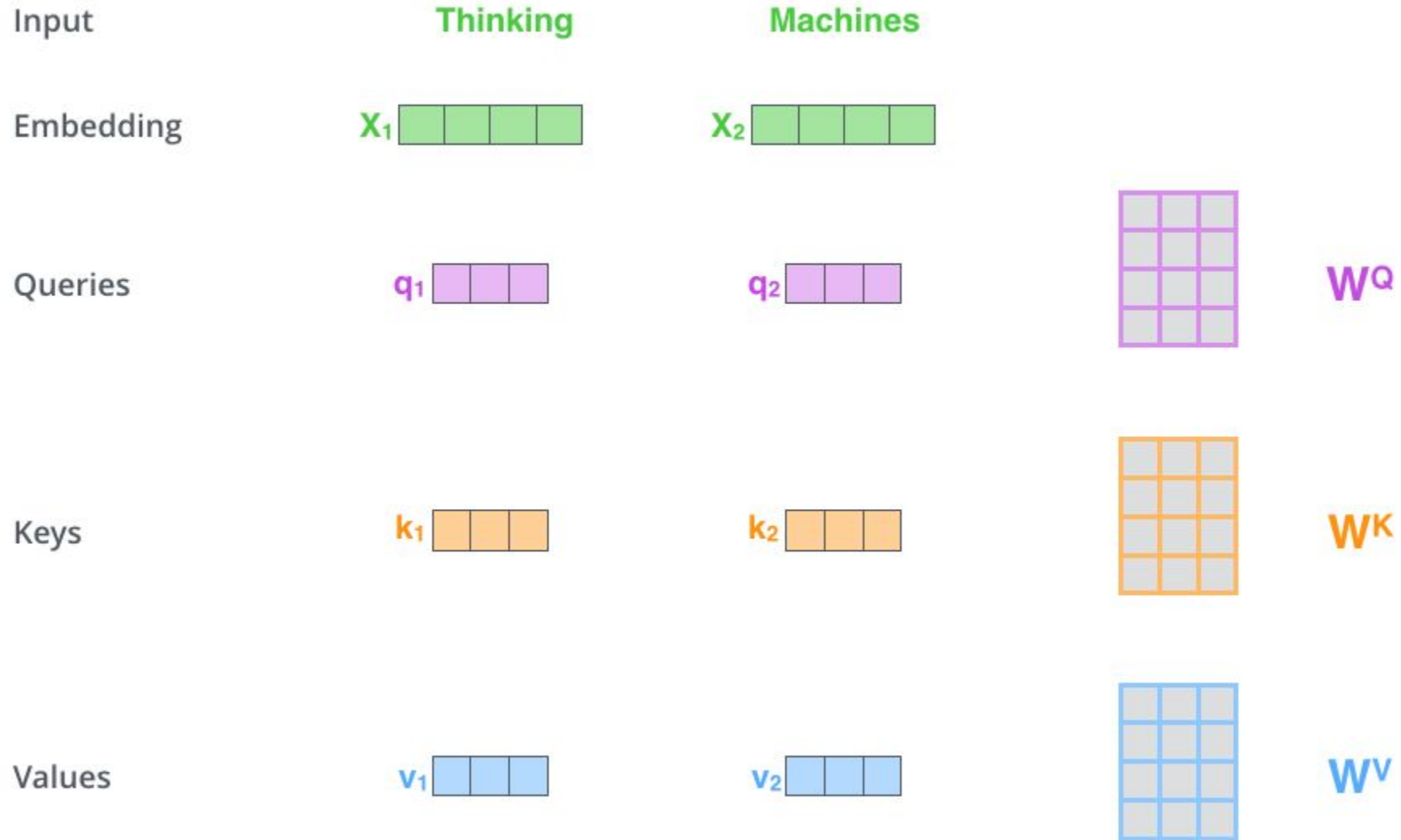


Transformer Architecture



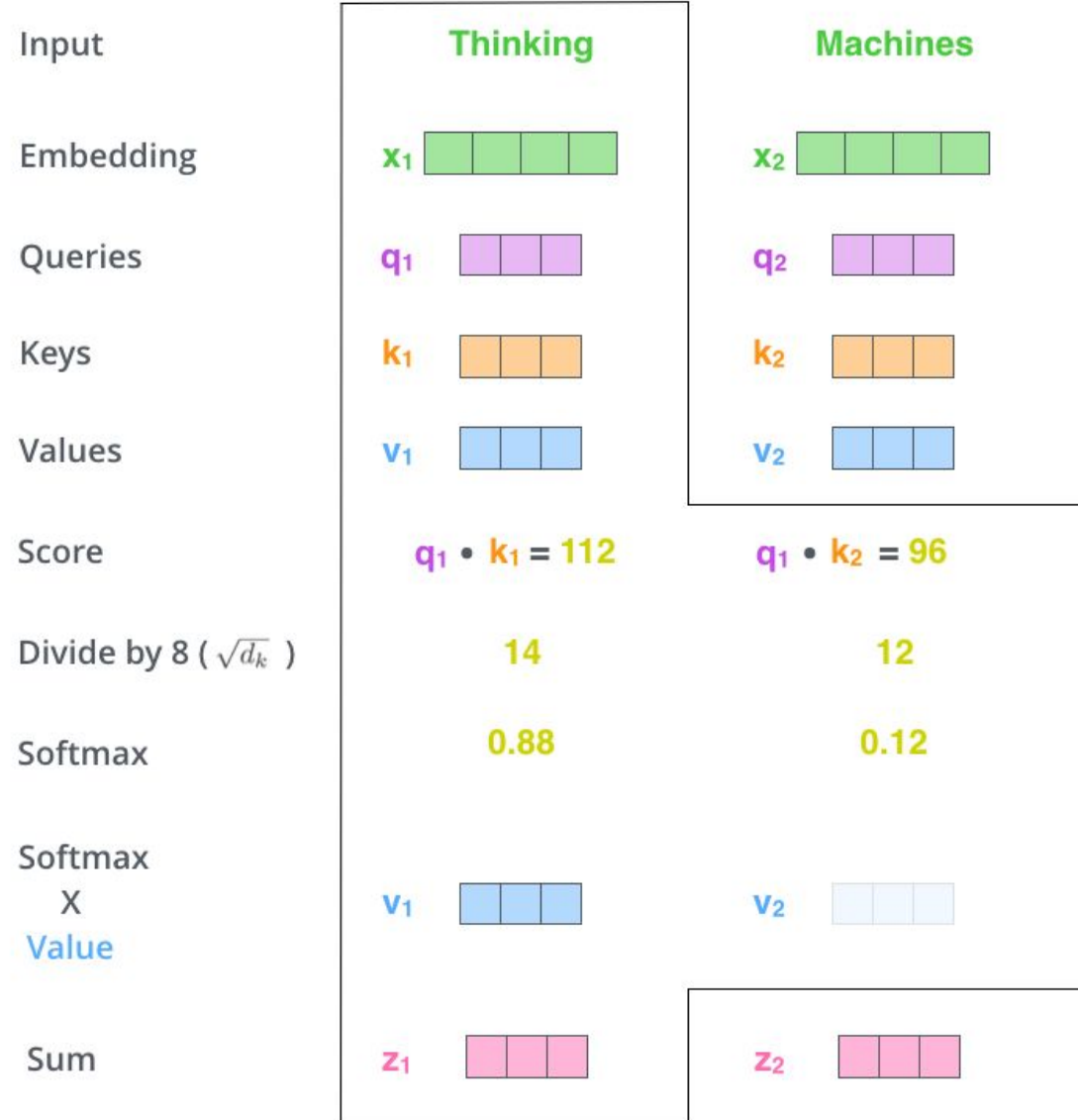


Transformer Architecture



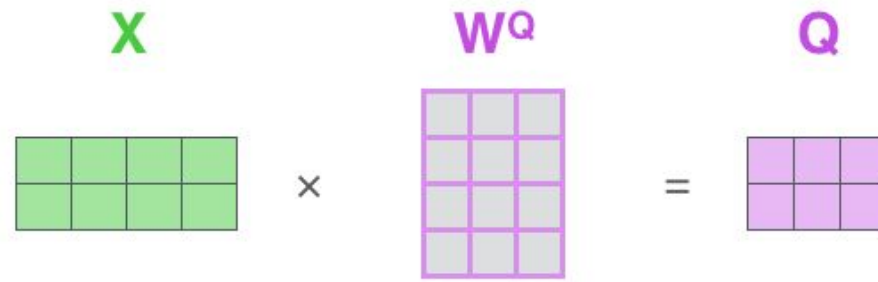


Transformer Architecture



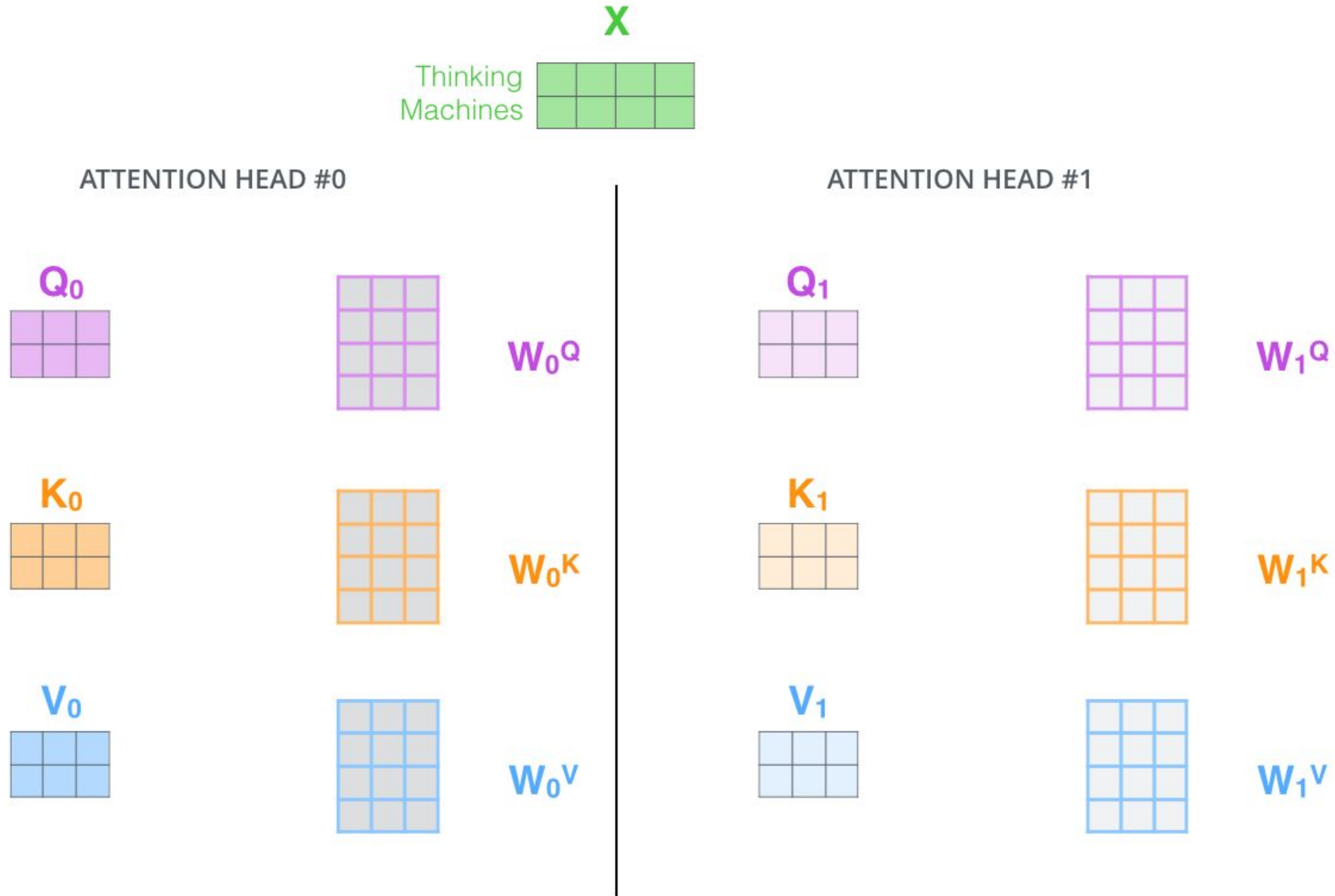


Transformer Architecture





Transformer Architecture



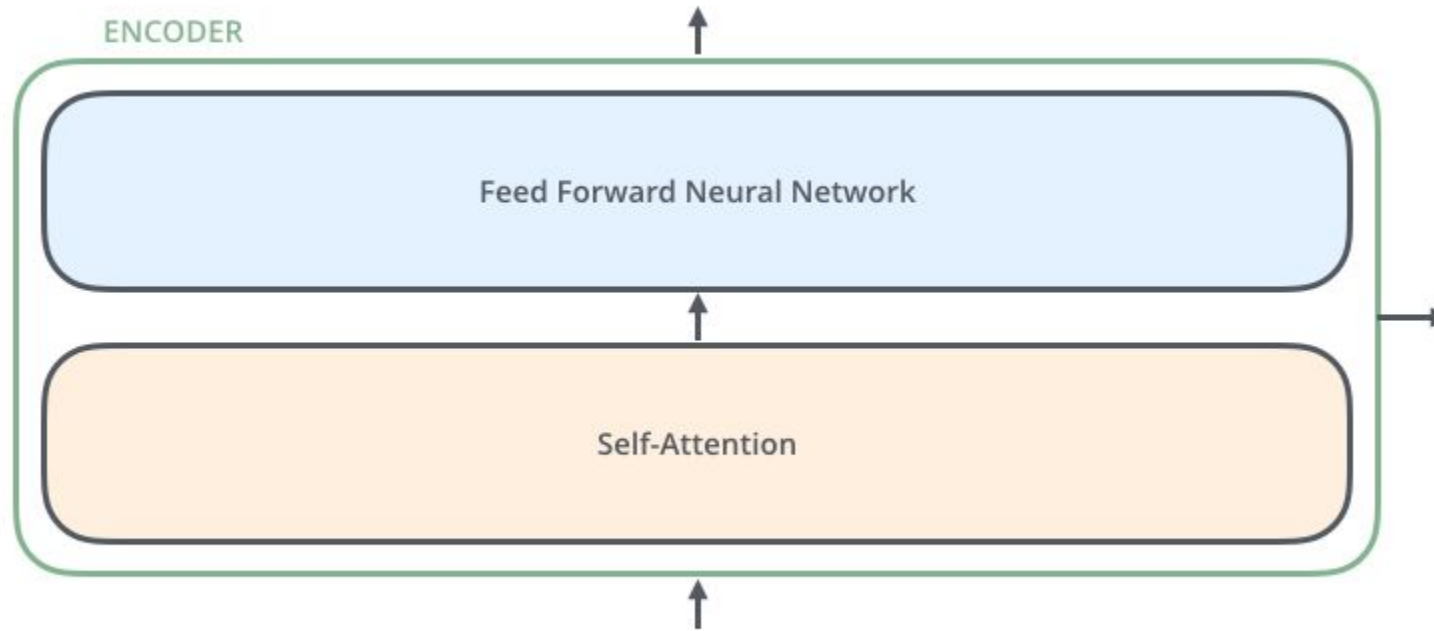


Position Embeddings



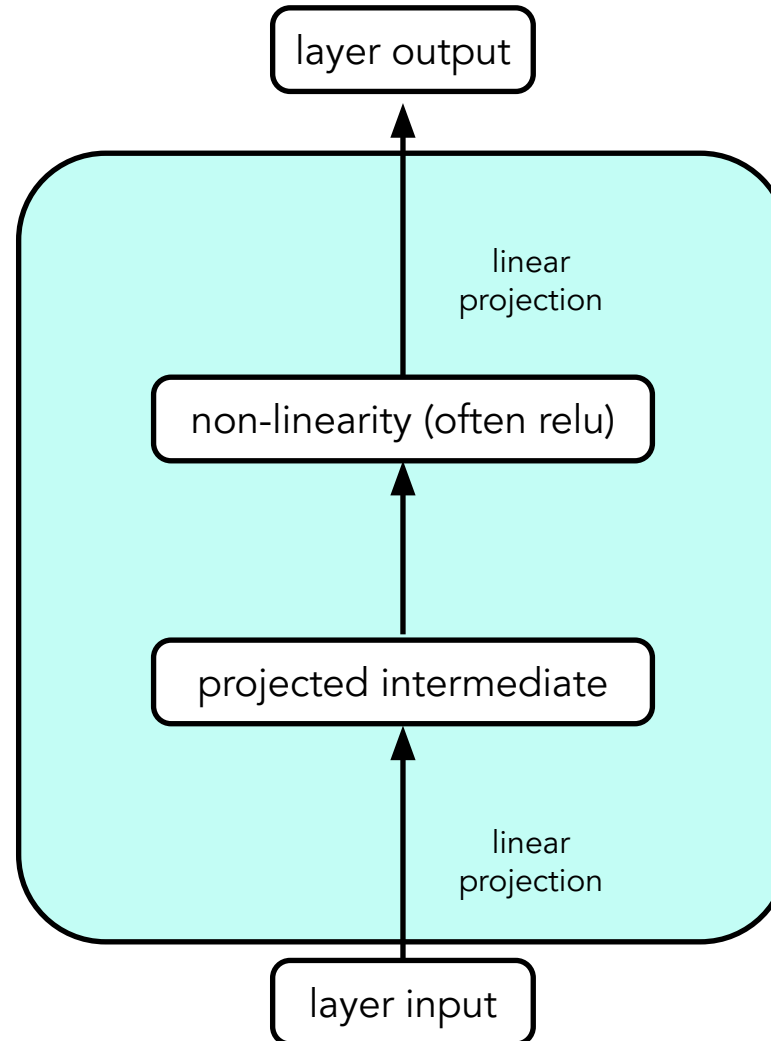


Transformer Architecture





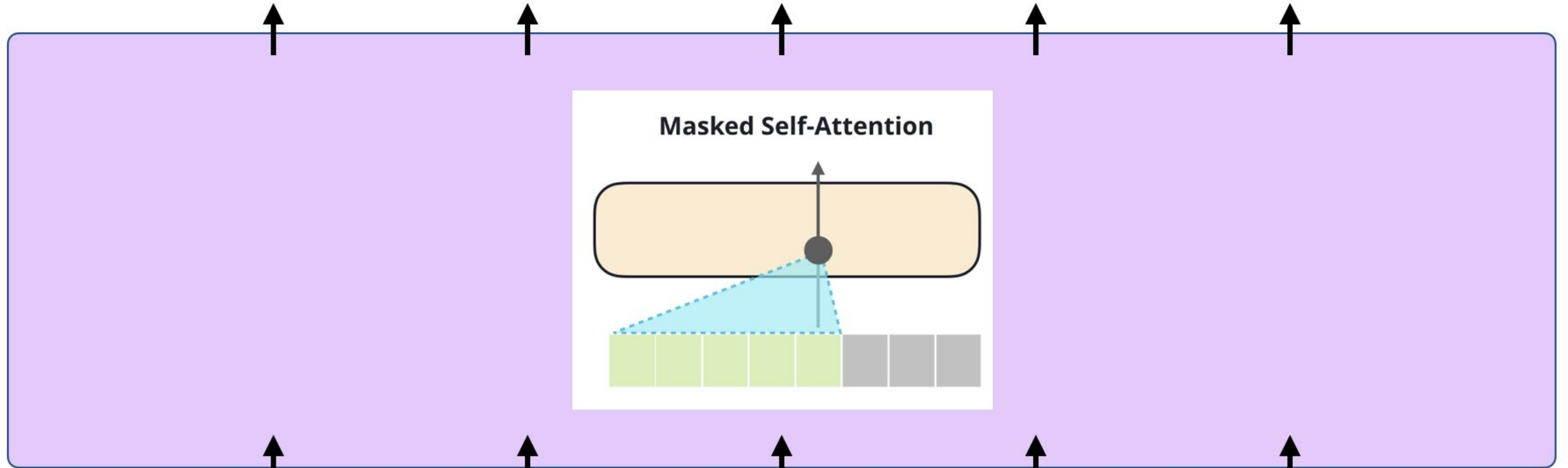
Feed-Forward





LM Training Objective

enjoys playing tennis . <eos>



↑ ↑ ↑ ↑ ↑
She enjoys playing tennis .

Existing LLMs

"First to pretrain on unlabeled text"



ELMO
Feb 2018

"Generative"



GPT-2
Feb 2019
(1.5B)



Megatron
Oct 2021
(530B)



PaLM
Apr 2022
(540B)

"Best open-release model at the moment"



LLAMA
Feb 2023
(65B)

BERT
Oct 2018
(340M)



"Pretrain and fine-tune"

GPT-3
June 2020
(175B)



"First to remove fine-tuning"

Chinchilla
March 2022
(70B)



"Data size is as important as parameter count"

ChatGPT
Nov 2022
(Unknown)



GPT-4
March 2023
(Unknown)



"Multimodal"





Failures of Pre-trained LMs

PROMPT

Explain the moon landing to a 6 year old in a few sentences.



Failures of Pre-trained LMs

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.



Failures of Pre-trained LMs

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

Language models are not *aligned* with user intent [[Ouyang et al., 2022](#)].



Failures of Pre-trained LMs

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

Language models are not *aligned* with user intent [[Ouyang et al., 2022](#)].

Finetuning to the rescue!



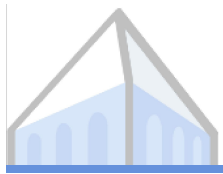
Failures of Pre-trained LMs

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION **Human**

A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

Language models are not *aligned* with user intent [[Ouyang et al., 2022](#)].
Finetuning to the rescue!



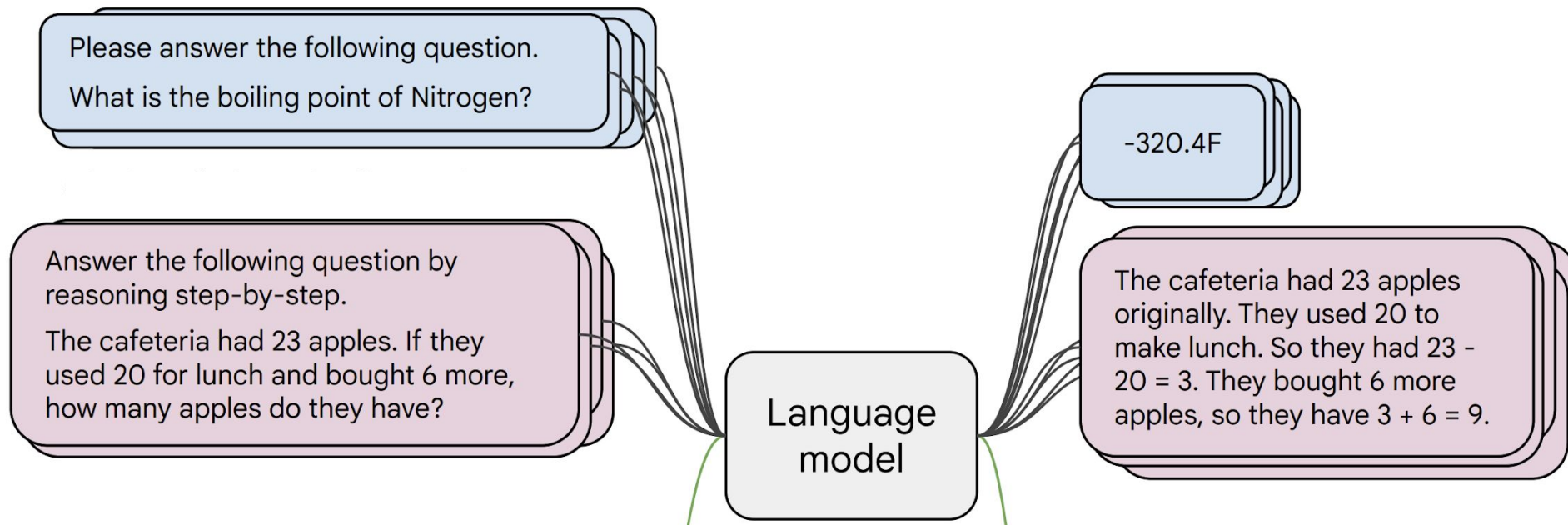
Instruction-tuning LLMs

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM



Instruction-tuning LLMs

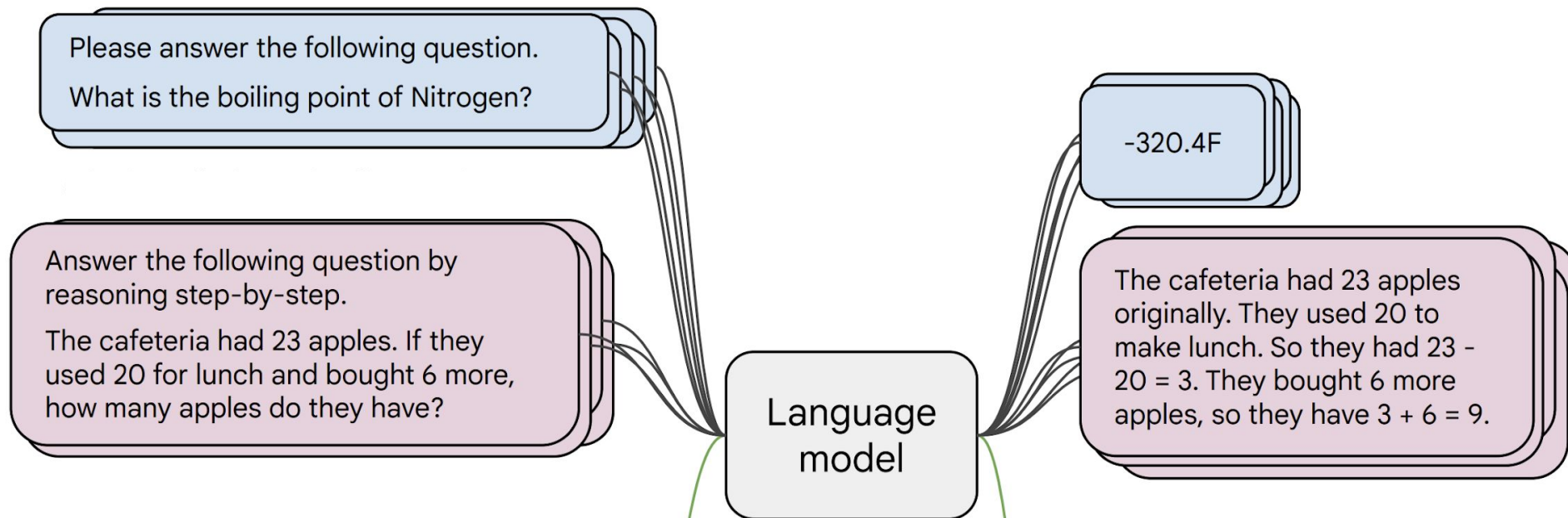
- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM





Instruction-tuning LLMs

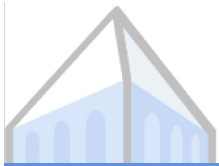
- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM



- Evaluate on **unseen tasks**

Q: Can Geoffrey Hinton have a conversation with George Washington?
Give the rationale before answering.

Geoffrey Hinton is a British-Canadian computer scientist born in 1947. George Washington died in 1799. Thus, they could not have had a conversation together. So the answer is "no".



Instruction-tuning LLMs for Chat

< 2/2 >



What is instruction tuning for LMs? Explain in one paragraph.



Instruction tuning for language models (LMs) involves modifying the model's behavior through explicit instructions or directives provided in the form of prompts or conditioning phrases. This approach aims to guide the output generation of the LM towards desired outcomes, making it more controllable and aligned with user intentions. By adjusting the instructions, users can influence the style, content, or context of the generated text, enabling more specific and tailored responses from the model. Instruction tuning enhances the practicality and adaptability of LMs by allowing users to exert finer control over the generated outputs and align the model's responses with their preferences and requirements.



Can you do it again, but now with 3 paragraphs?



Instruction tuning for language models (LMs) involves the strategic manipulation of input prompts or conditioning phrases to steer the model's output towards desired outcomes. In the context of fine-tuning or adapting pre-trained LMs, instruction tuning provides a mechanism for users to exert greater control over the generated text, ensuring that the



Regenerate

Send a message





Limitations of Instruction-tuning

- **Problem 1:** it's expensive to collect ground-truth data for tasks
 - *Provide me five active research areas in April 2023 for LLMs*
- **Problem 2:** tasks like open-ended creative generation have no right answer.
 - *Write me a story about a dog and her pet grasshopper.*
- Can we explicitly attempt to satisfy human preferences?



RL from Human Feedback



RL from Human Feedback

You

Write a Python function that checks if a given string is a palindrome.



ChatGPT

Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(word):
    # Convert the word to lowercase and remove spaces
    word = word.lower().replace(" ", "")

    # Compare the word with its reverse
    if word == word[::-1]:
        return True
    else:
        return False
```

Here's how you can use the function:

```
python Copy code
word1 = "radar"
print(is_palindrome(word1)) # Output: True

word2 = "Hello World"
print(is_palindrome(word2)) # Output: False
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing (`[::-1]`). If the word is equal to its reverse, it is a palindrome, and the function returns `True`. Otherwise, it returns `False`.



RL from Human Feedback

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(word):
    # Convert the word to lowercase and remove spaces
    word = word.lower().replace(" ", "")

    # Compare the word with its reverse
    if word == word[::-1]:
        return True
    else:
        return False
```

Here's how you can use the function:

```
python Copy code
word1 = "radar"
print(is_palindrome(word1)) # Output: True

word2 = "Hello World"
print(is_palindrome(word2)) # Output: False
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing ('[::-1]'). If the word is equal to its reverse, it is a palindrome, and the function returns 'True'. Otherwise, it returns 'False'.

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(string):
    # Convert the string to lowercase and remove spaces
    string = string.lower().replace(" ", "")

    # Check if the reversed string is equal to the original string
    return string == string[::-1]
```

This function takes a string as input, converts it to lowercase, and removes any spaces. It then checks if the reversed string is equal to the original string. If they are equal, the function returns 'True', indicating that the string is a palindrome. Otherwise, it returns 'False'.

Here's an example of how you can use the function:

```
python Copy code
word = "radar"
if is_palindrome(word):
    print(f"{word} is a palindrome.")
else:
    print(f"{word} is not a palindrome.")
```

Output:

```
csharp Copy code
radar is a palindrome.
```

Note that this function considers only alphanumeric characters and ignores case sensitivity and spaces. If you want to consider other characters as well, you can modify the function accordingly.

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(s):
    # Remove all non-alphanumeric characters and convert to lowercase
    s = ''.join(char.lower() for char in s if char.isalnum())

    # Check if the reversed string is equal to the original string
    return s == s[::-1]
```

In this function, we first remove all non-alphanumeric characters from the string and convert it to lowercase using a list comprehension. Then, we check if the reversed string is equal to the original string using slicing ('[::-1]') creates a reversed copy of the string. If they are equal, the string is a palindrome, and the function returns 'True'; otherwise, it returns 'False'.

Here's an example usage of the function:

```
python Copy code
string1 = "A man, a plan, a canal: Panama"
print(is_palindrome(string1)) # True

string2 = "race car"
print(is_palindrome(string2)) # True

string3 = "hello world"
print(is_palindrome(string3)) # False
```

In the above example, 'is_palindrome' is called with three different strings. The function correctly identifies the palindromes ("A man, a plan, a canal: Panama" and "race car") and returns 'True'. For the non-palindrome "hello world," it returns 'False'.



RL from Human Feedback

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(word):
    # Convert the word to lowercase and remove spaces
    word = word.lower().replace(" ", "")

    # Compare the word with its reverse
    if word == word[::-1]:
        return True
    else:
        return False
```

Here's how you can use the function:

```
python Copy code
word1 = "radar"
print(is_palindrome(word1)) # Output: True

word2 = "Hello World"
print(is_palindrome(word2)) # Output: False
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing ('[::-1]'). If the word is equal to its reverse, it is a palindrome, and the function returns 'True'. Otherwise, it returns 'False'.

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(string):
    # Convert the string to lowercase and remove spaces
    string = string.lower().replace(" ", "")

    # Check if the reversed string is equal to the original string
    return string == string[::-1]
```

This function takes a string as input, converts it to lowercase, and removes any spaces. It then checks if the reversed string is equal to the original string. If they are equal, the function returns 'True', indicating that the string is a palindrome. Otherwise, it returns 'False'.

Here's an example of how you can use the function:

```
python Copy code
word = "radar"
if is_palindrome(word):
    print(f"{word} is a palindrome.")
else:
    print(f"{word} is not a palindrome.")
```

Output:

```
csharp Copy code
radar is a palindrome.
```

Note that this function considers only alphanumeric characters and ignores case sensitivity and spaces. If you want to consider other characters as well, you can modify the function accordingly.

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```
python Copy code
def is_palindrome(s):
    # Remove all non-alphanumeric characters and convert to lowercase
    s = ''.join(char.lower() for char in s if char.isalnum())

    # Check if the reversed string is equal to the original string
    return s == s[::-1]
```

In this function, we first remove all non-alphanumeric characters from the string and convert it to lowercase using a list comprehension. Then, we check if the reversed string is equal to the original string using slicing ('[::-1]') creates a reversed copy of the string. If they are equal, the string is a palindrome, and the function returns 'True'; otherwise, it returns 'False'.

Here's an example usage of the function:

```
python Copy code
string1 = "A man, a plan, a canal: Panama"
print(is_palindrome(string1)) # True

string2 = "race car"
print(is_palindrome(string2)) # True

string3 = "hello world"
print(is_palindrome(string3)) # False
```

In the above example, 'is_palindrome' is called with three different strings. The function correctly identifies the palindromes ('A man, a plan, a canal: Panama' and 'race car') and returns 'True'. For the non-palindrome ('hello world'), it returns 'False'.





RL from Human Feedback

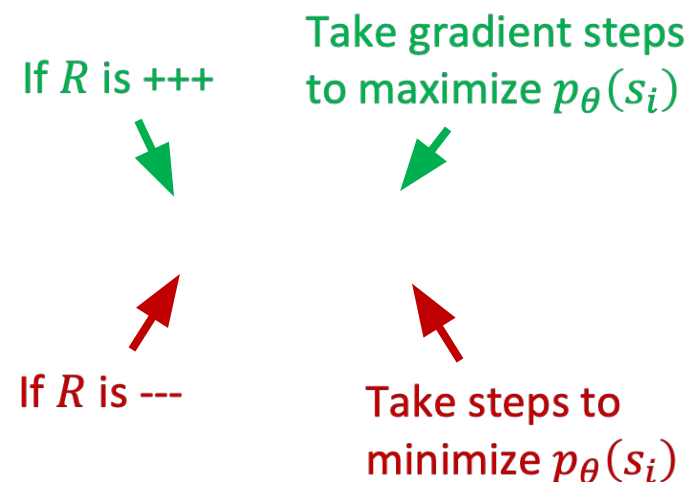
- **Step 1:** train reward model to predict human ratings
- **Step 2:** use RL to directly optimize against reward model



RL from Human Feedback

- **Step 1:** train reward model to predict human ratings
- **Step 2:** use RL to directly optimize against reward model

We **reinforce** good actions, increasing the chance they happen again.



GPT Assistant training pipeline

