Copyright Notice

DeepLearning.AI

aws

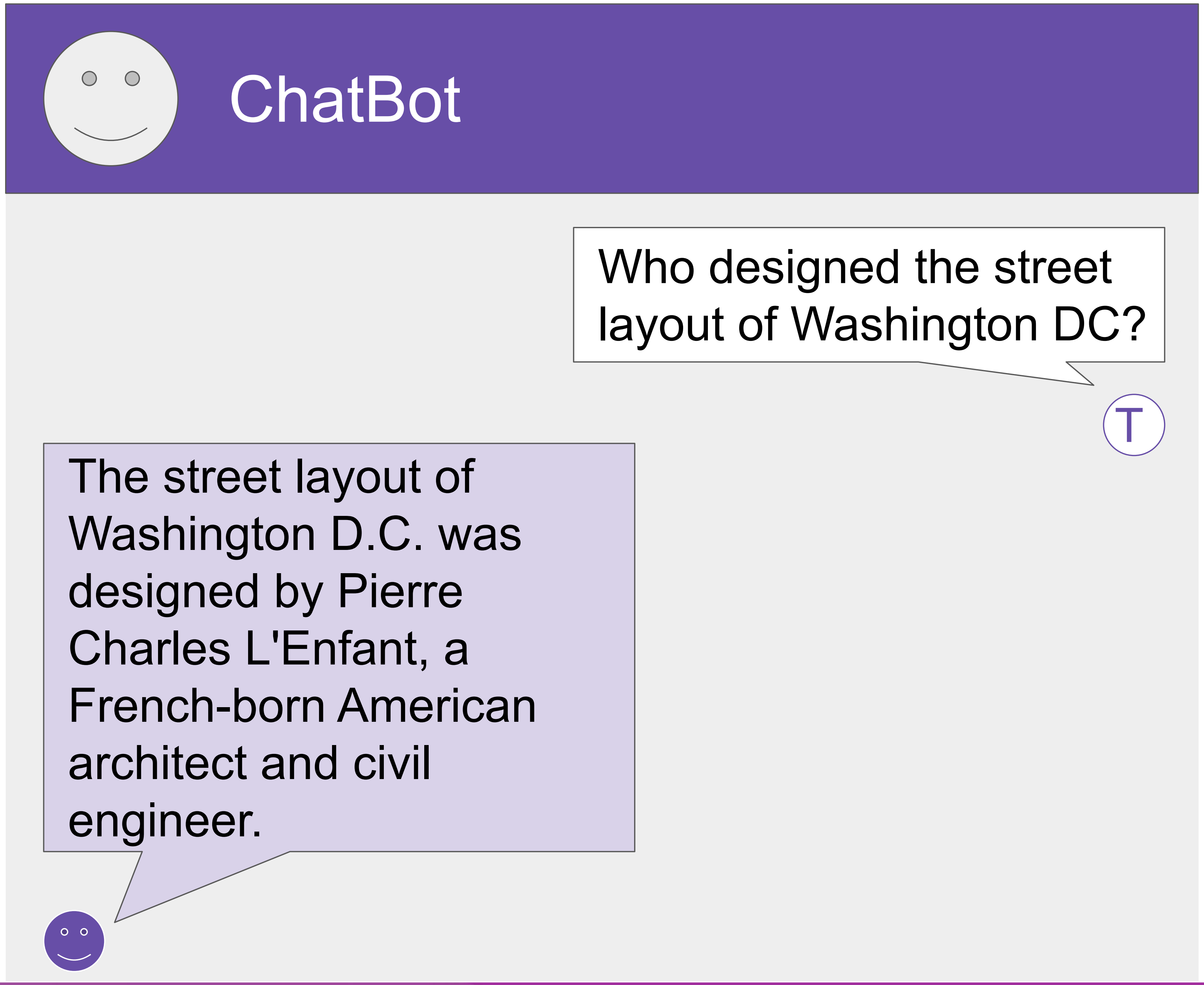
Generative AI & Large Language Models (LLMs)

USE CASES, PROJECT LIFECYCLE, AND MODEL PRE-TRAINING

Generative AI & Large Language Model Use Cases & Model Lifecycle

Generative AI & Large Language Models

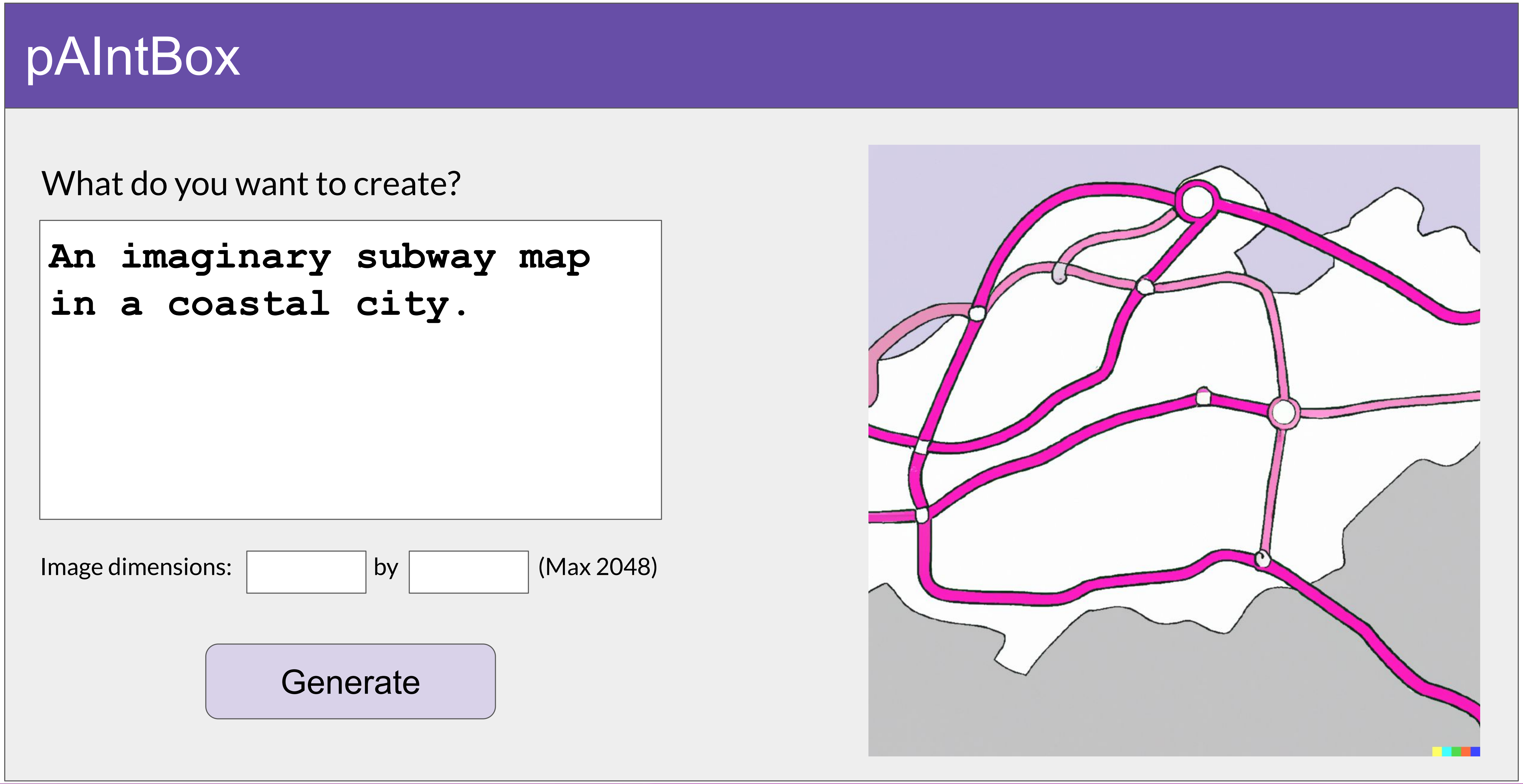
Generative AI



Who designed the street layout of Washington DC?

The street layout of Washington D.C. was designed by Pierre Charles L'Enfant, a French-born American architect and civil engineer.

Generative AI



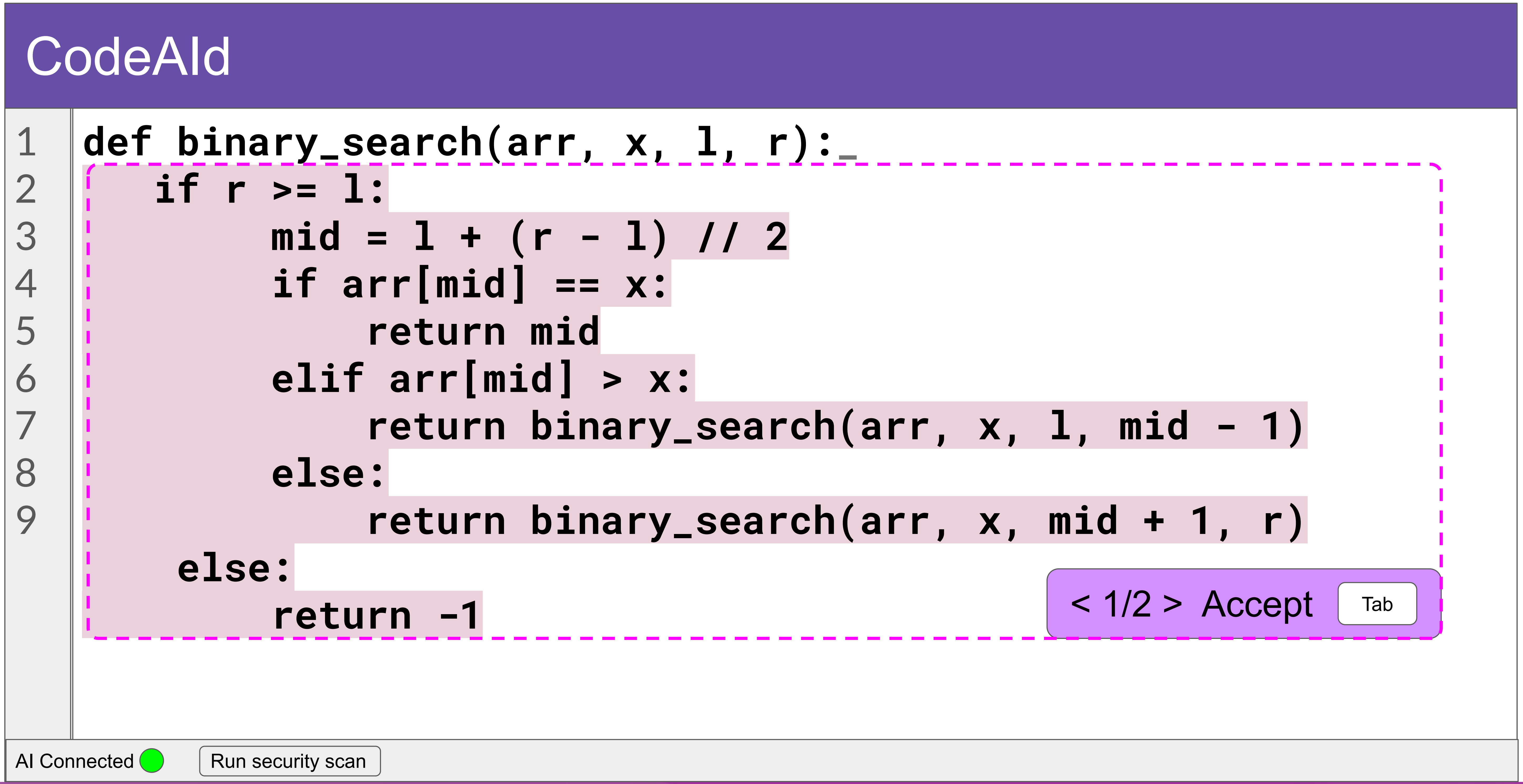
pAIntBox

What do you want to create?

```
An imaginary subway map
in a coastal city.
```

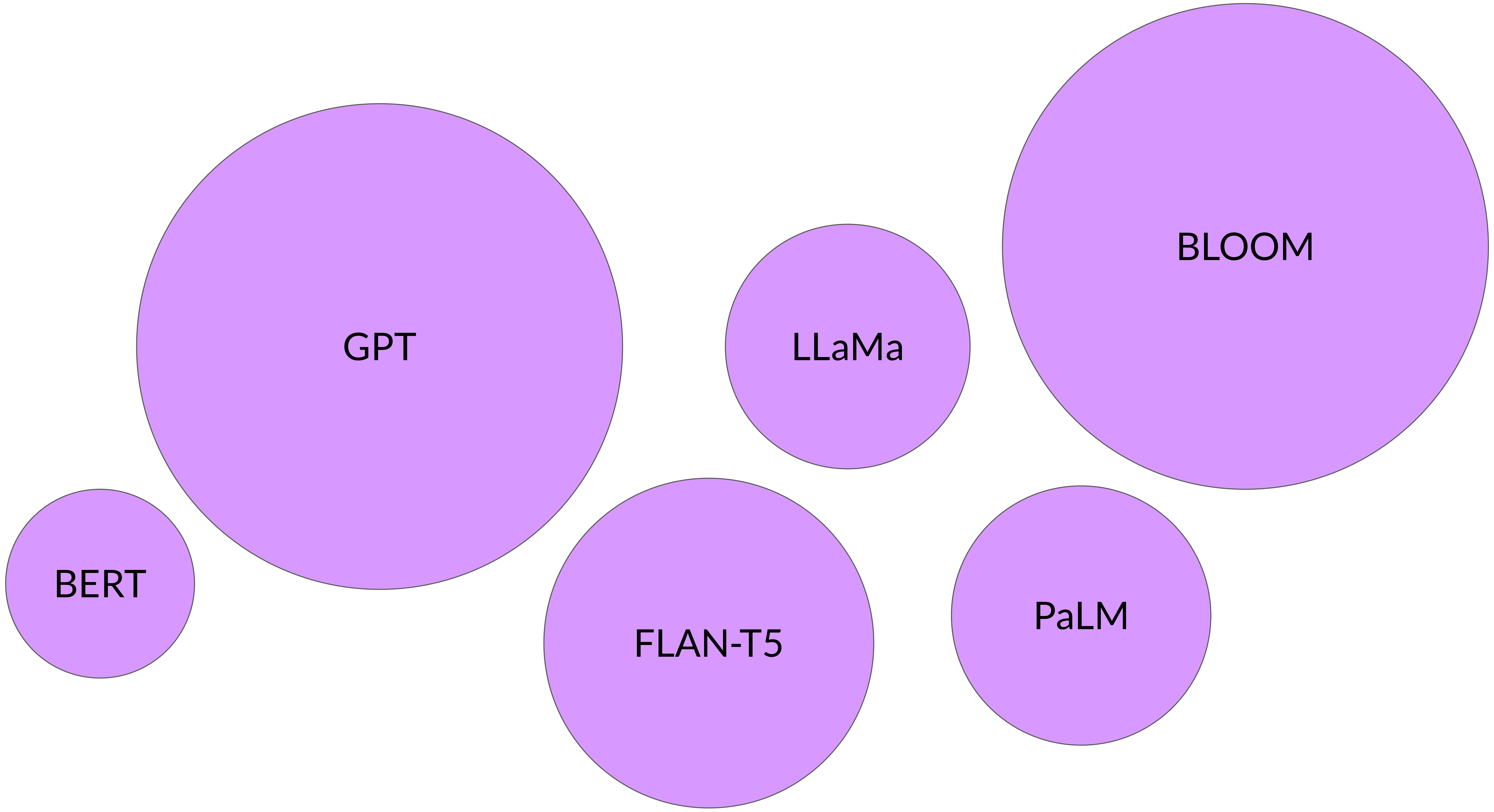
Image dimensions: [ ] by [ ] (Max 2048)

Generative AI

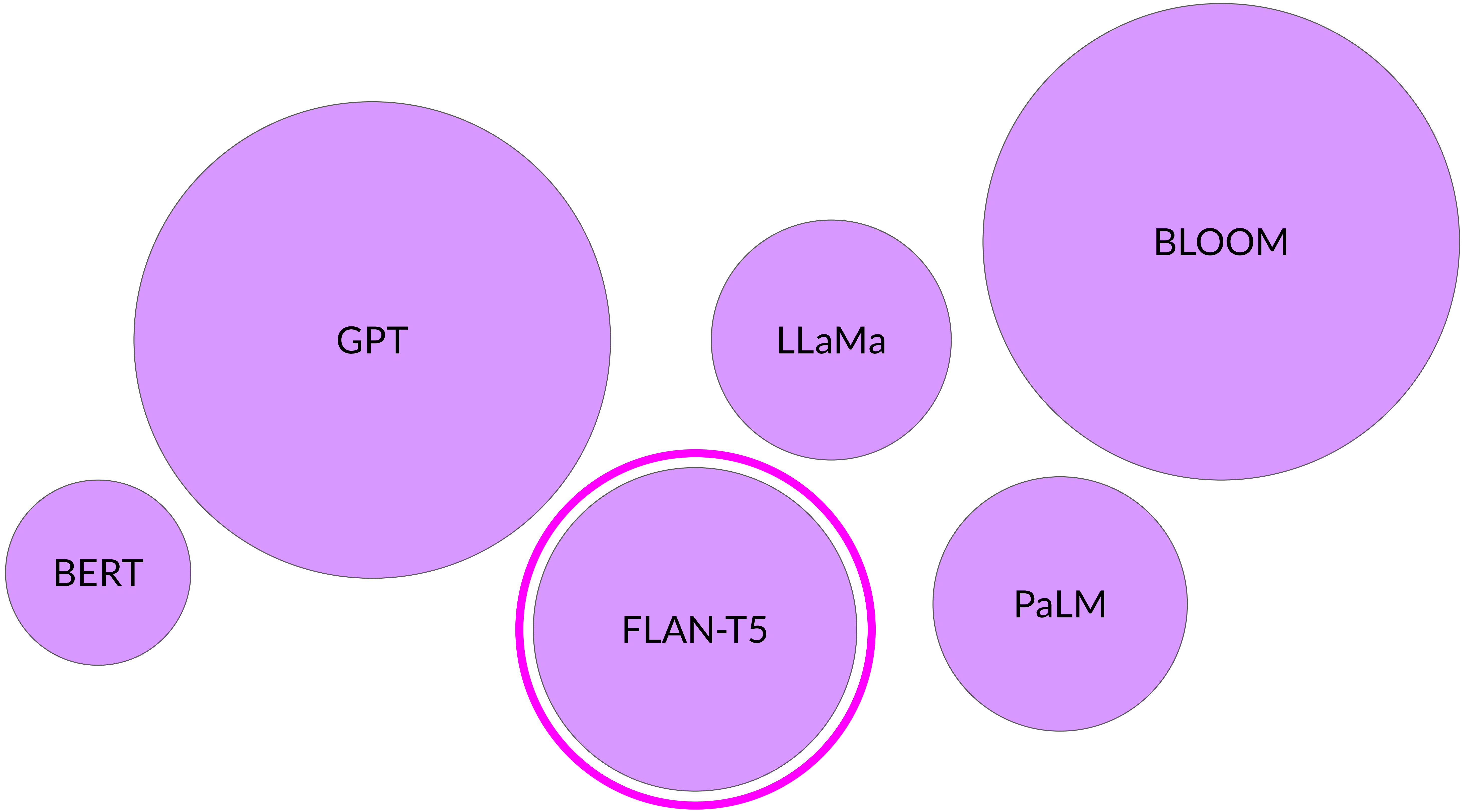


```
def binary_search(arr, x, l, r):_
   if r >= l:
        mid = l + (r - l) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binary_search(arr, x, l, mid - 1)
        else:
            return binary_search(arr, x, mid + 1, r)
    else:
        return -1
```

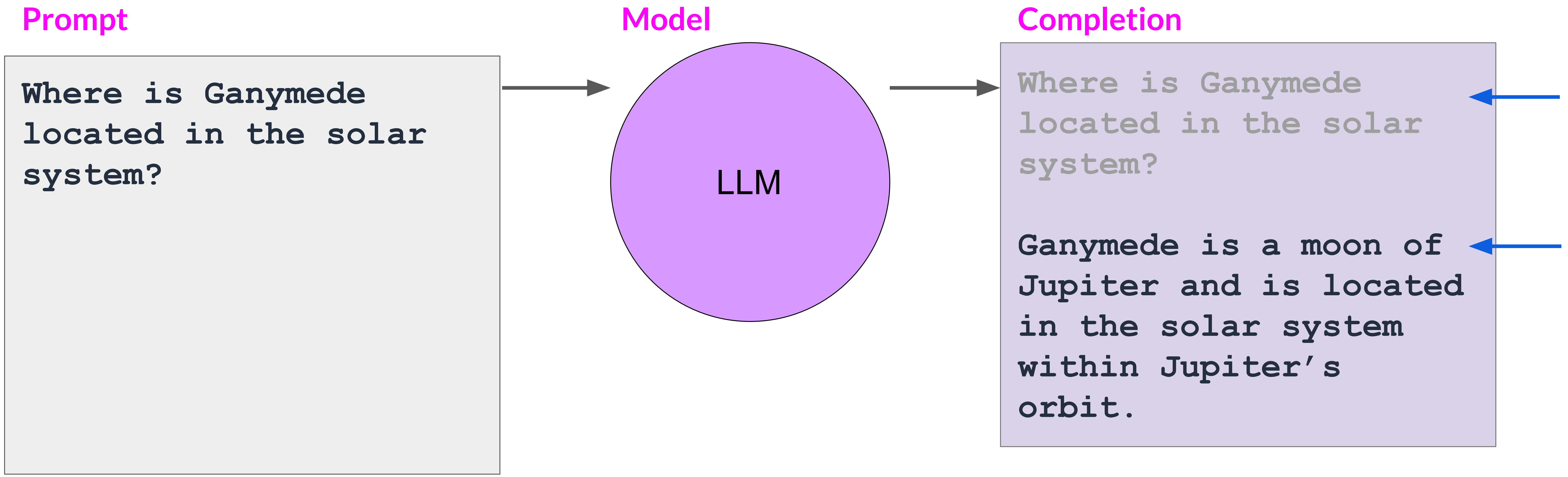
Large Language Models



Large Language Models



Prompts and completions



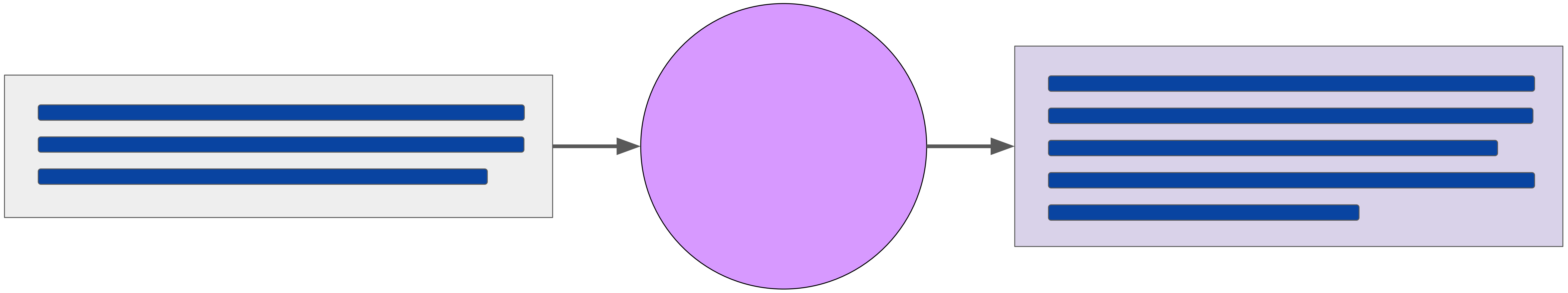
```
Where is Ganymede
located in the solar
system?
```

```
Where is Ganymede
located in the solar
system?

Ganymede is a moon of
Jupiter and is located
in the solar system
within Jupiter's
orbit.
```

Context window

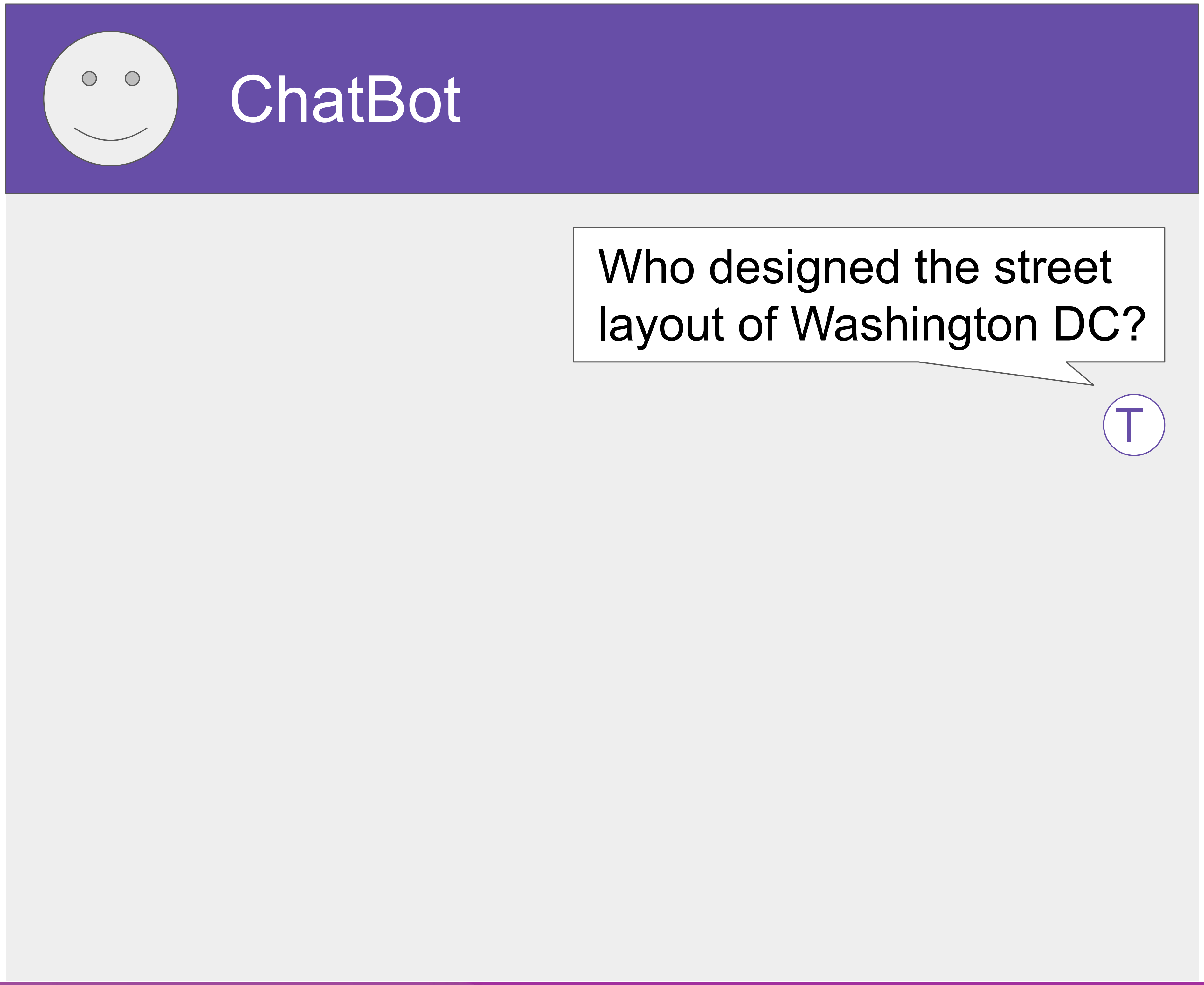
- typically a few 1000 words.

Prompts and completions



Use cases & tasks

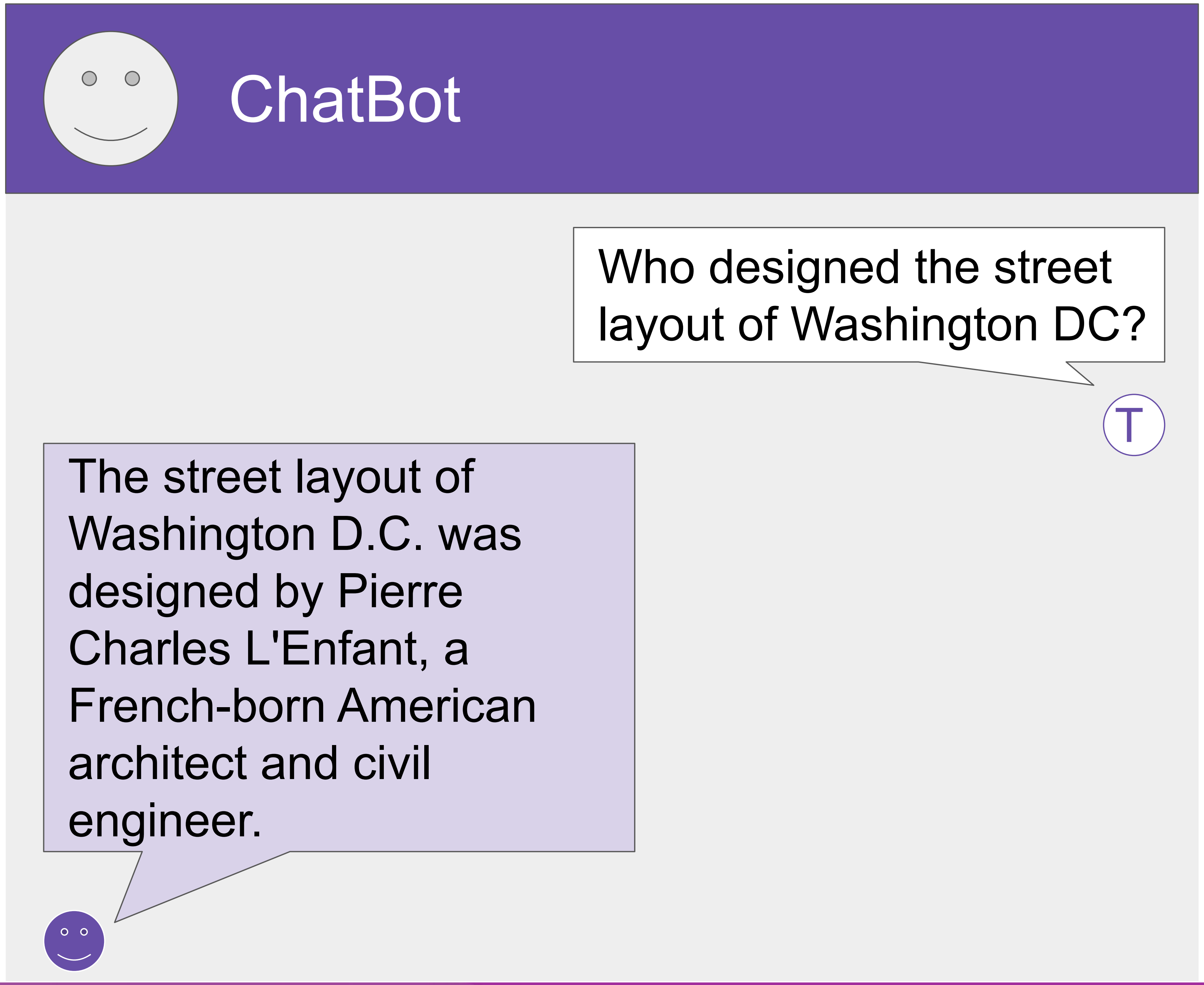
LLM chatbot



ChatBot

Who designed the street layout of Washington DC?

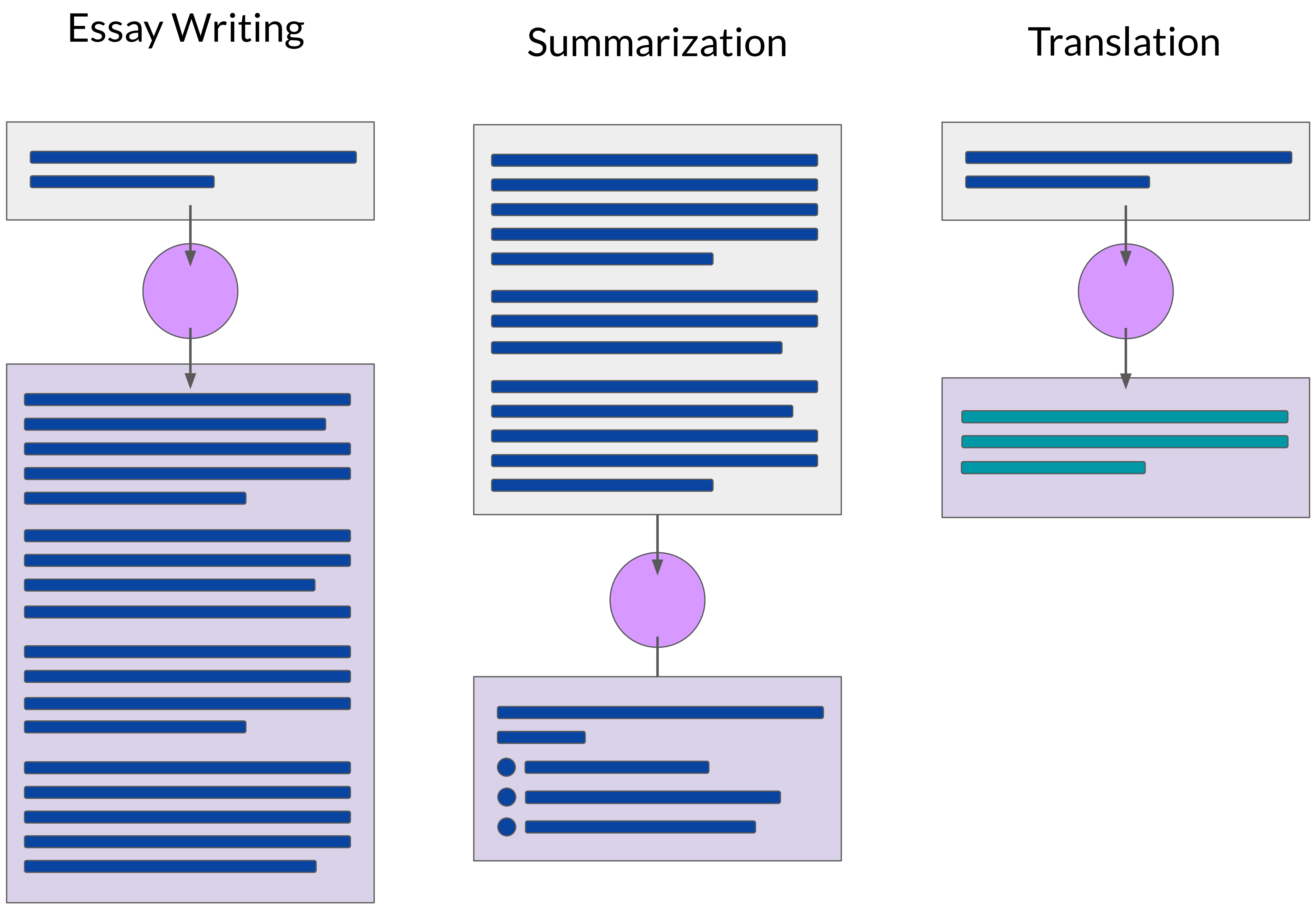
LLM chatbot



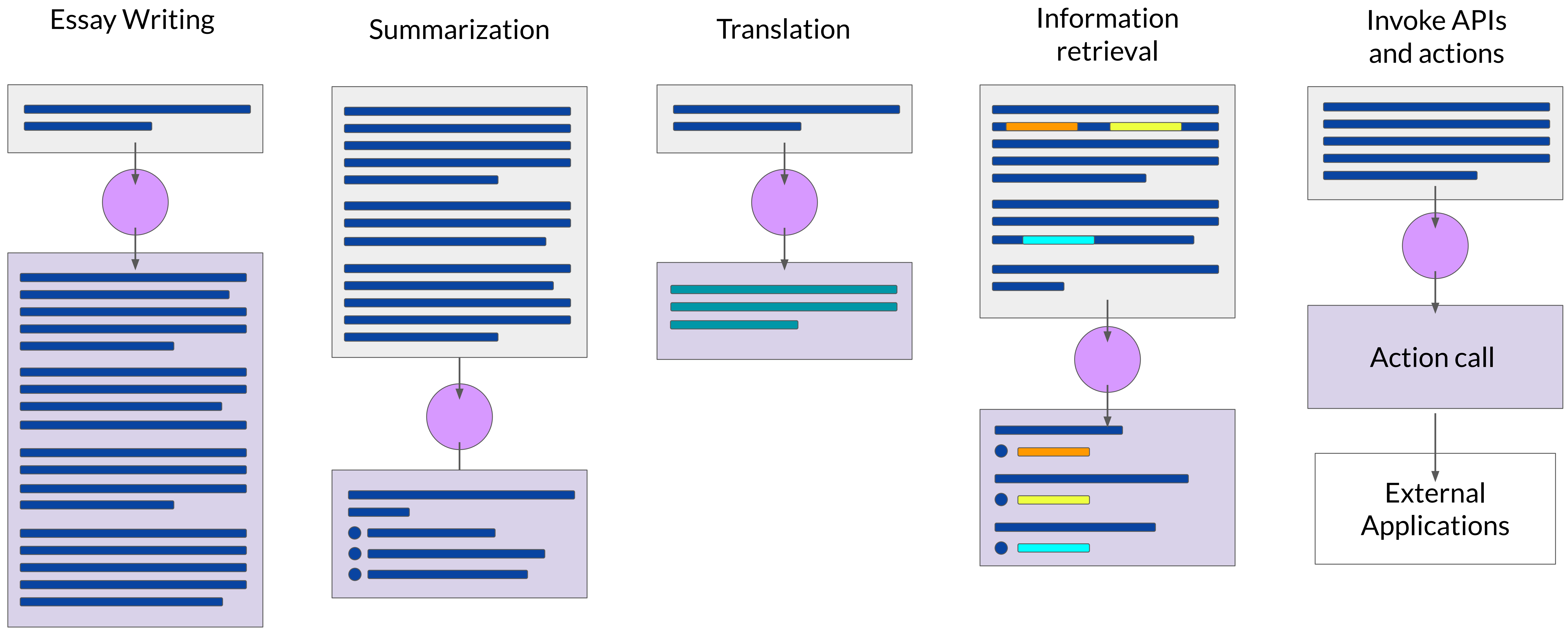
Who designed the street layout of Washington DC?

The street layout of Washington D.C. was designed by Pierre Charles L'Enfant, a French-born American architect and civil engineer.

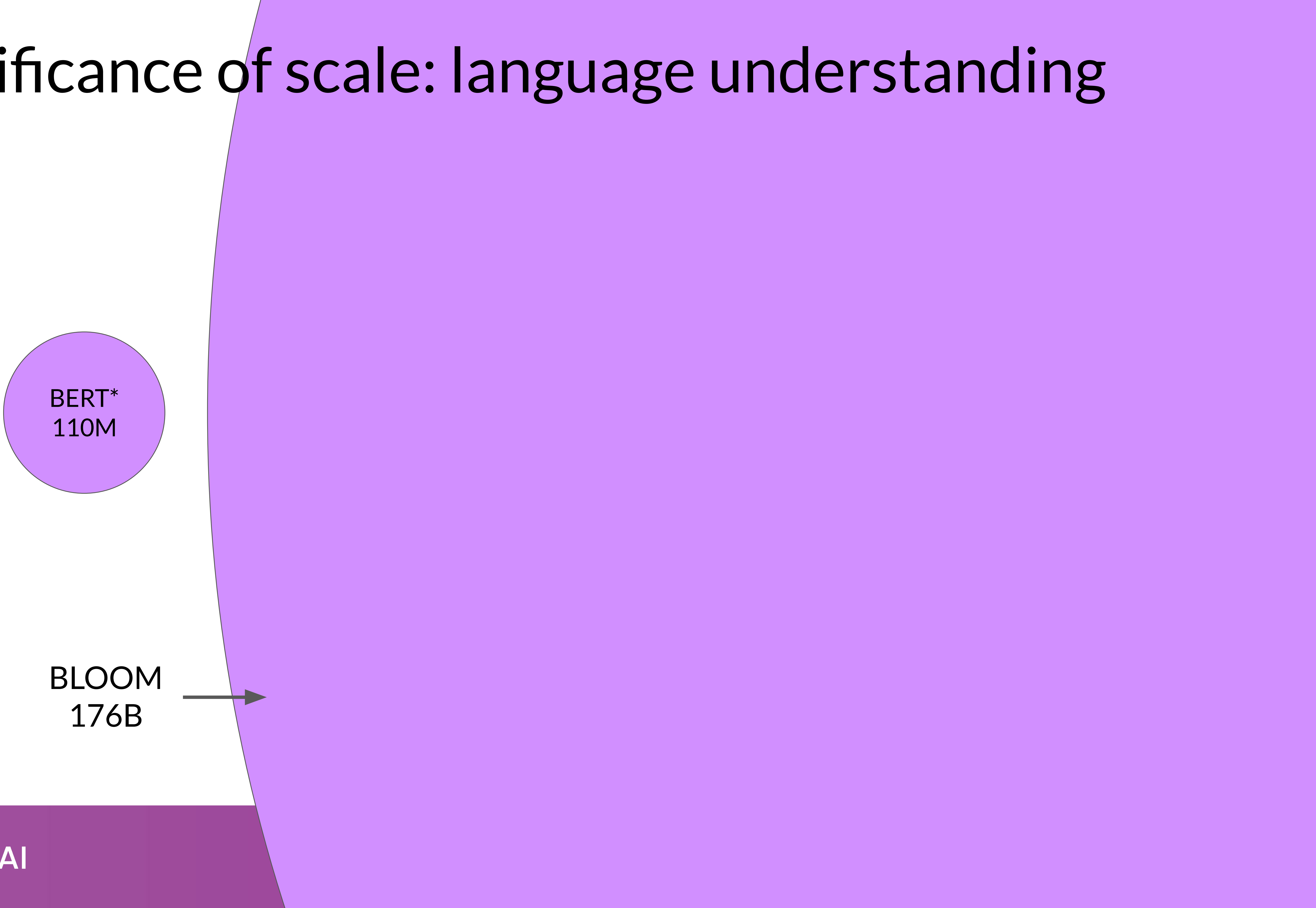
LLM use cases & tasks



LLM use cases & tasks



The significance of scale: language understanding

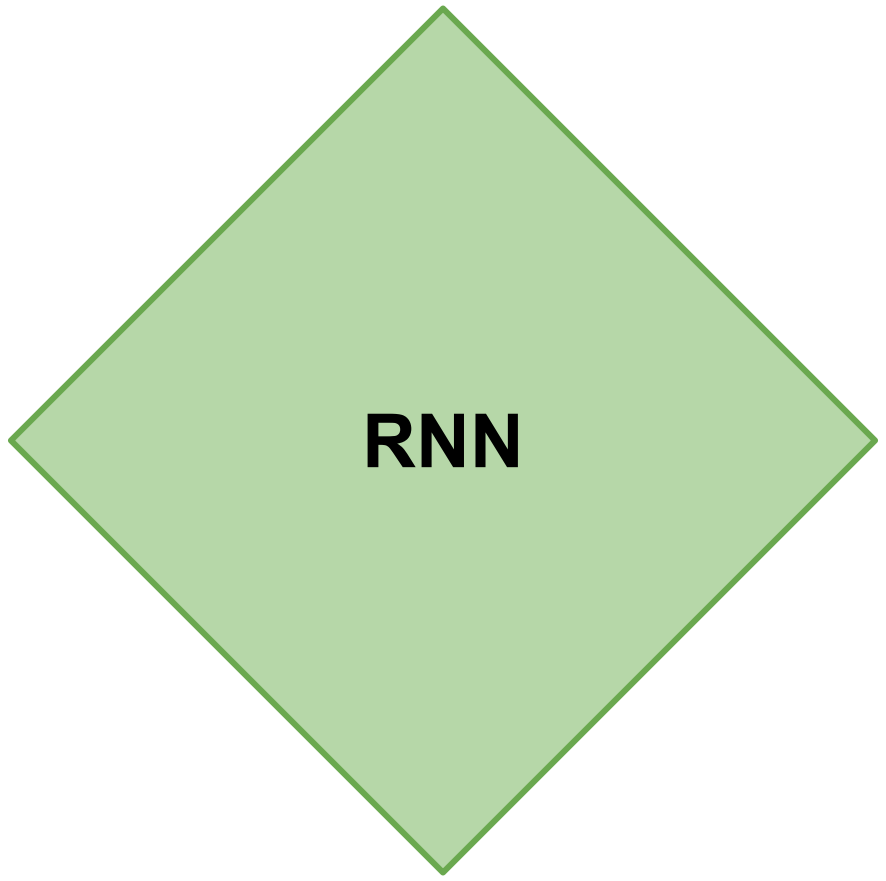


*Bert-base

DeepLearning.AI

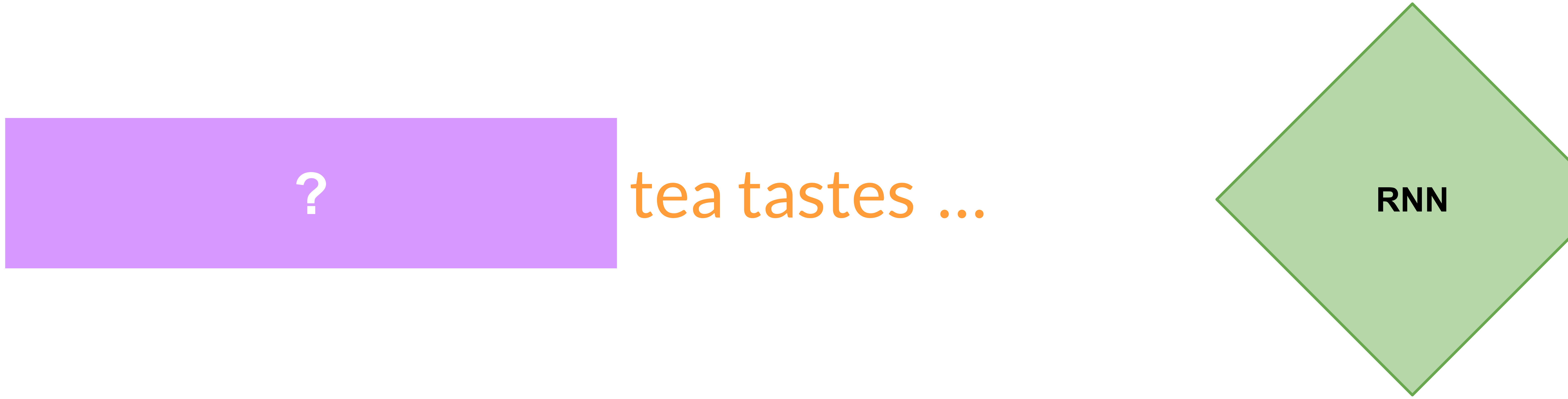
How LLMs work - Transformers architecture

Generating text with RNNs

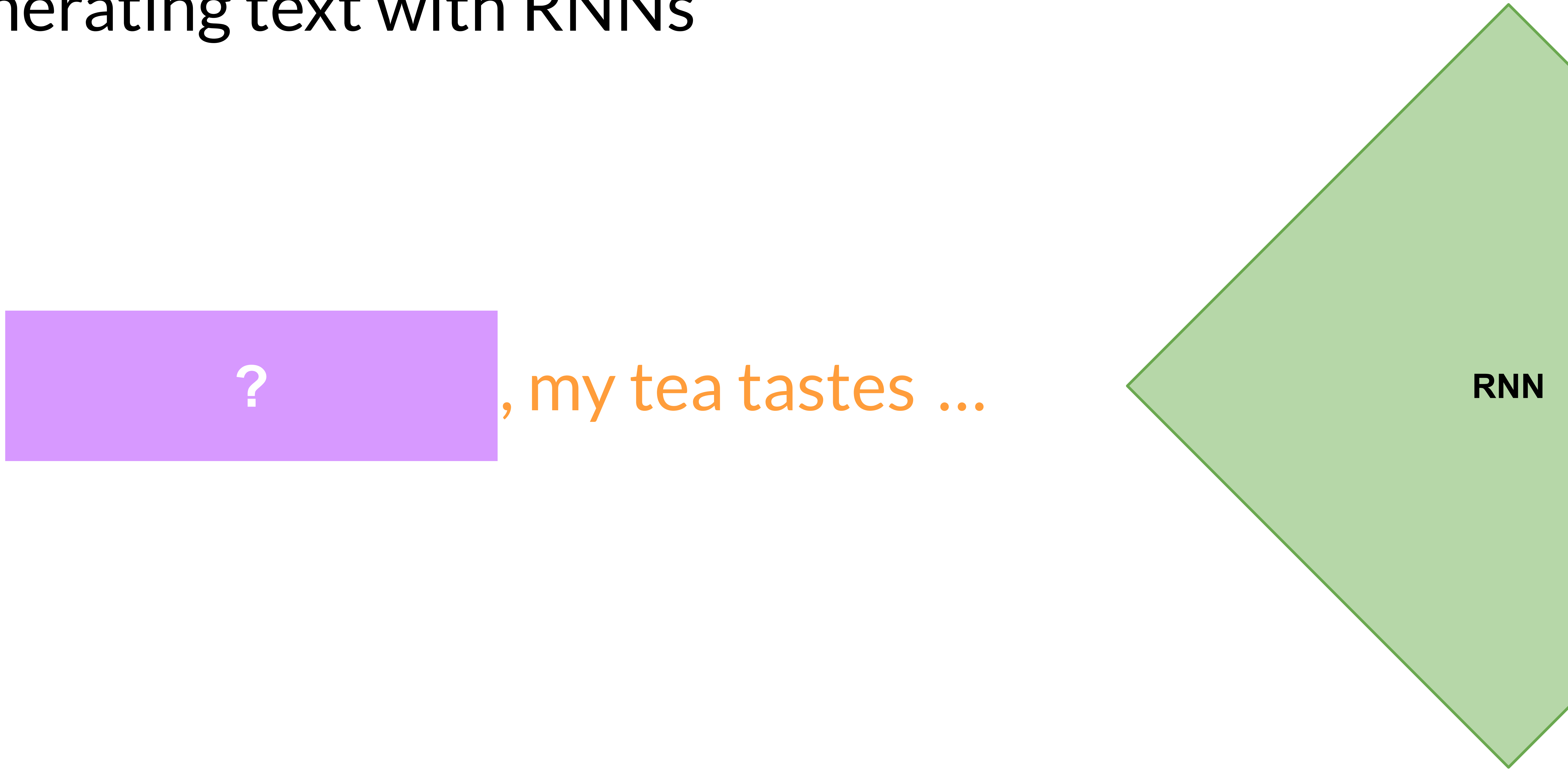


Generating text with RNNs

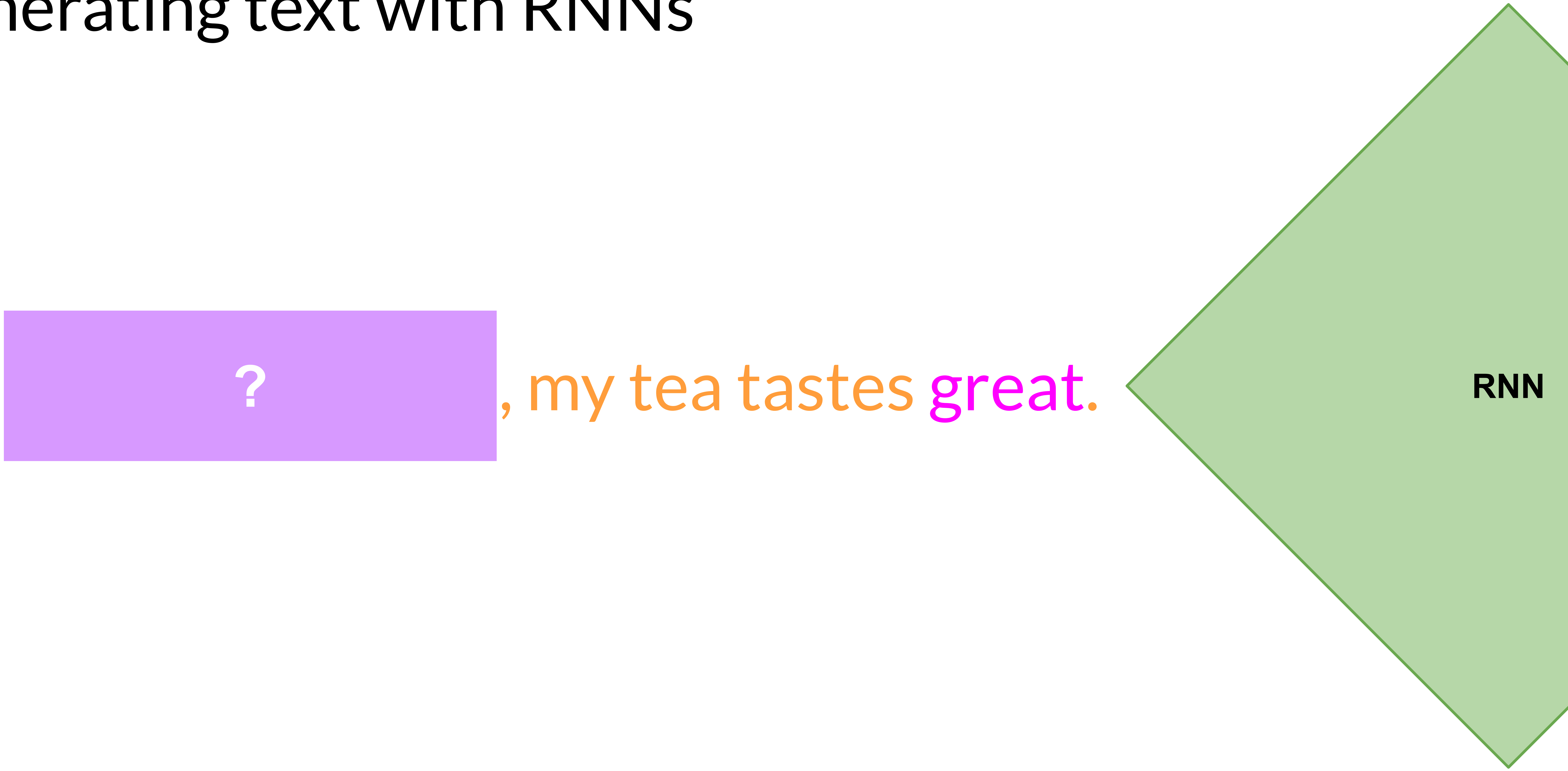
Generating text with RNNs



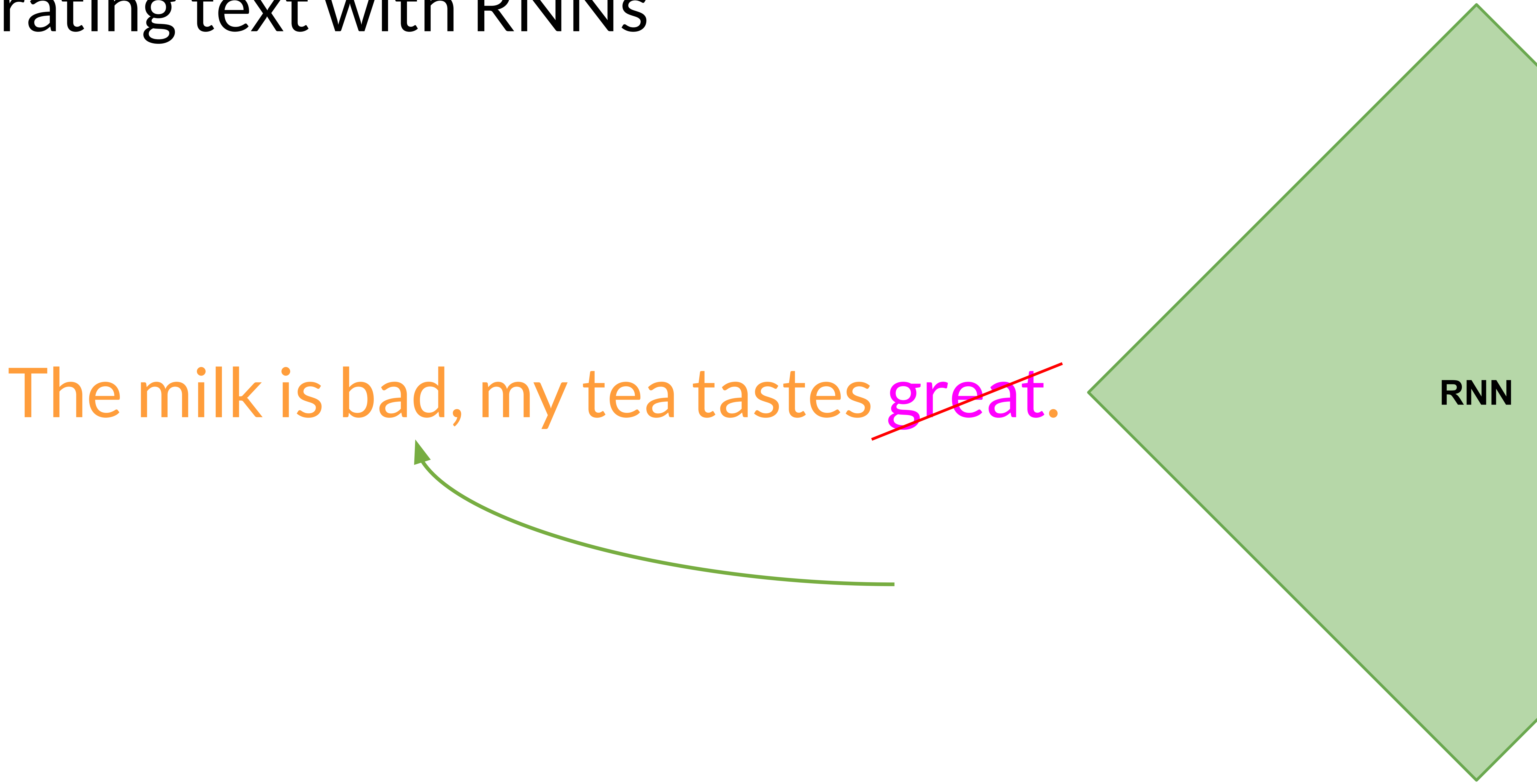
Generating text with RNNs



Generating text with RNNs



Generating text with RNNs



Understanding language can be challenging

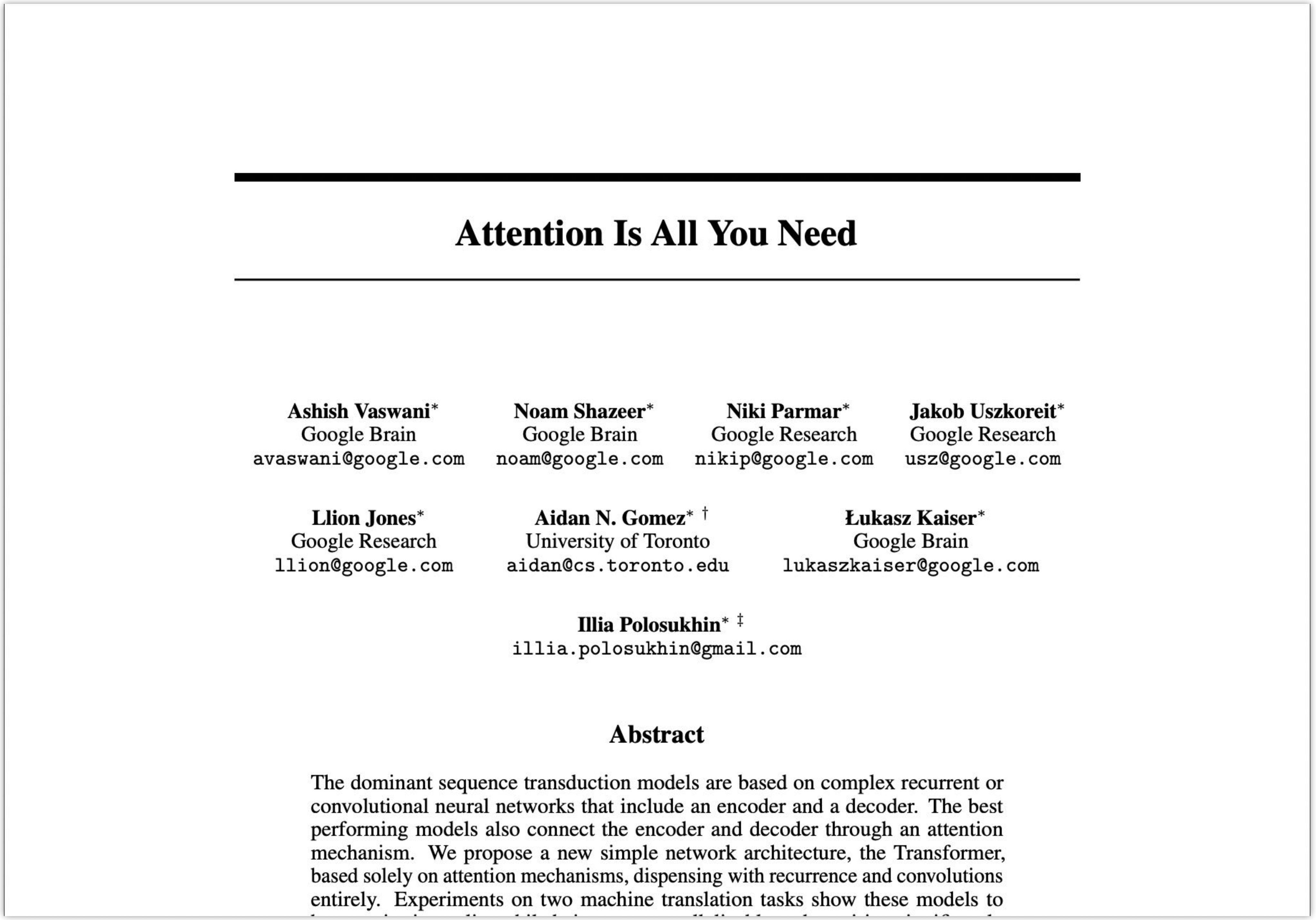
I took my money to the <u>bank</u>.

River bank?

Understanding language can be challenging

$$\text{The } \overbrace{\text{teacher taught the student } \underbrace{\text{with the book}}_{\text{The student's book?}}}^{\text{The teacher's book?}}\text{.}$$

Transformers



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

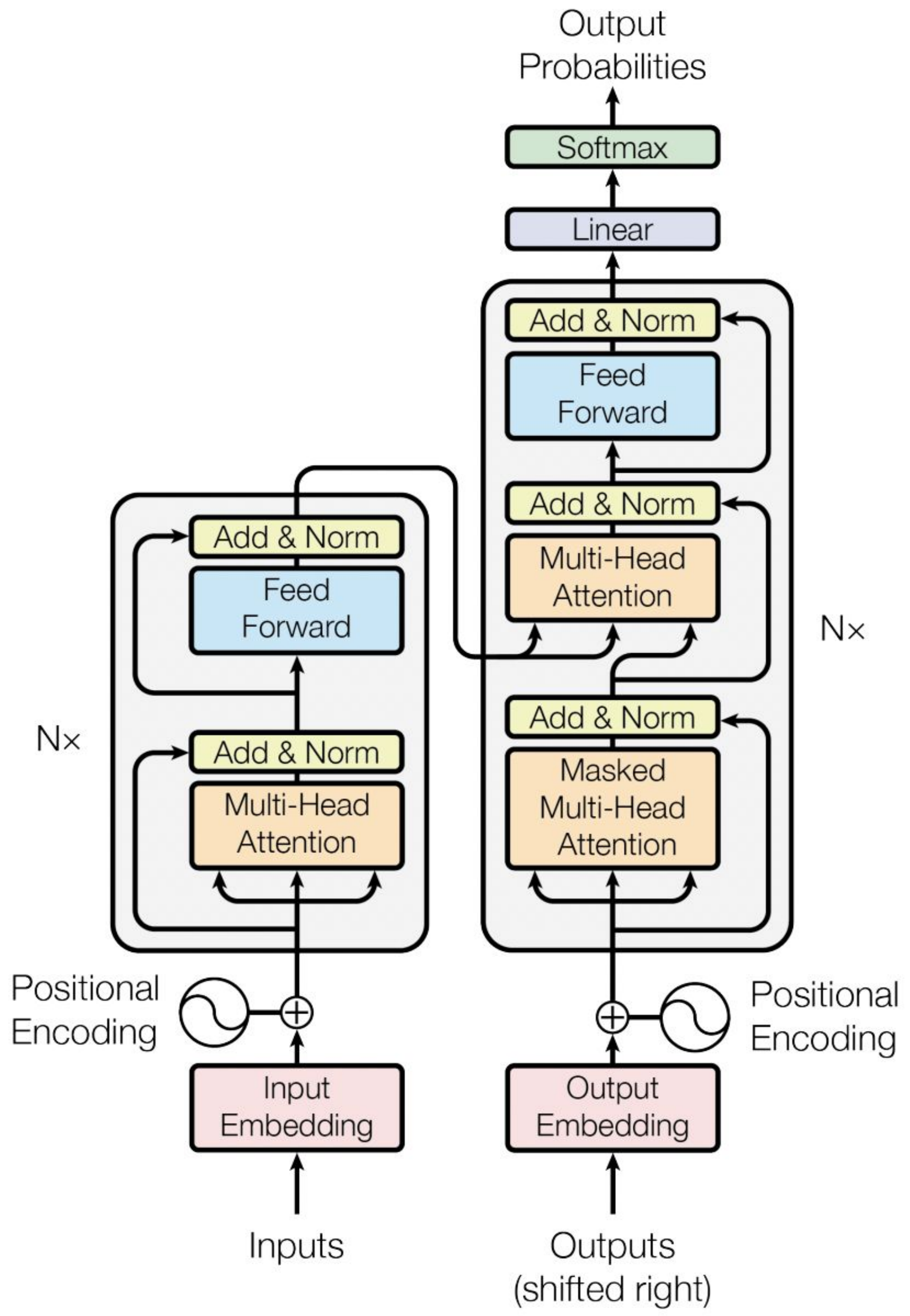
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

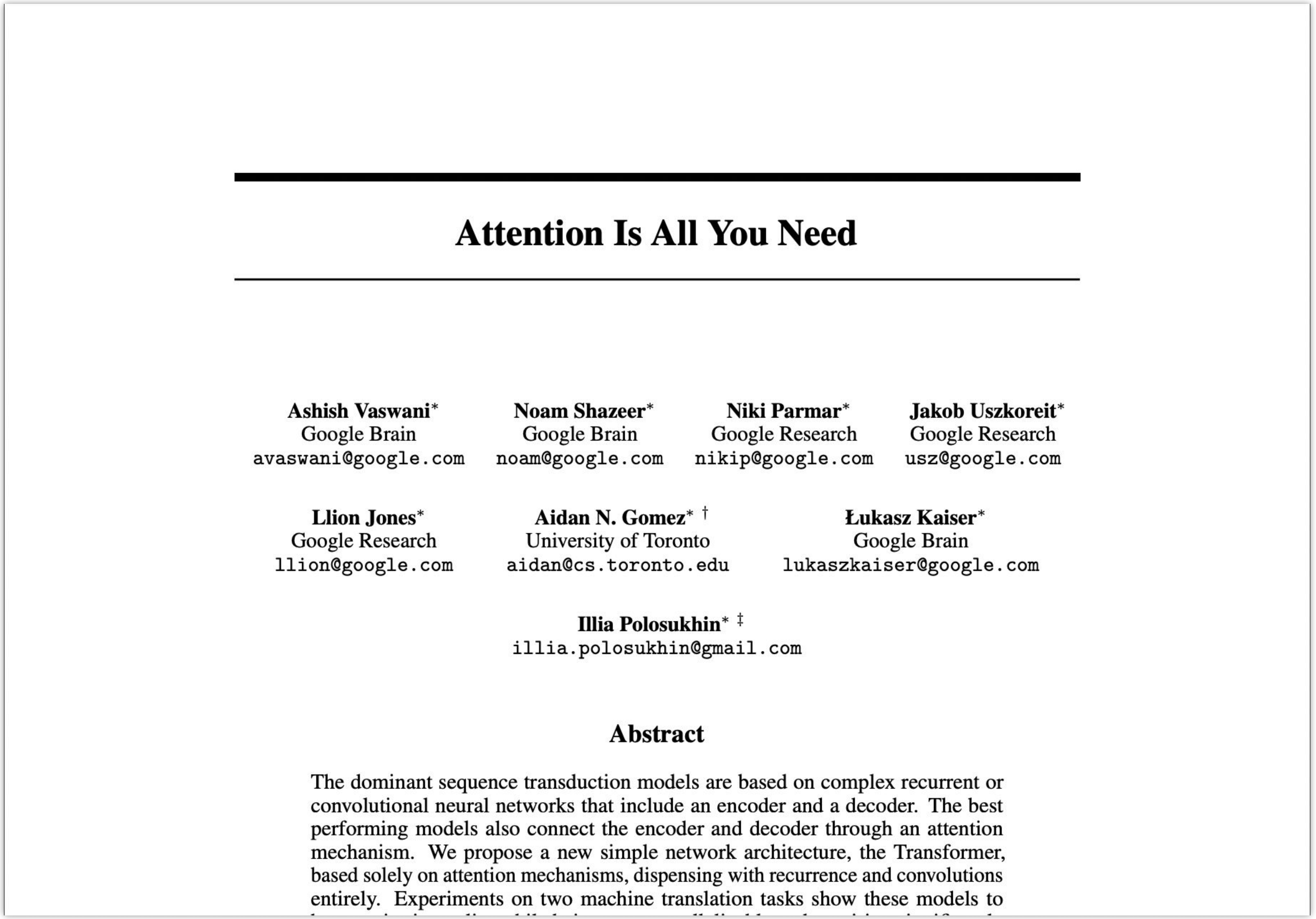
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to



DeepLearning.AI

aws

Transformers



Attention Is All You Need

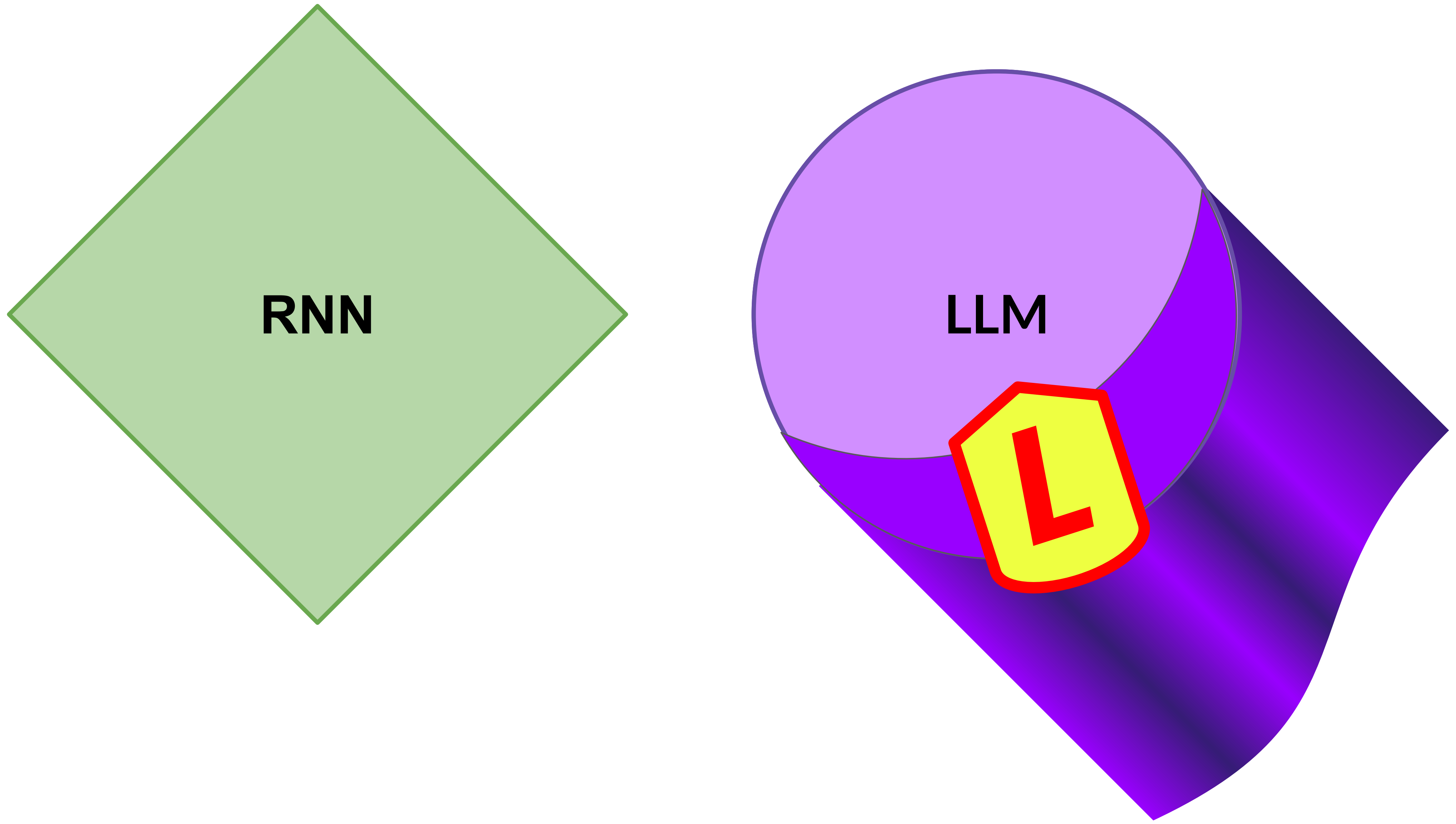
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

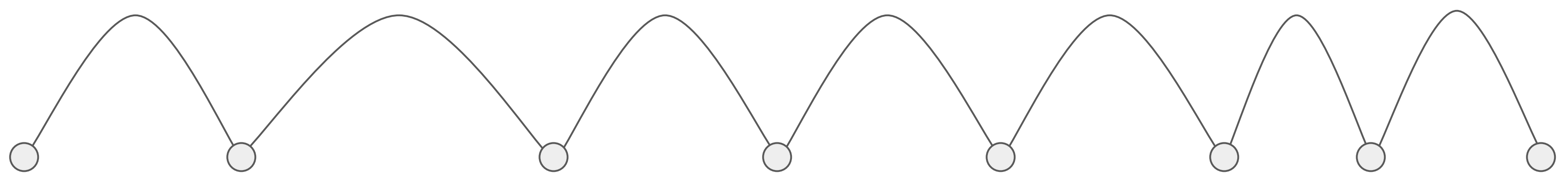
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

- Scale efficiently
- Parallel process
- Attention to input meaning

Transformers

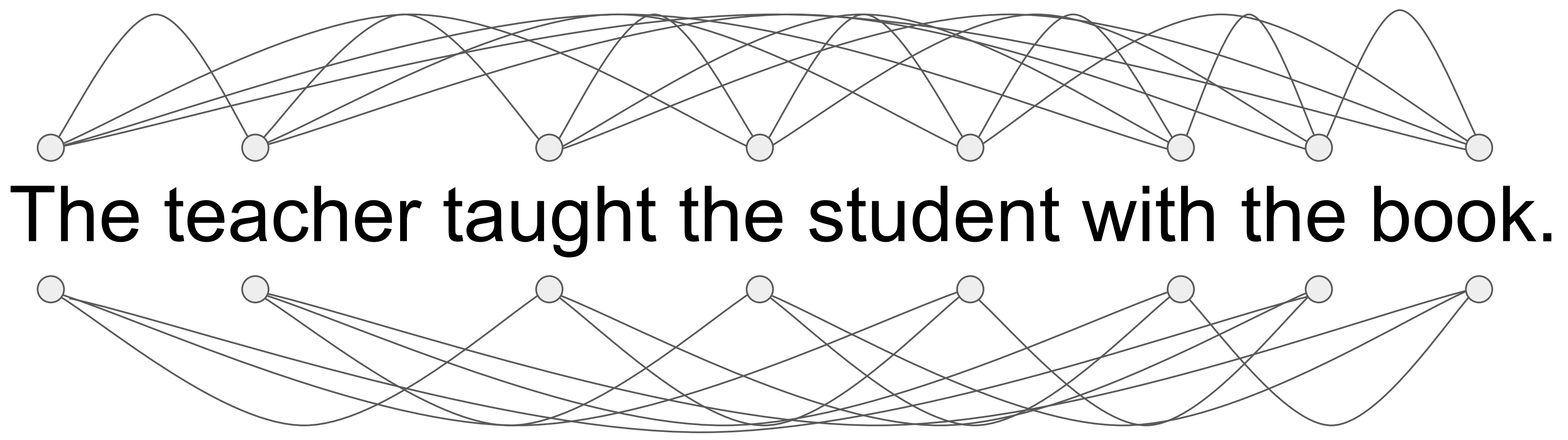


Transformers

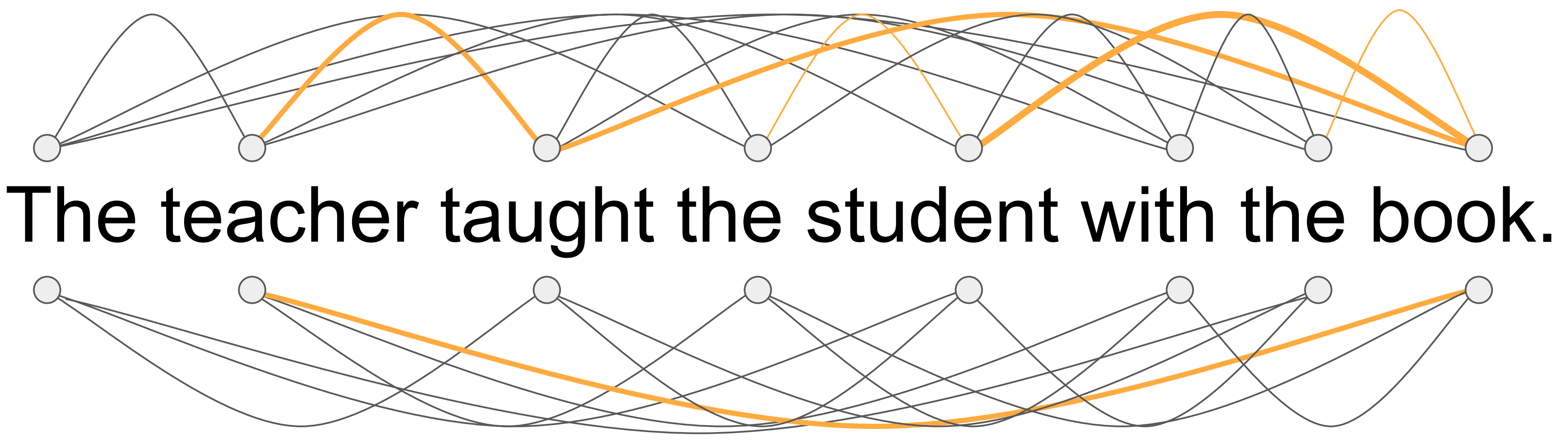


The teacher taught the student with the book.

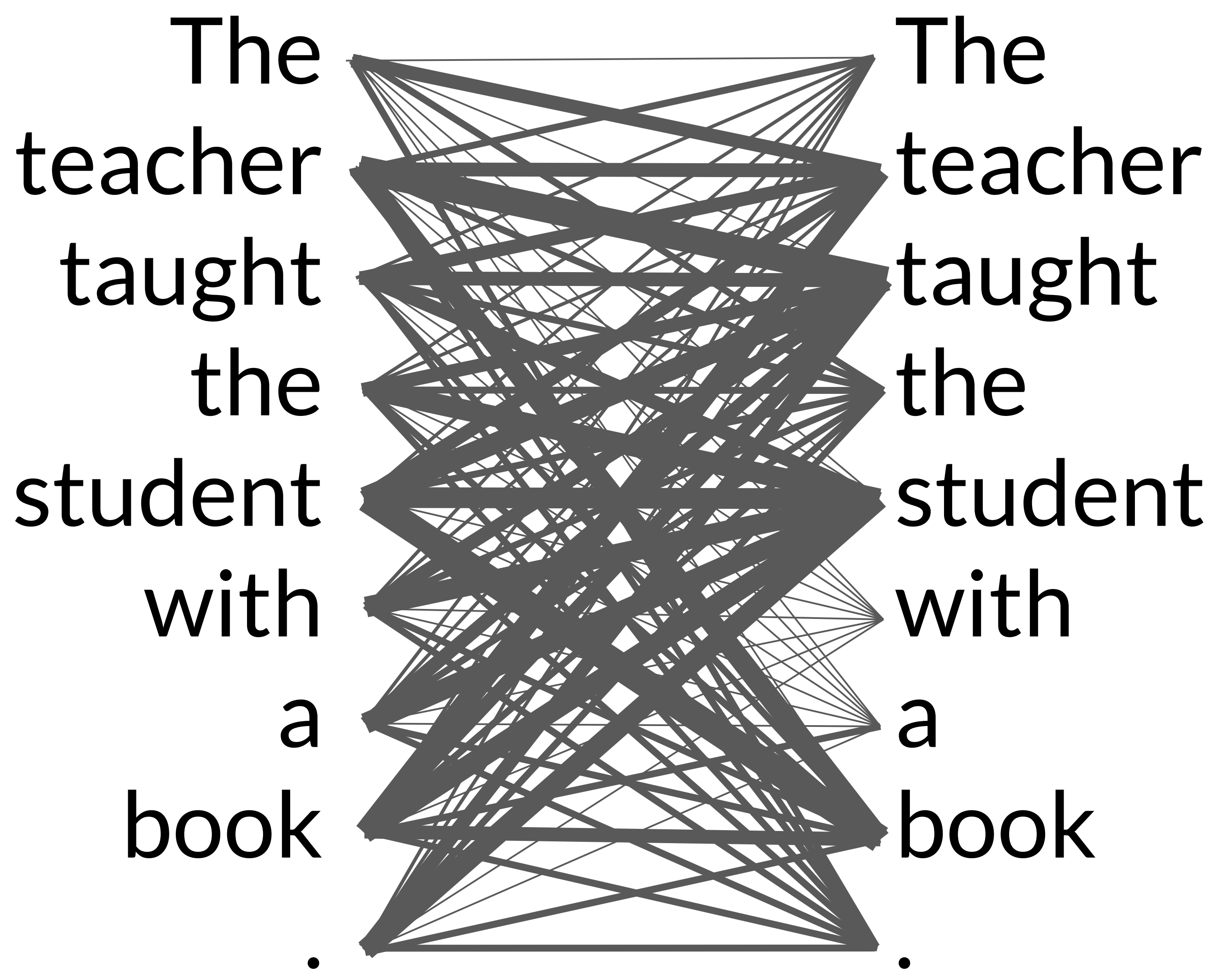
Transformers



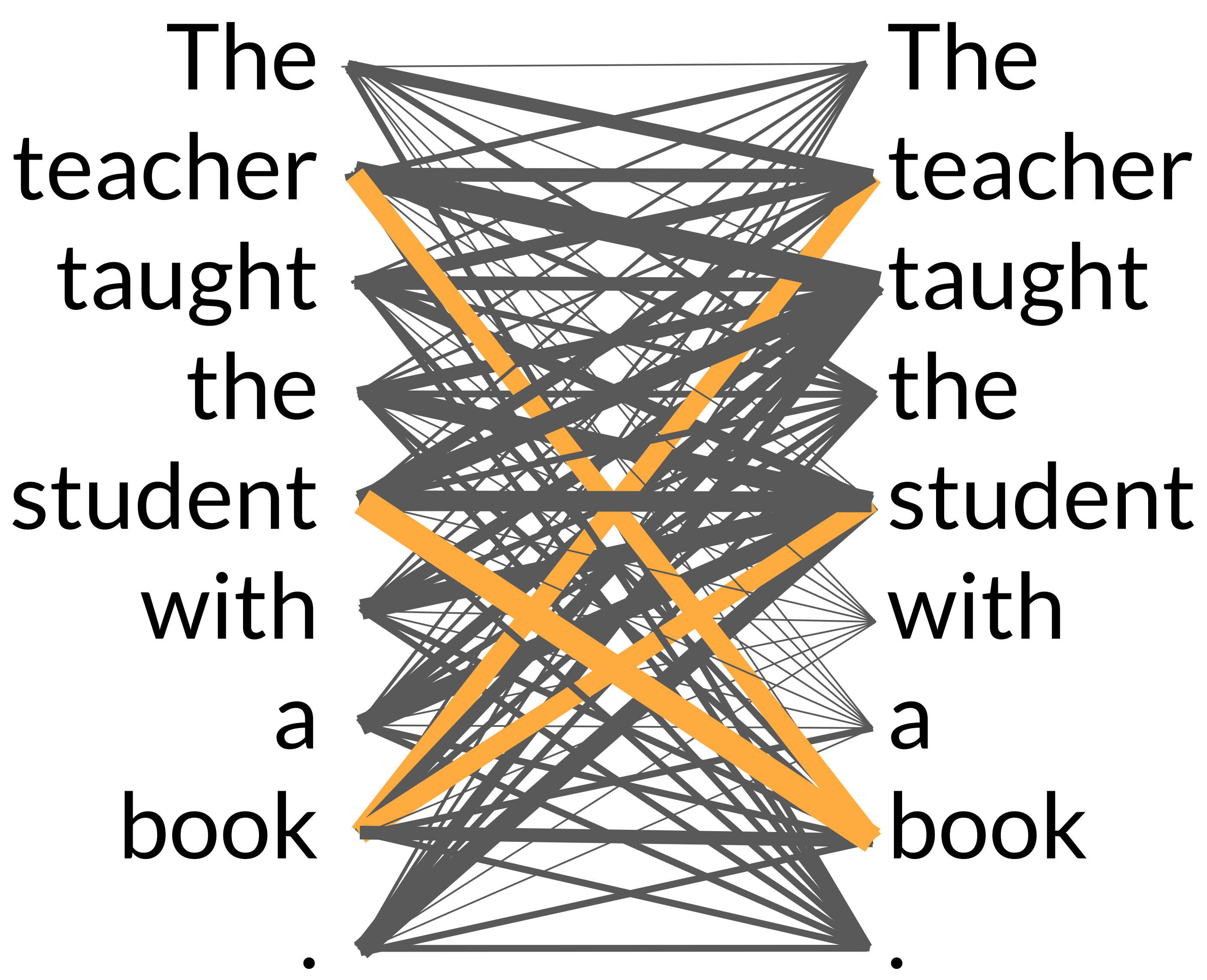
Transformers



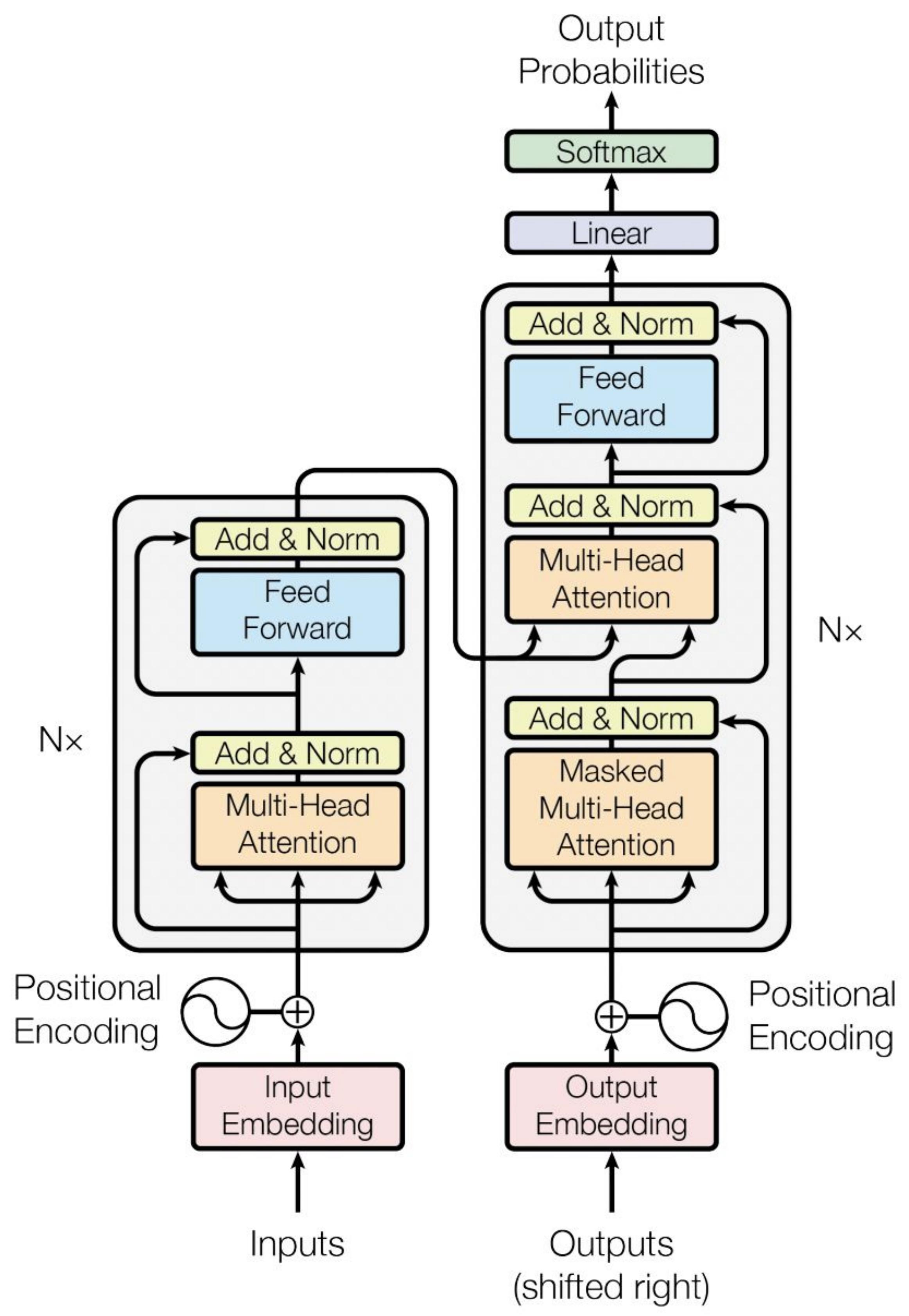
Self-attention



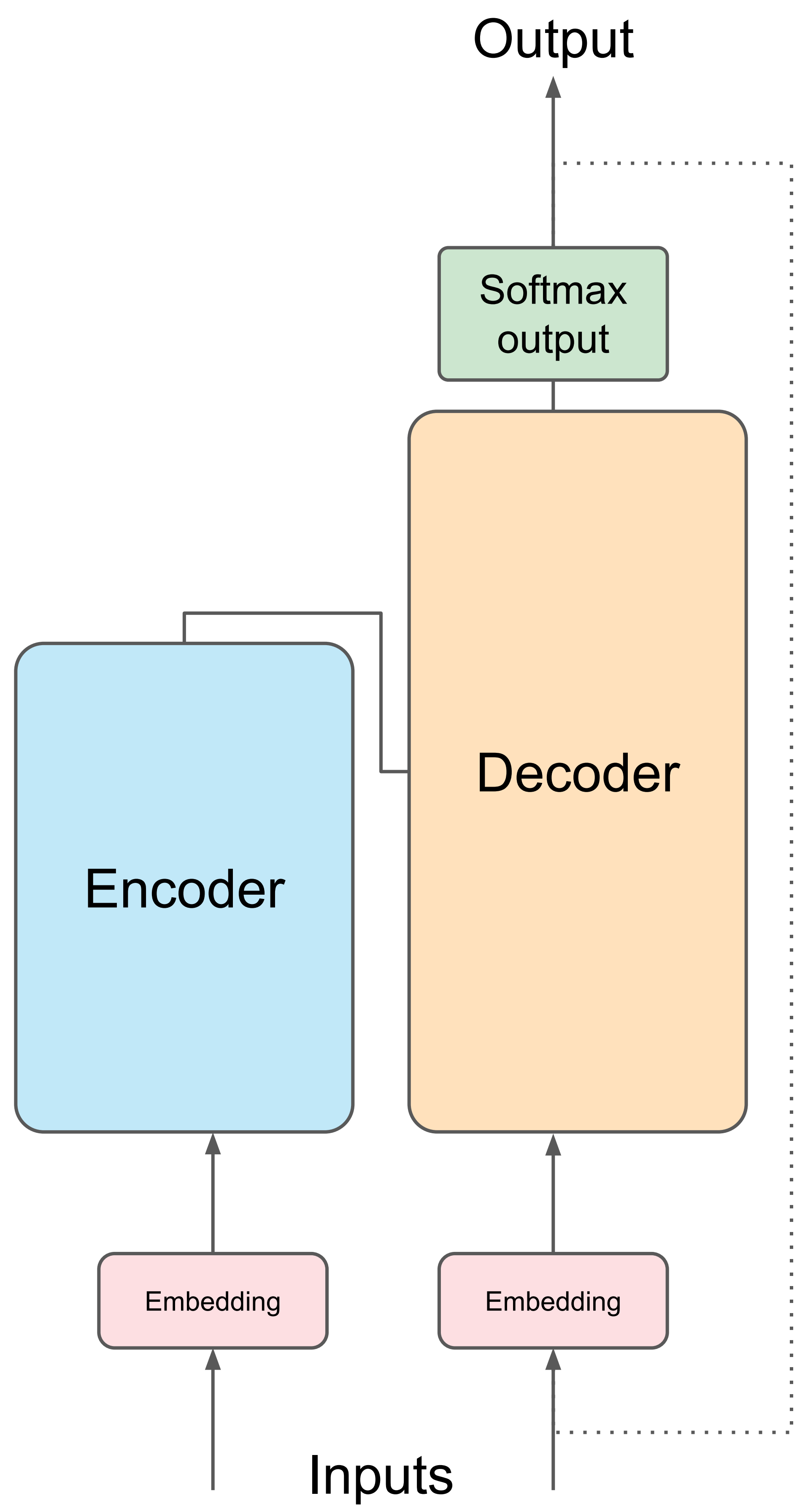
Self-attention



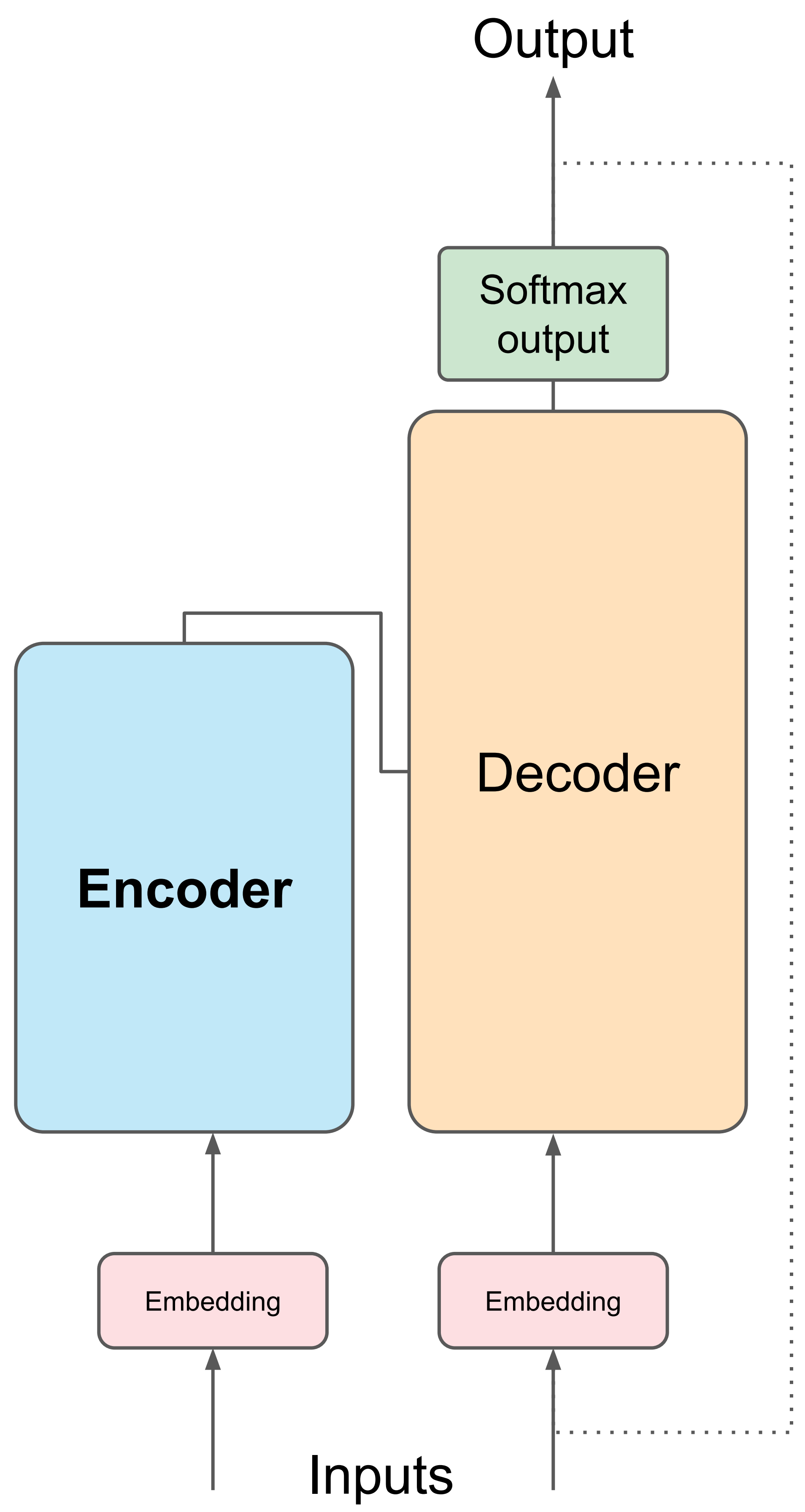
Transformers



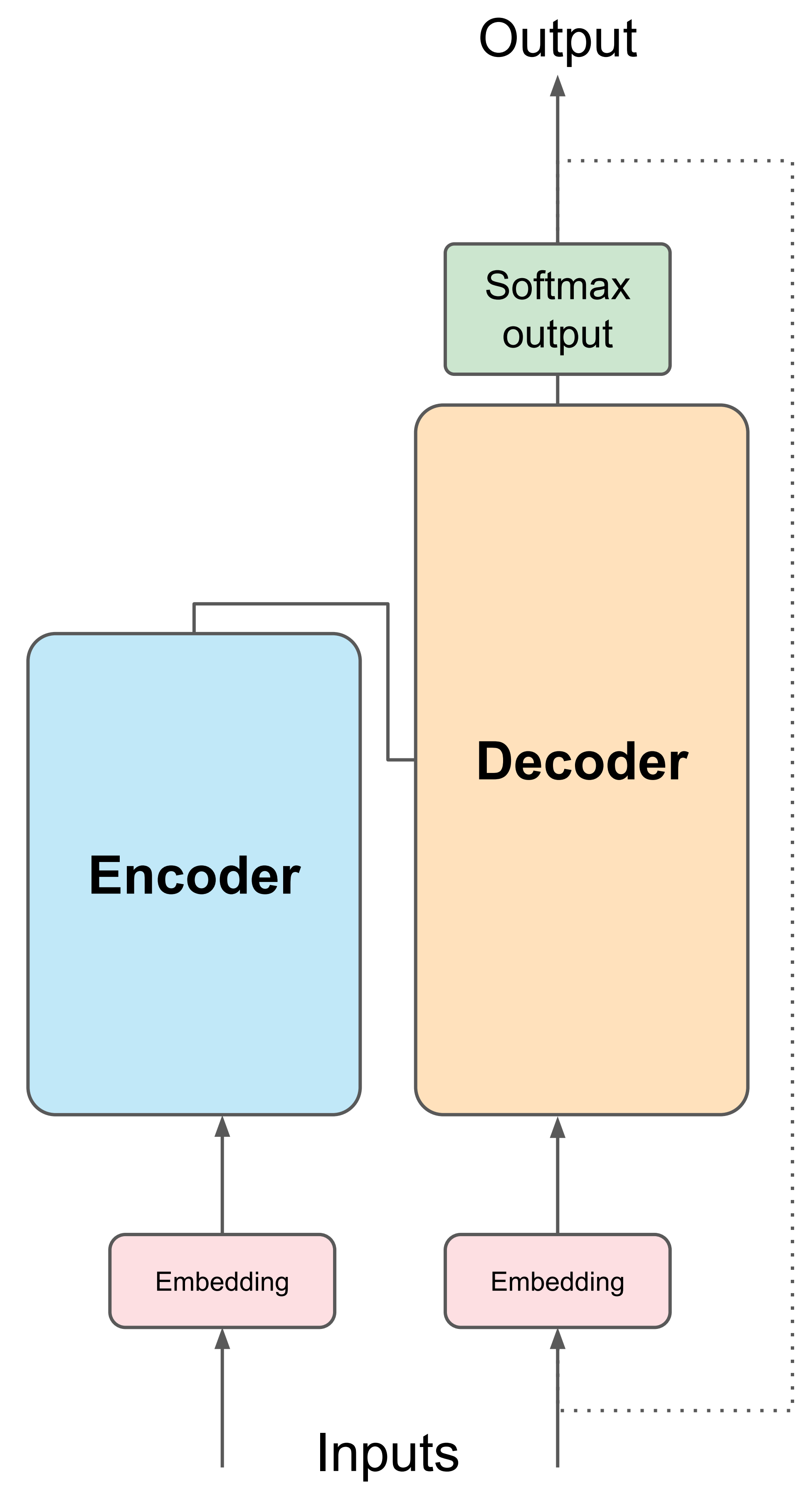
Transformers



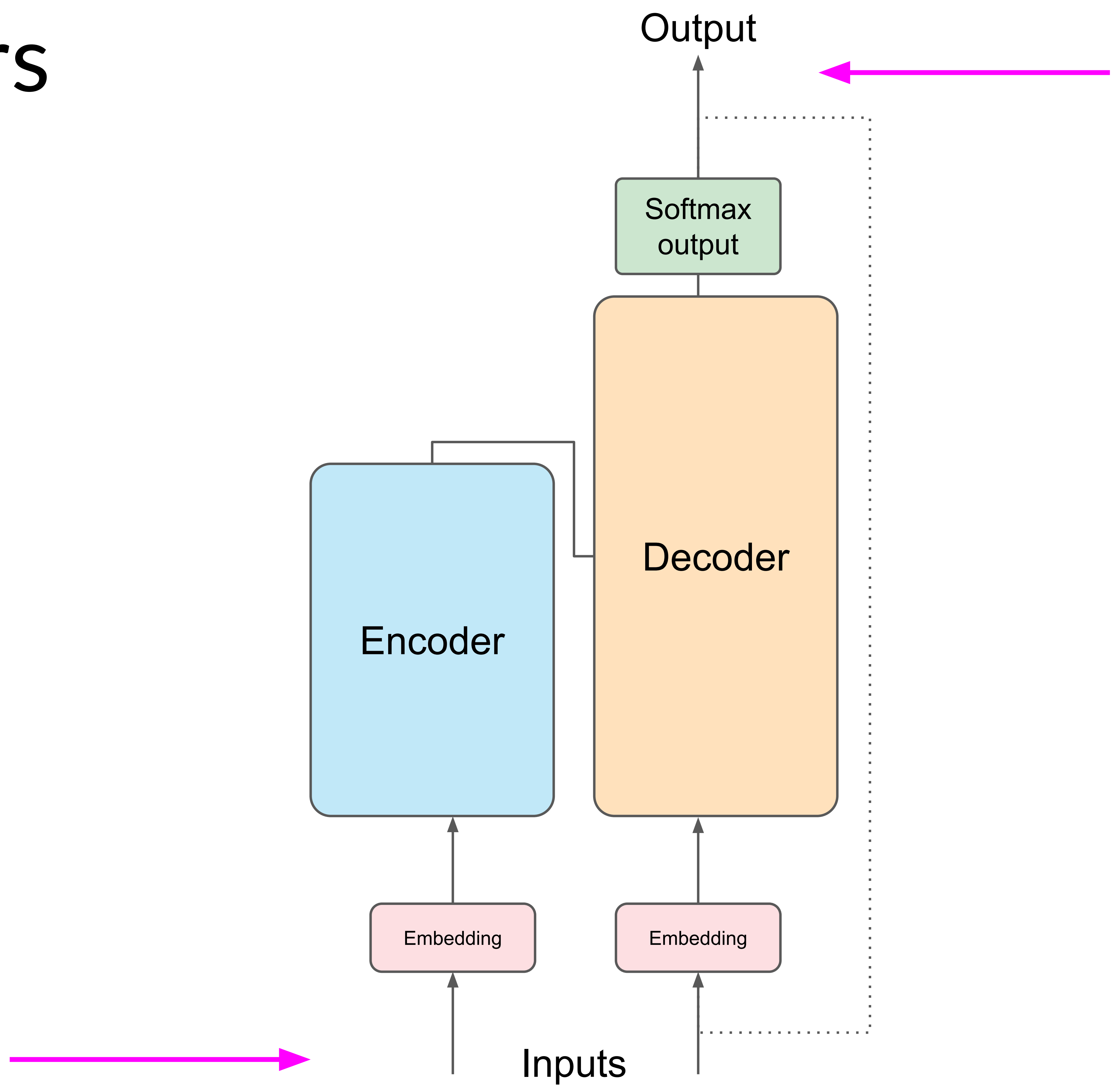
Transformers



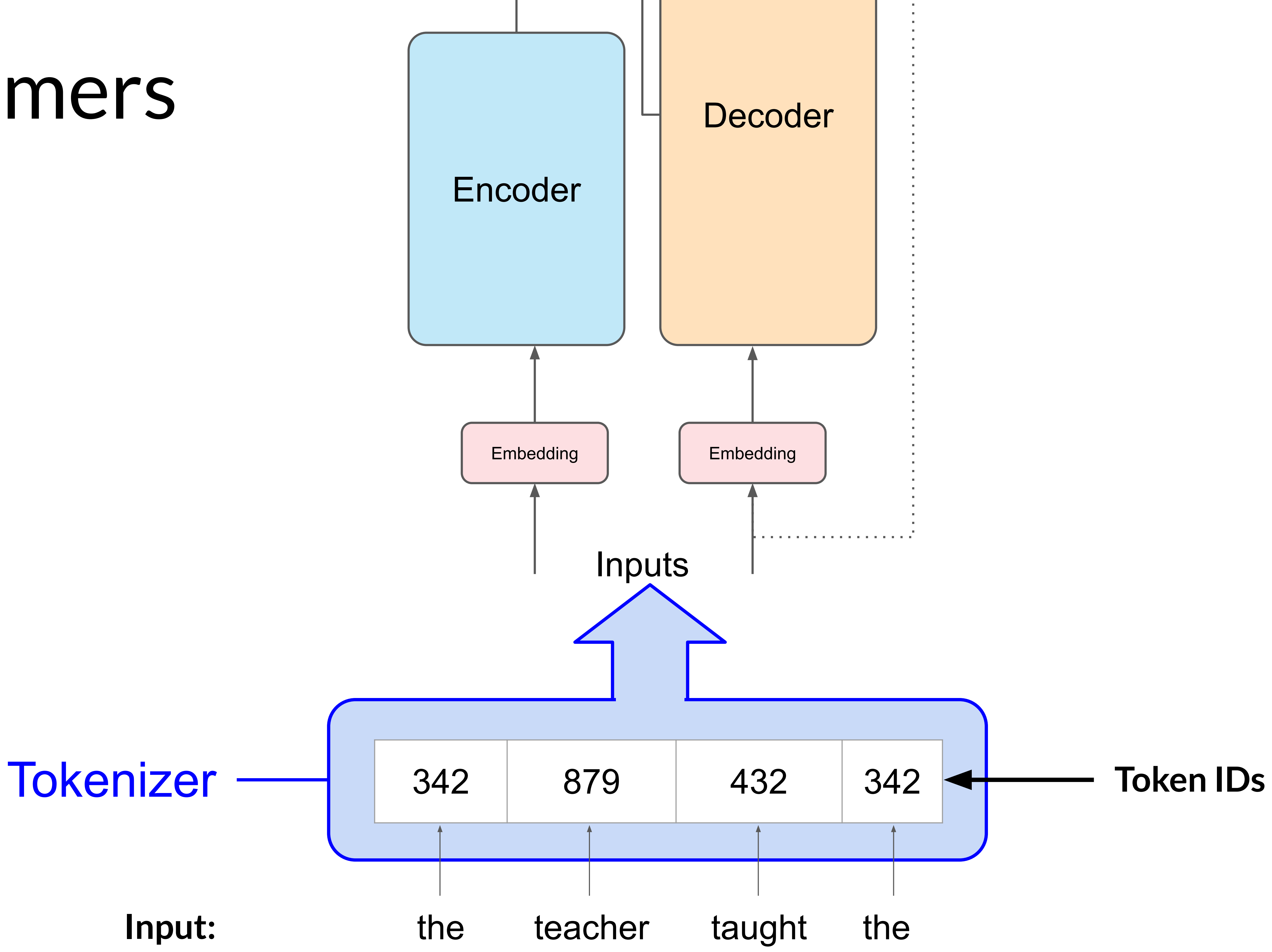
Transformers



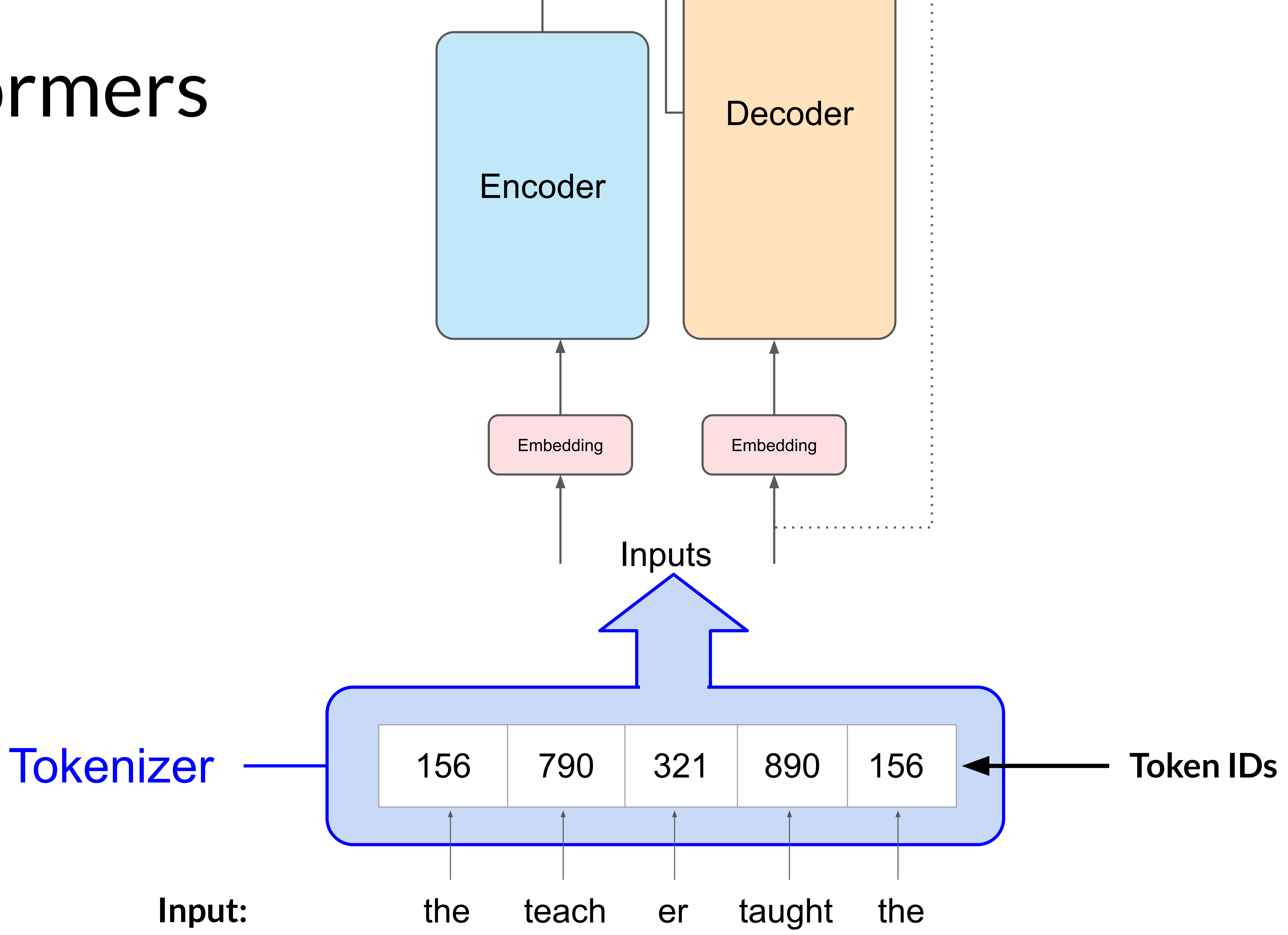
Transformers



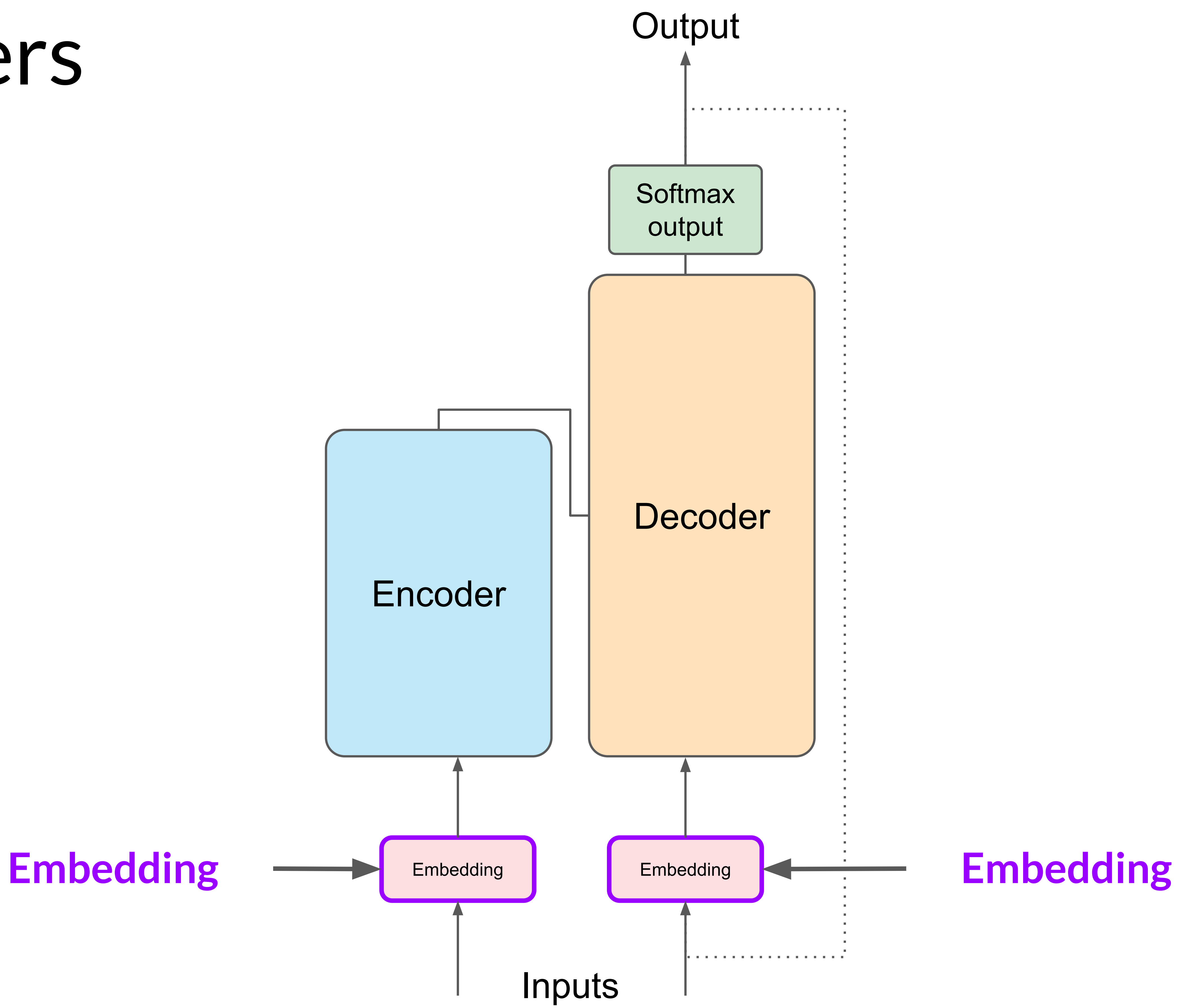
Transformers



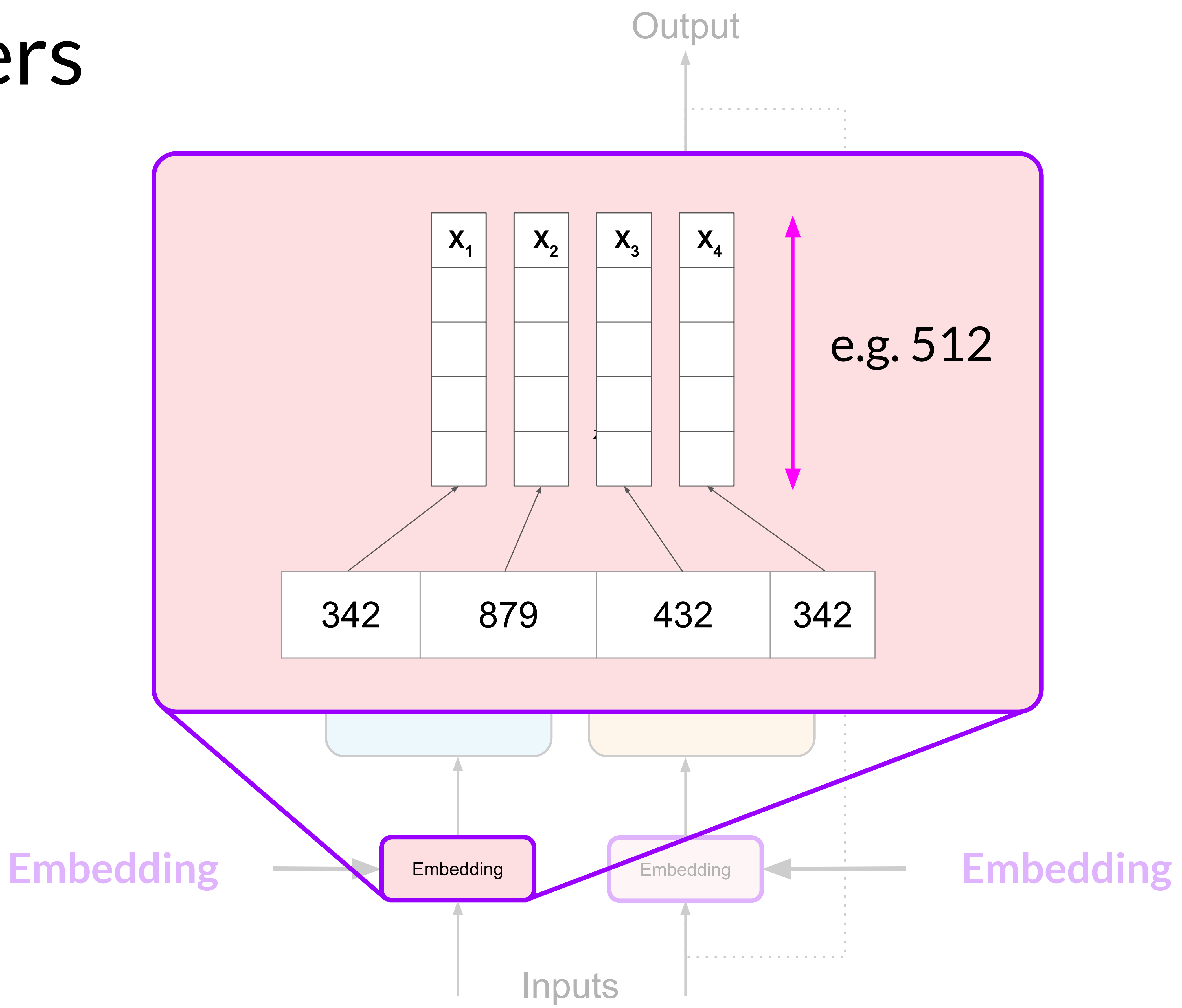
Transformers



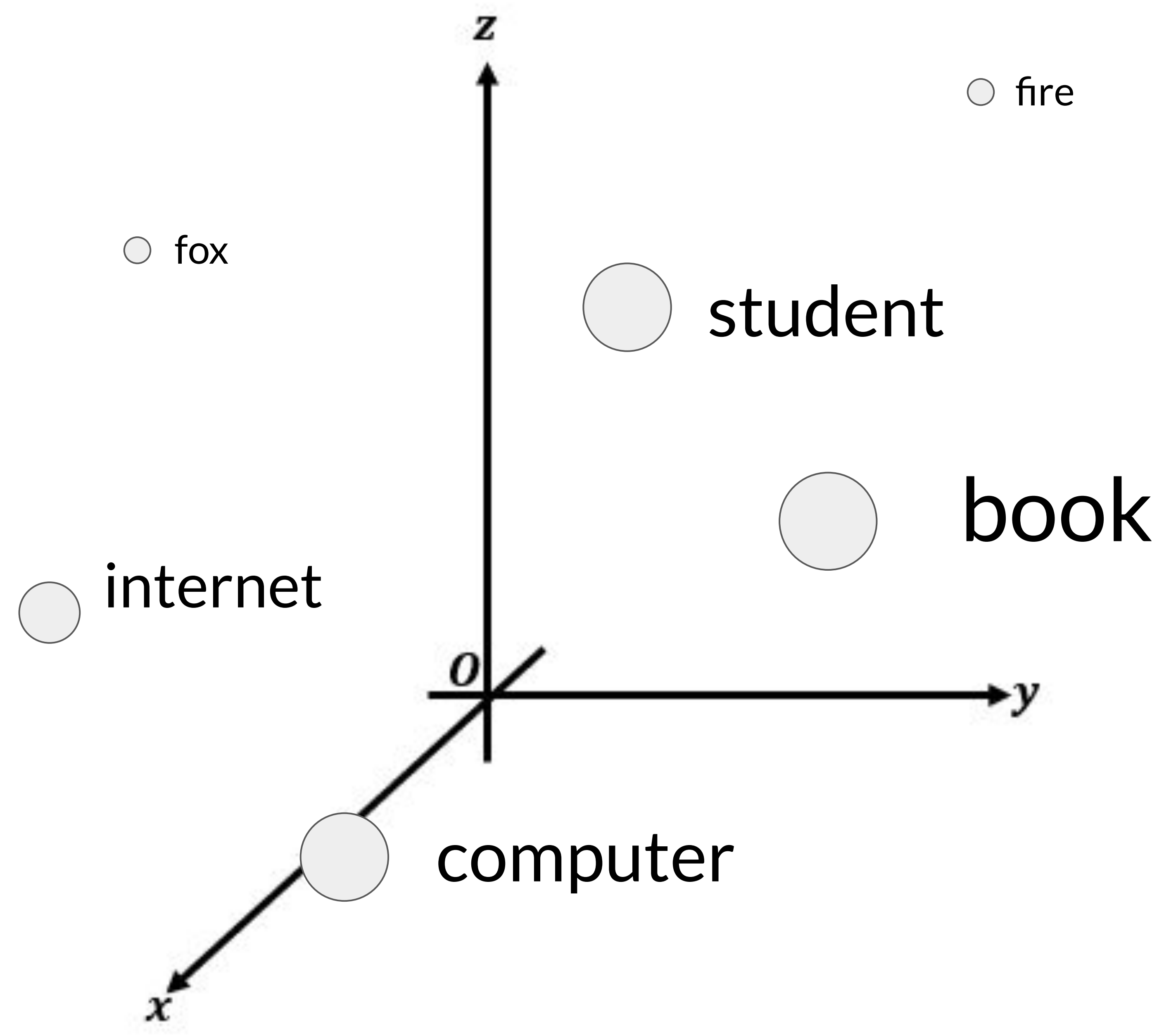
Transformers



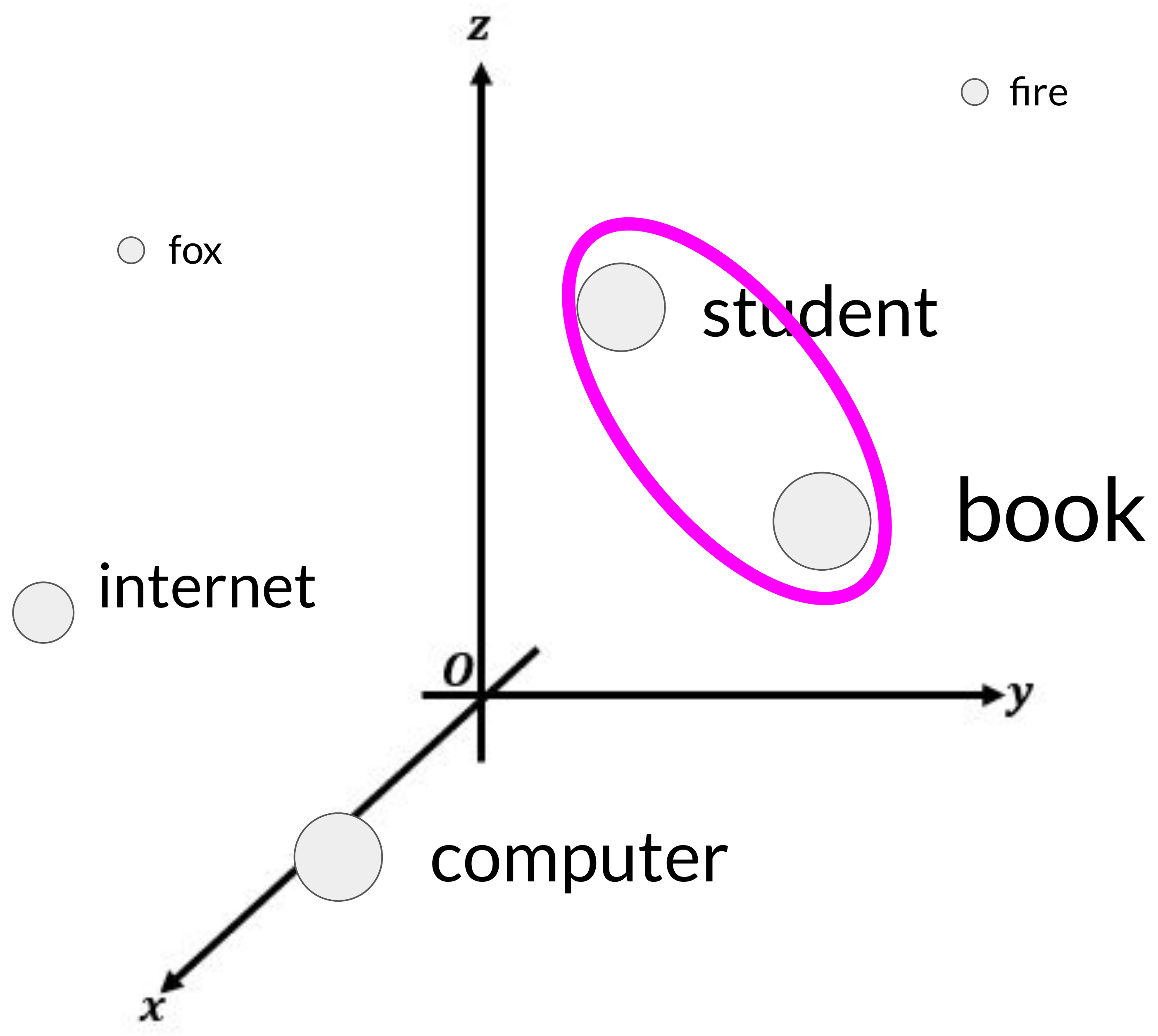
Transformers



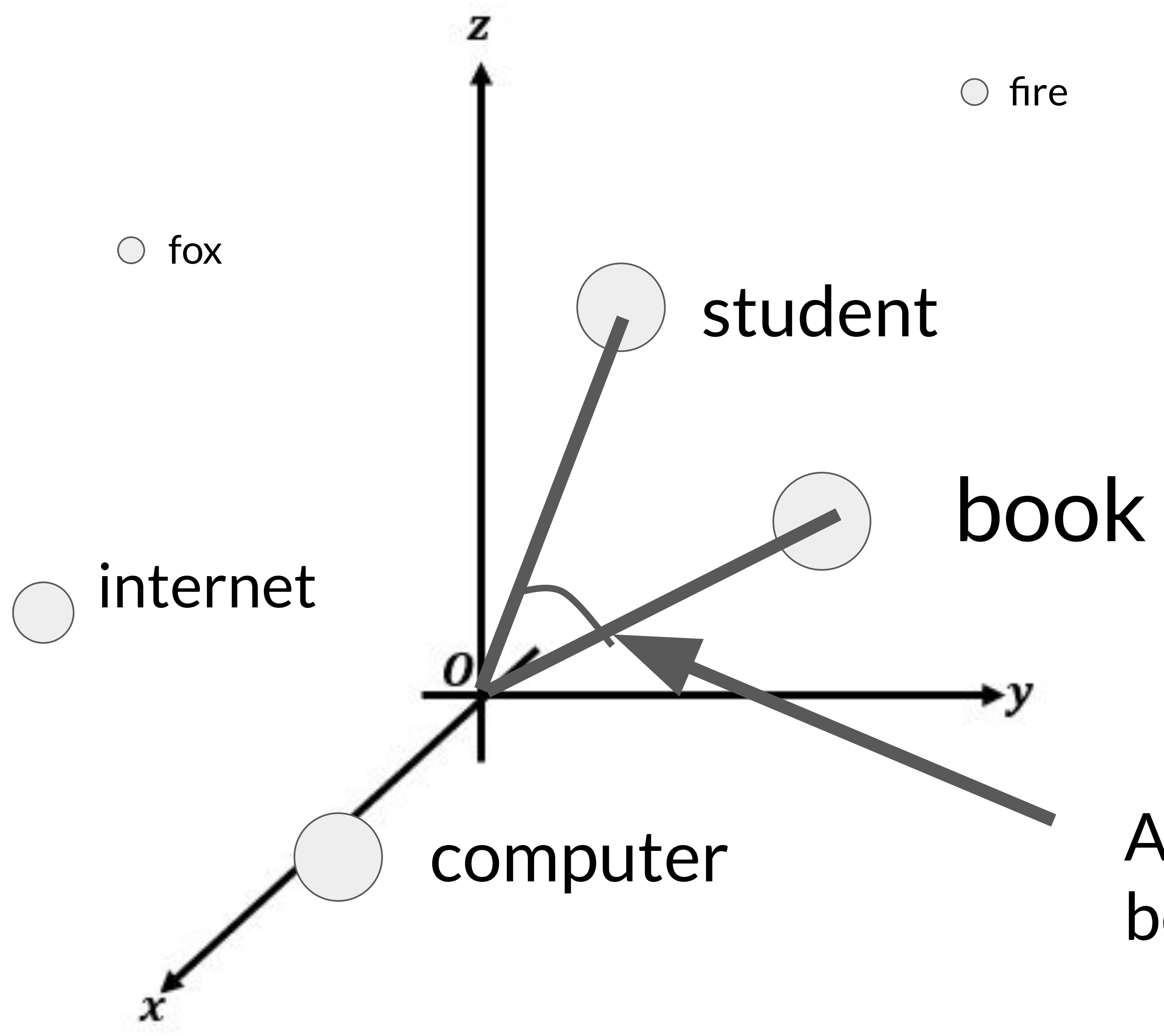
Transformers



Transformers

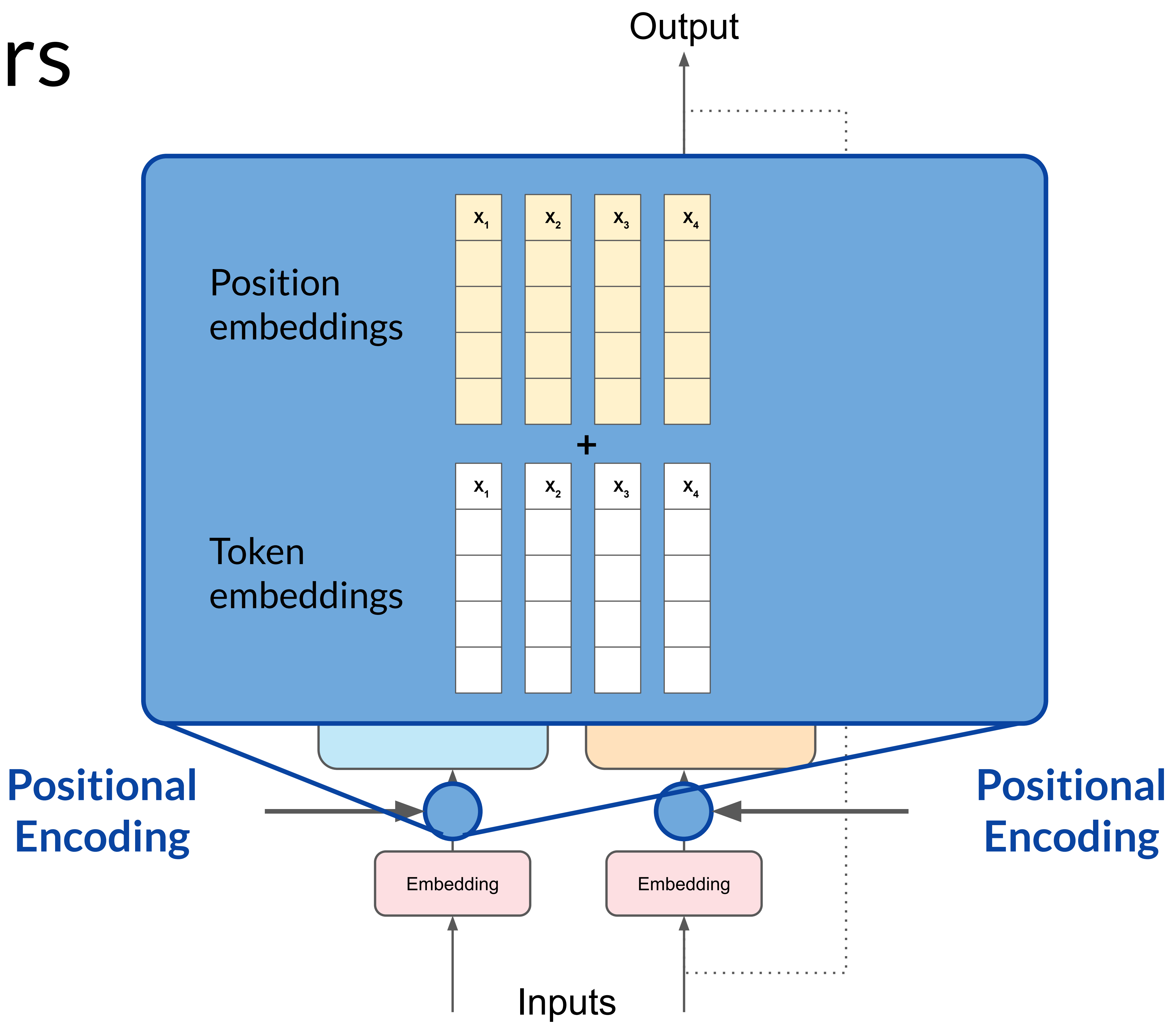


Transformers

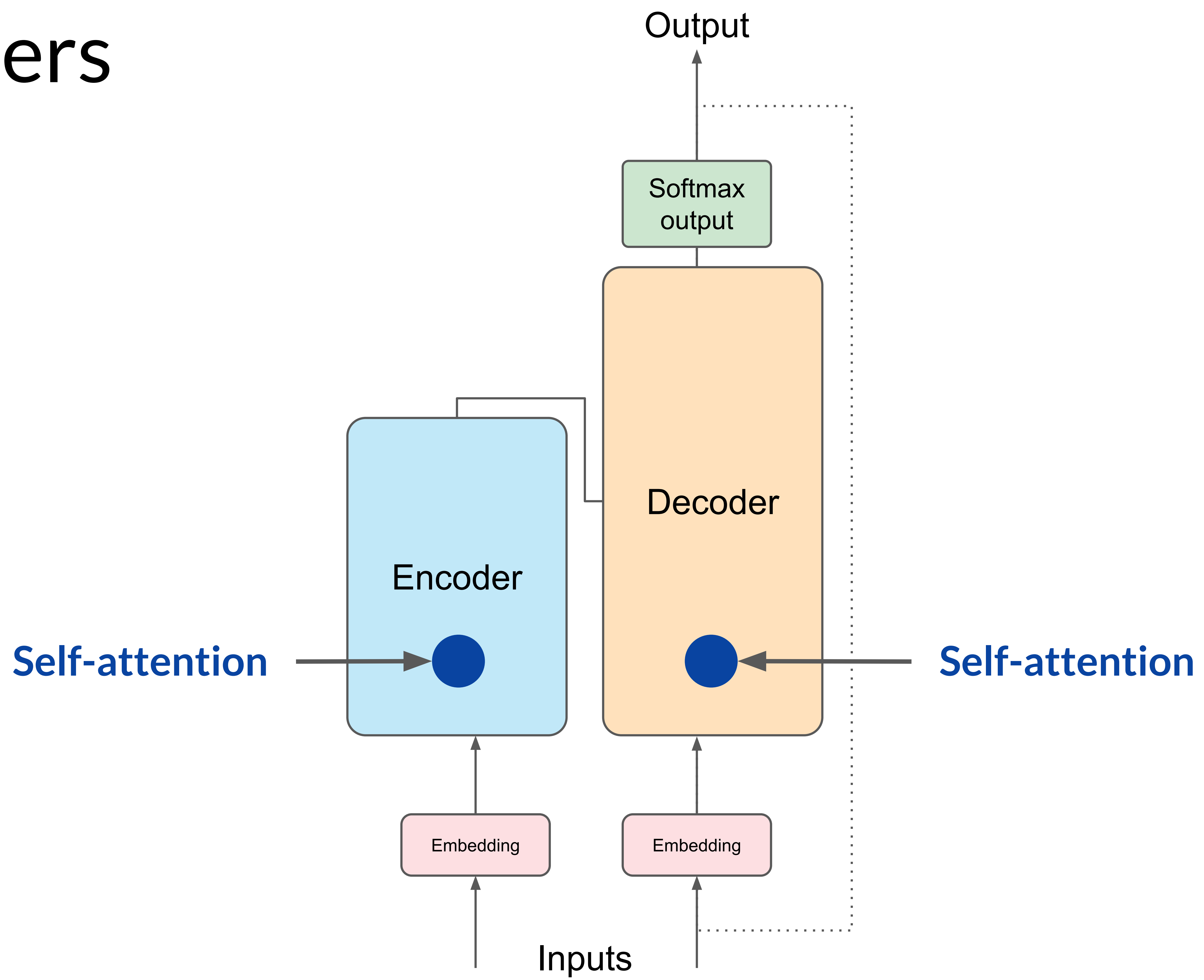


Angle measures distance between words

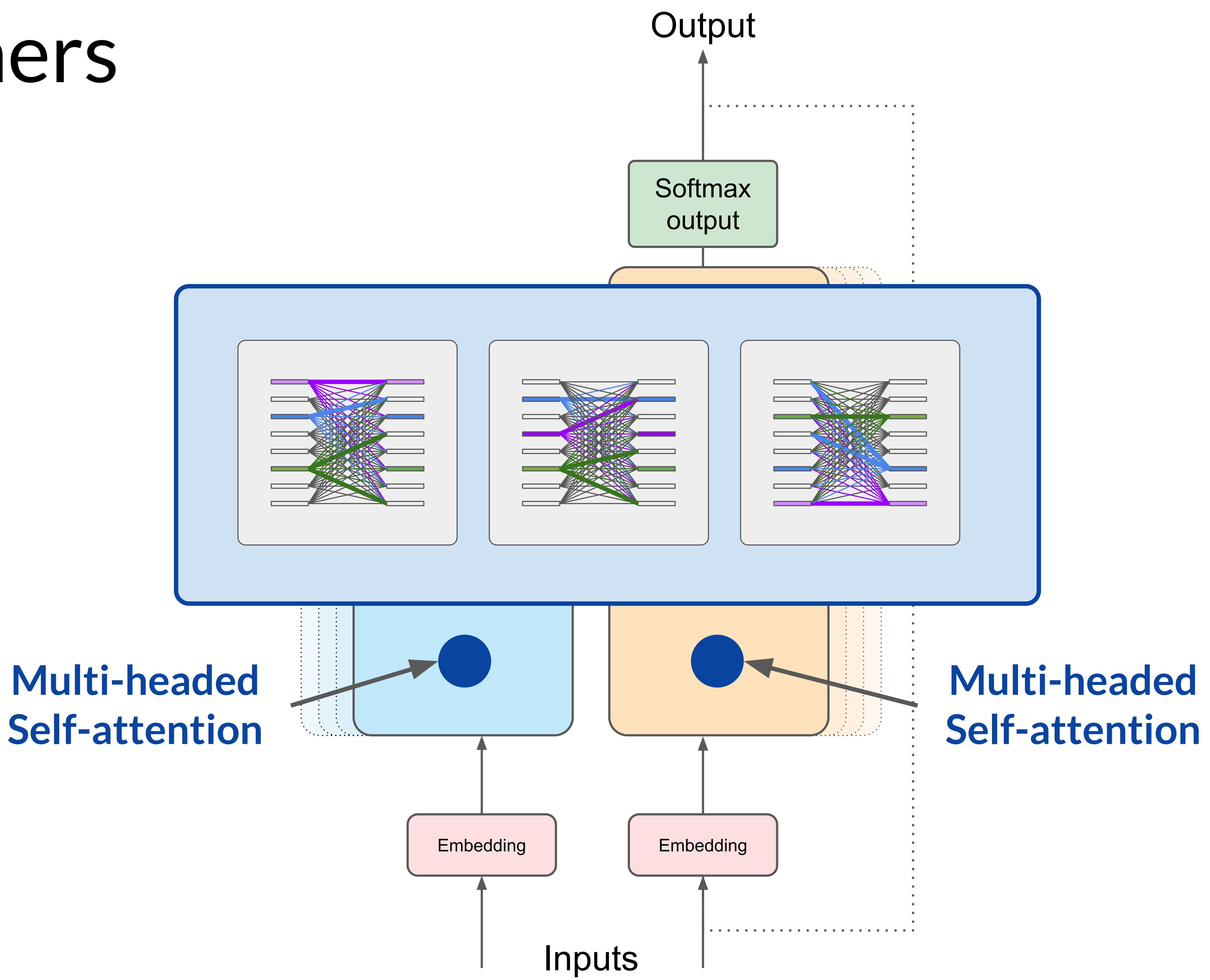
Transformers



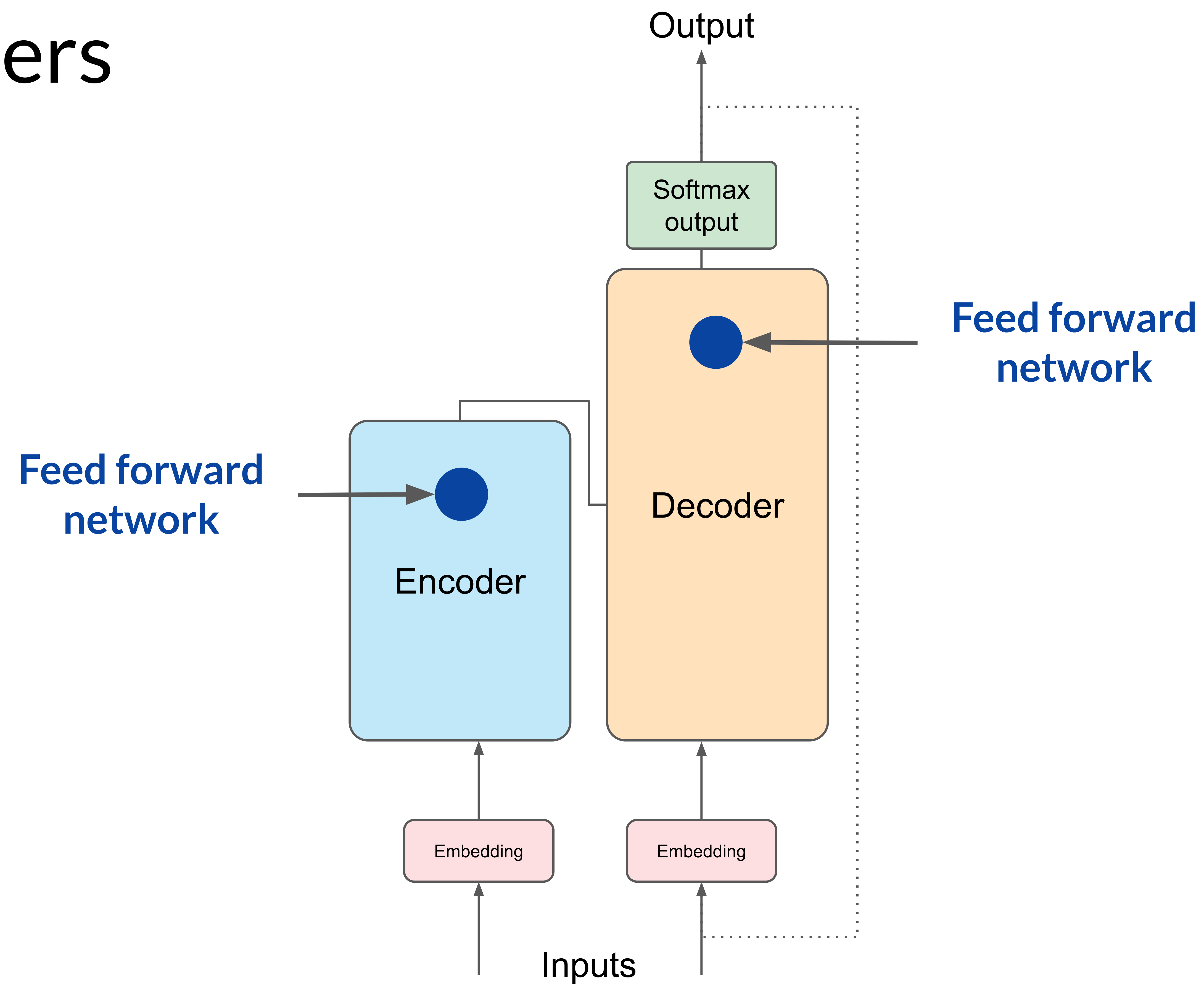
Transformers



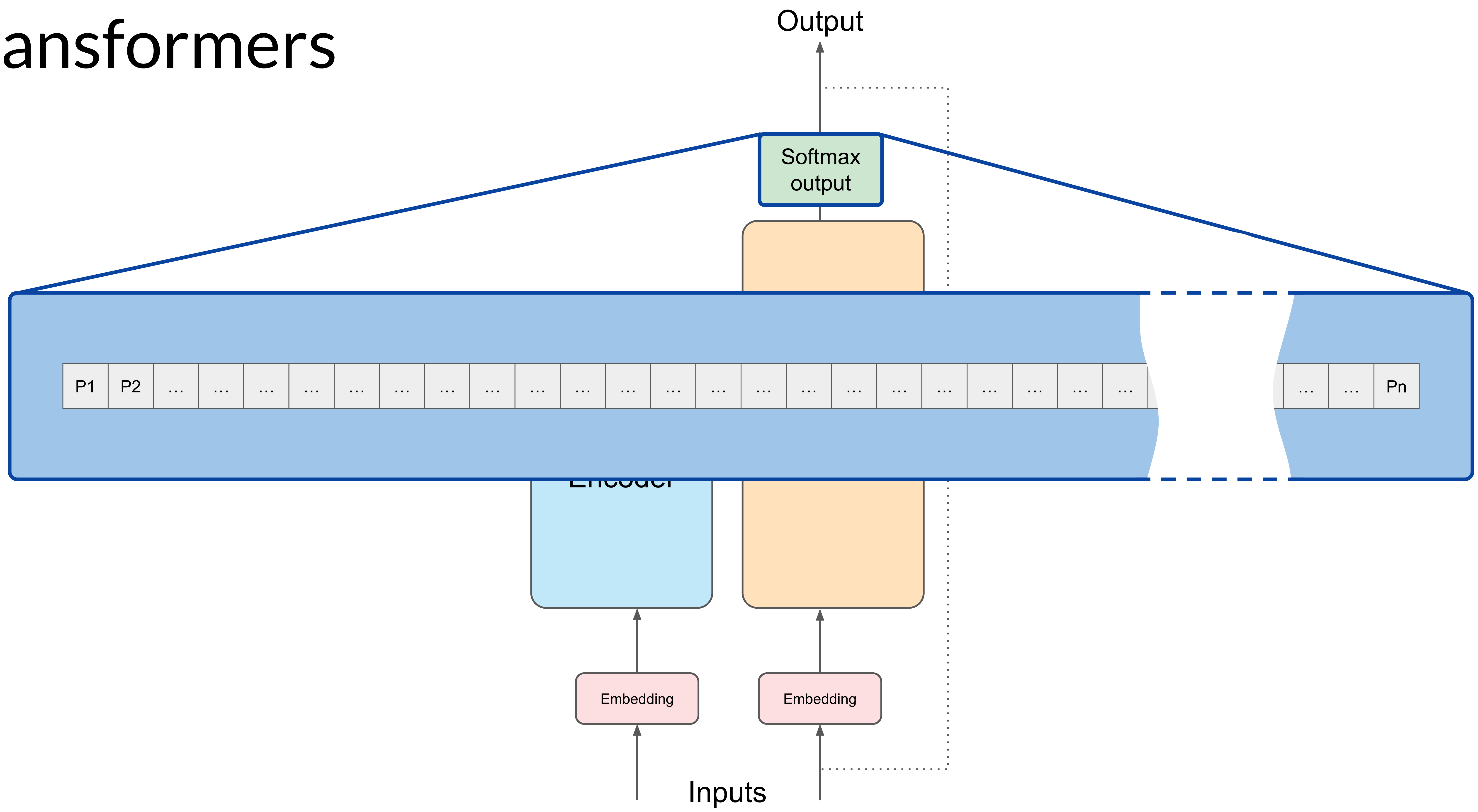
Transformers



Transformers



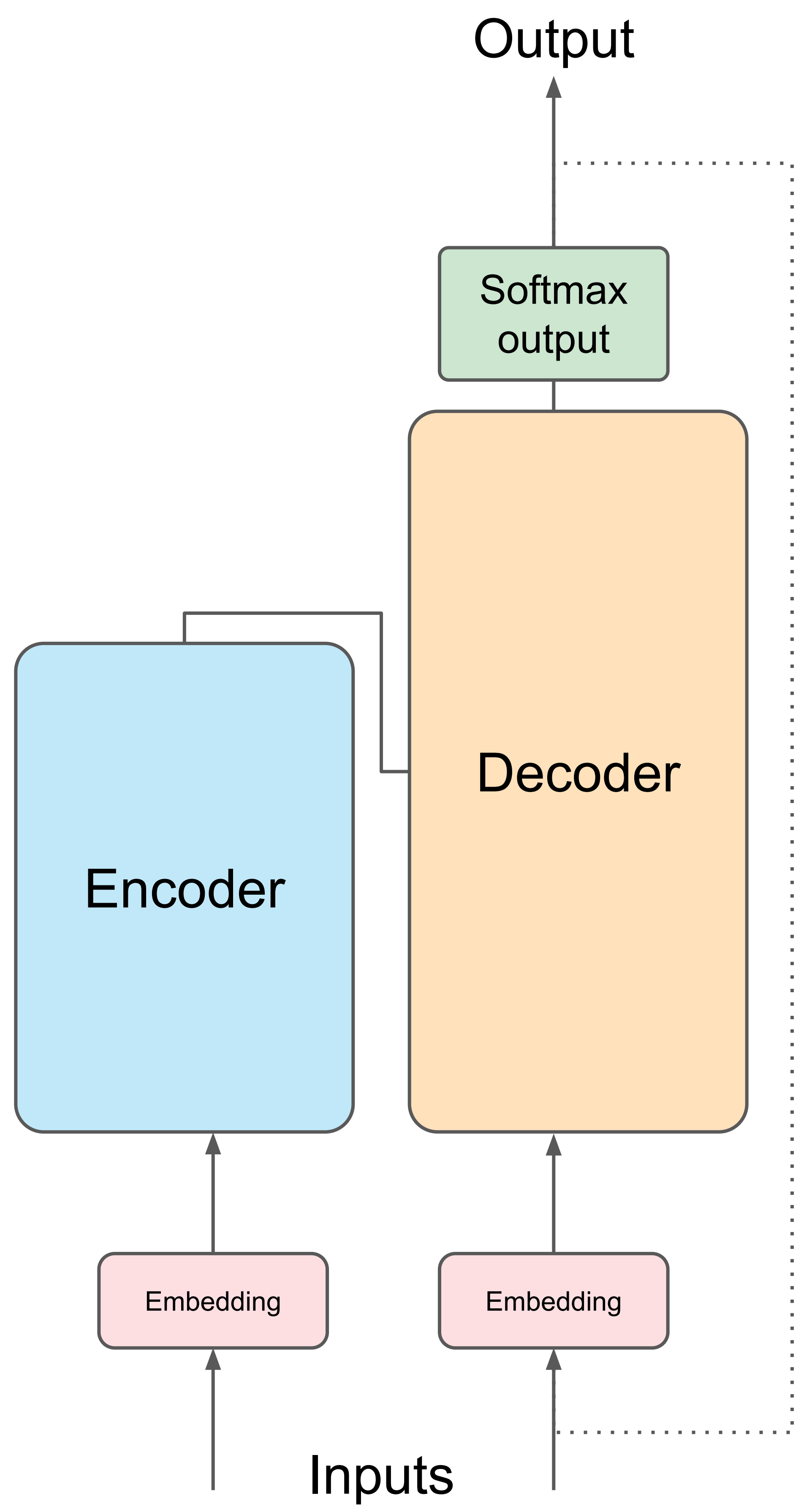
Transformers



Transformers

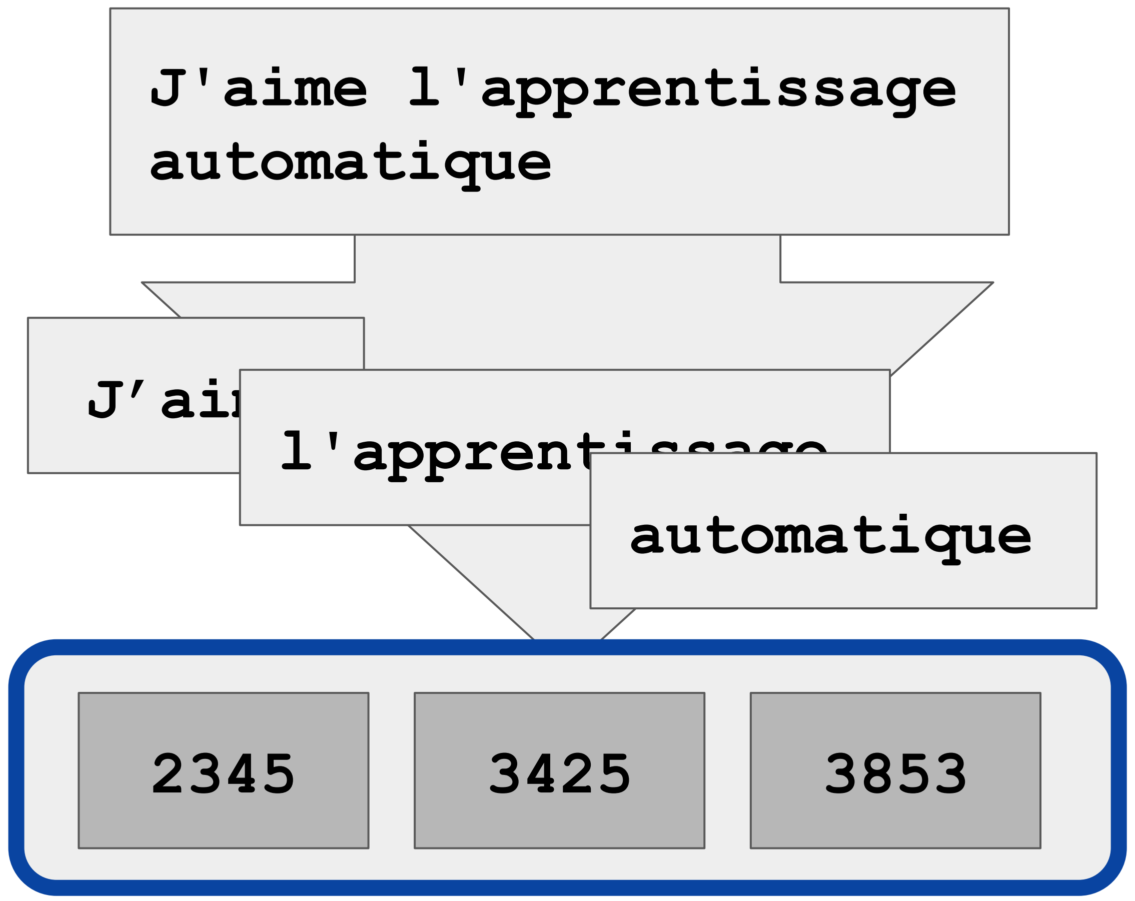
Translation:
sequence-to-sequence task

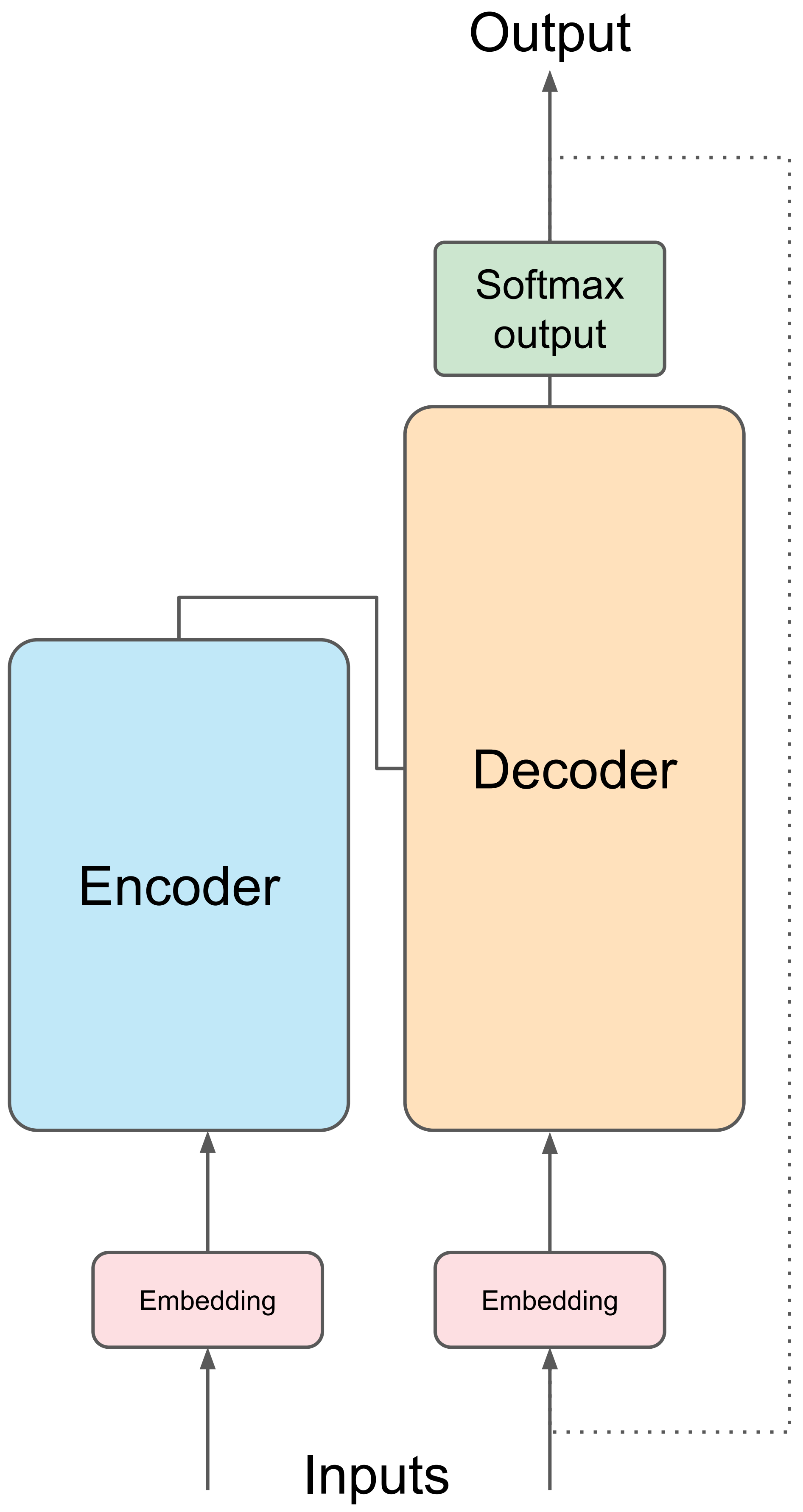
```
J'aime l'apprentissage
automatique
```



Transformers

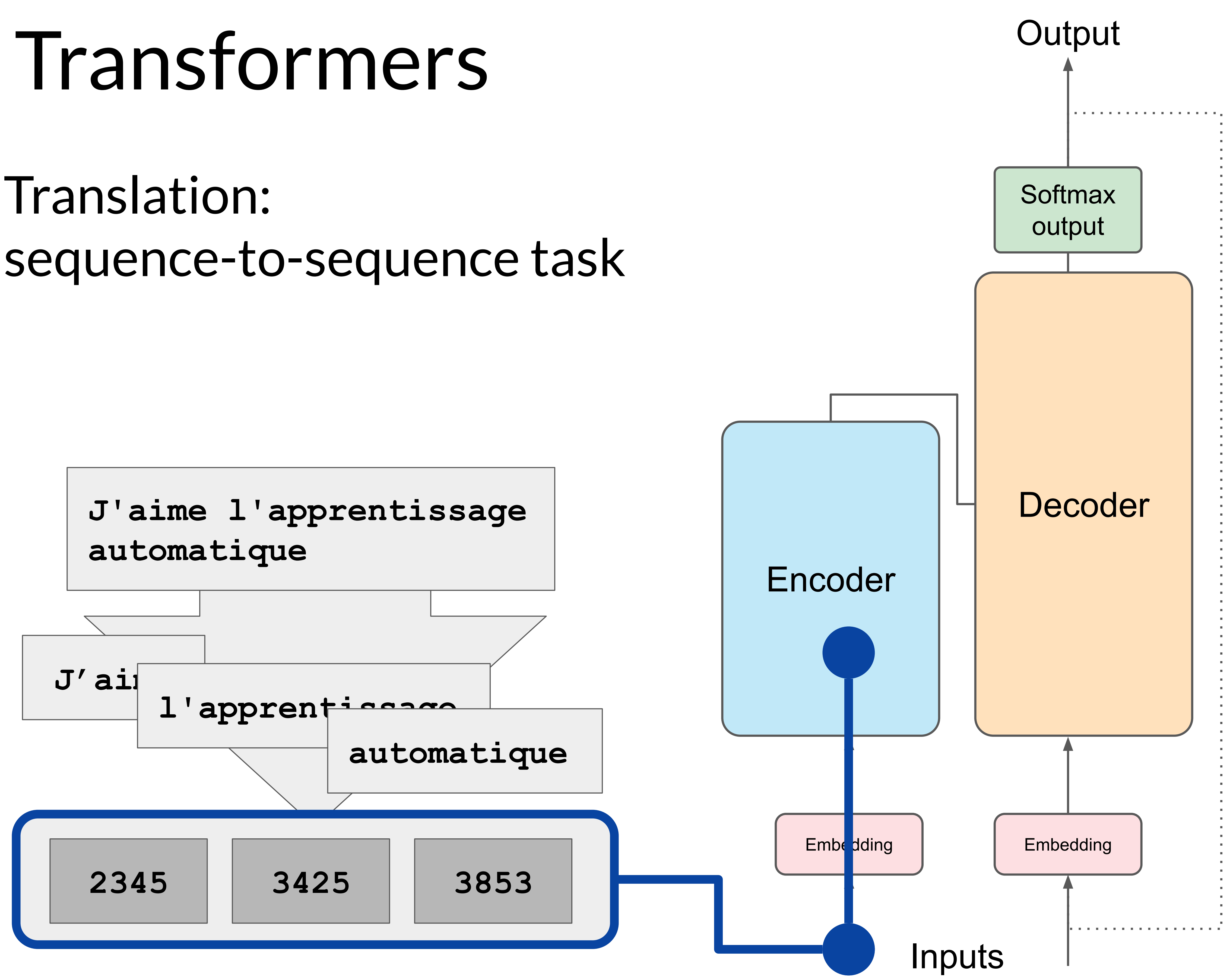
Translation:
sequence-to-sequence task





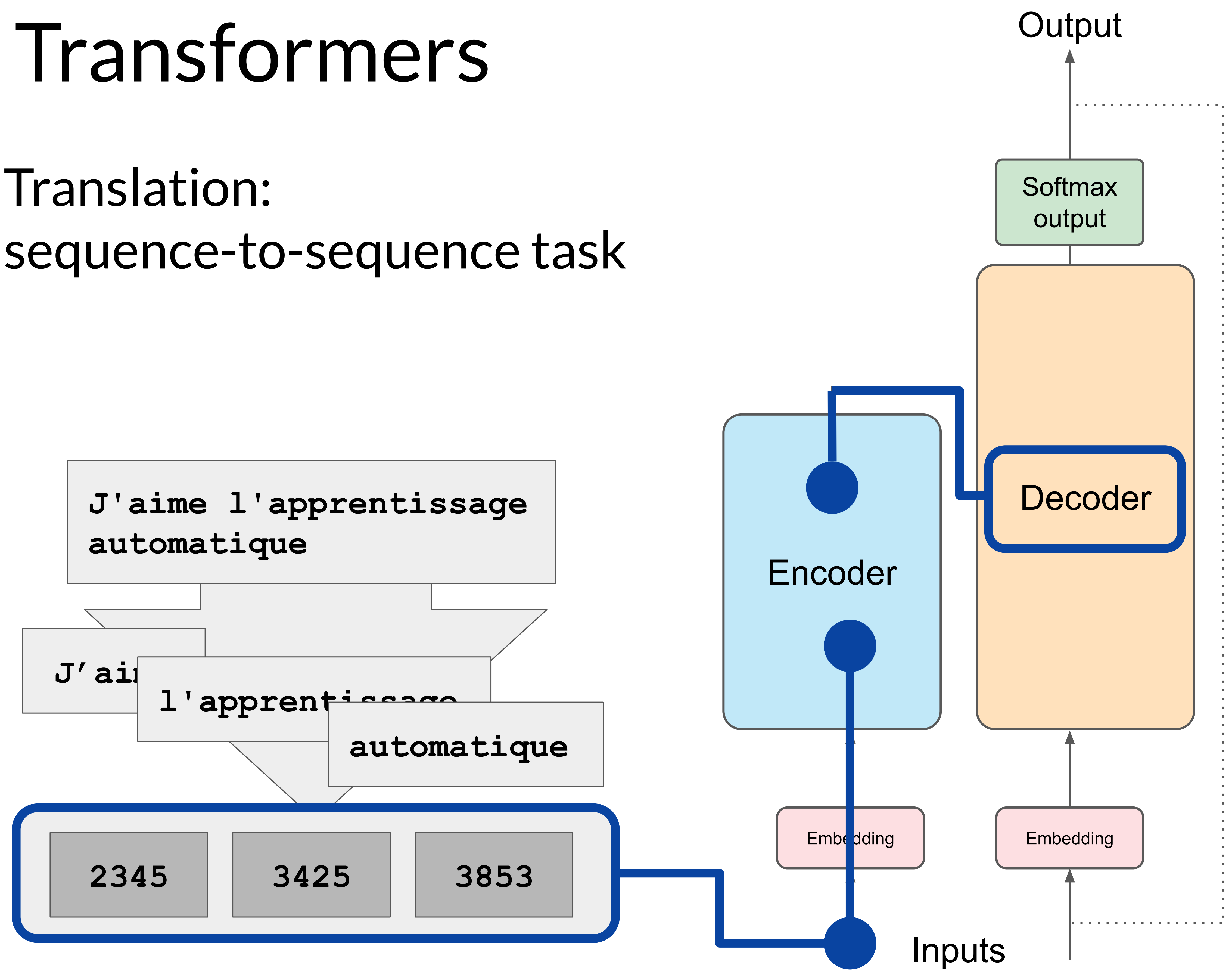
Transformers

Translation:
sequence-to-sequence task



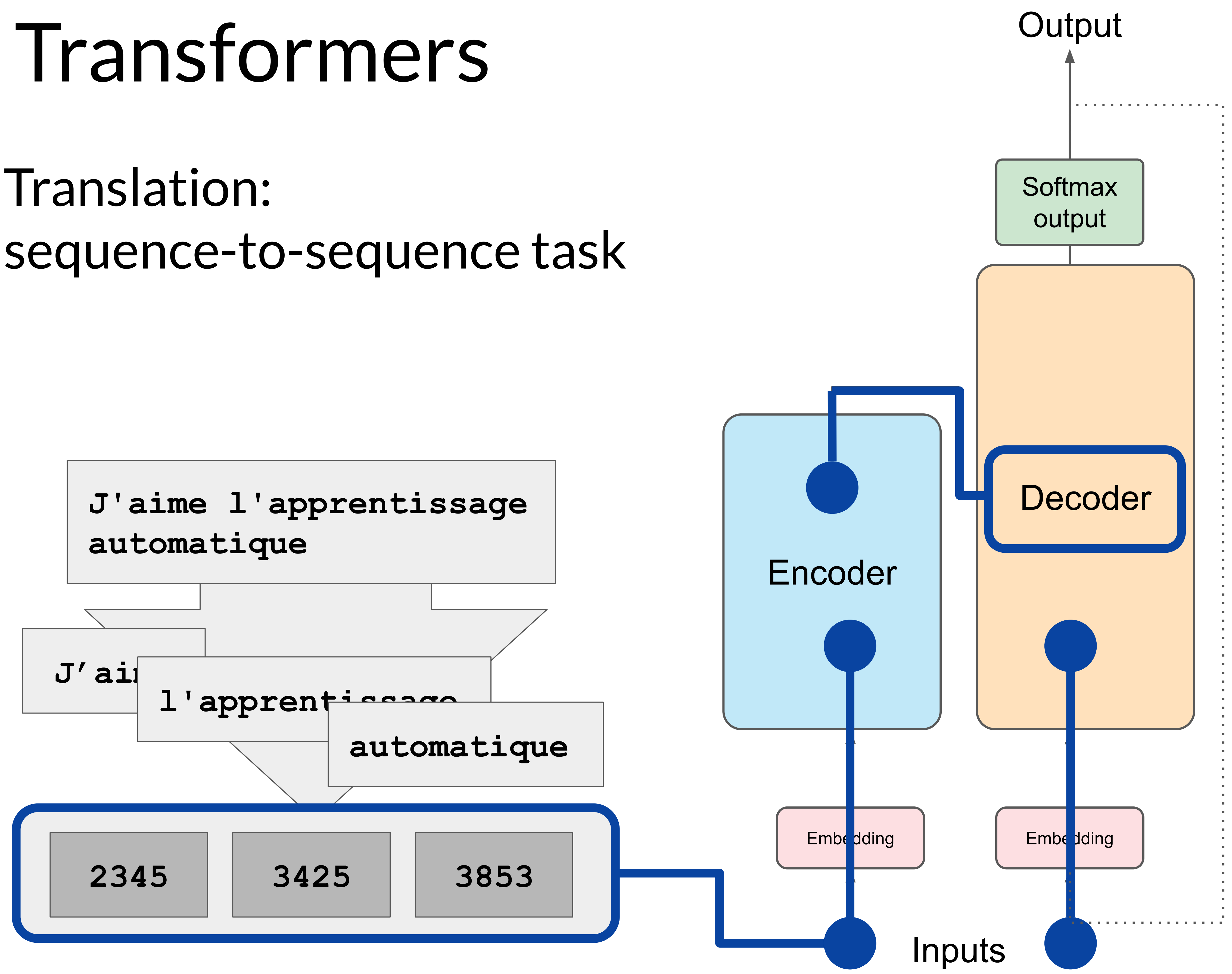
Transformers

Translation:
sequence-to-sequence task



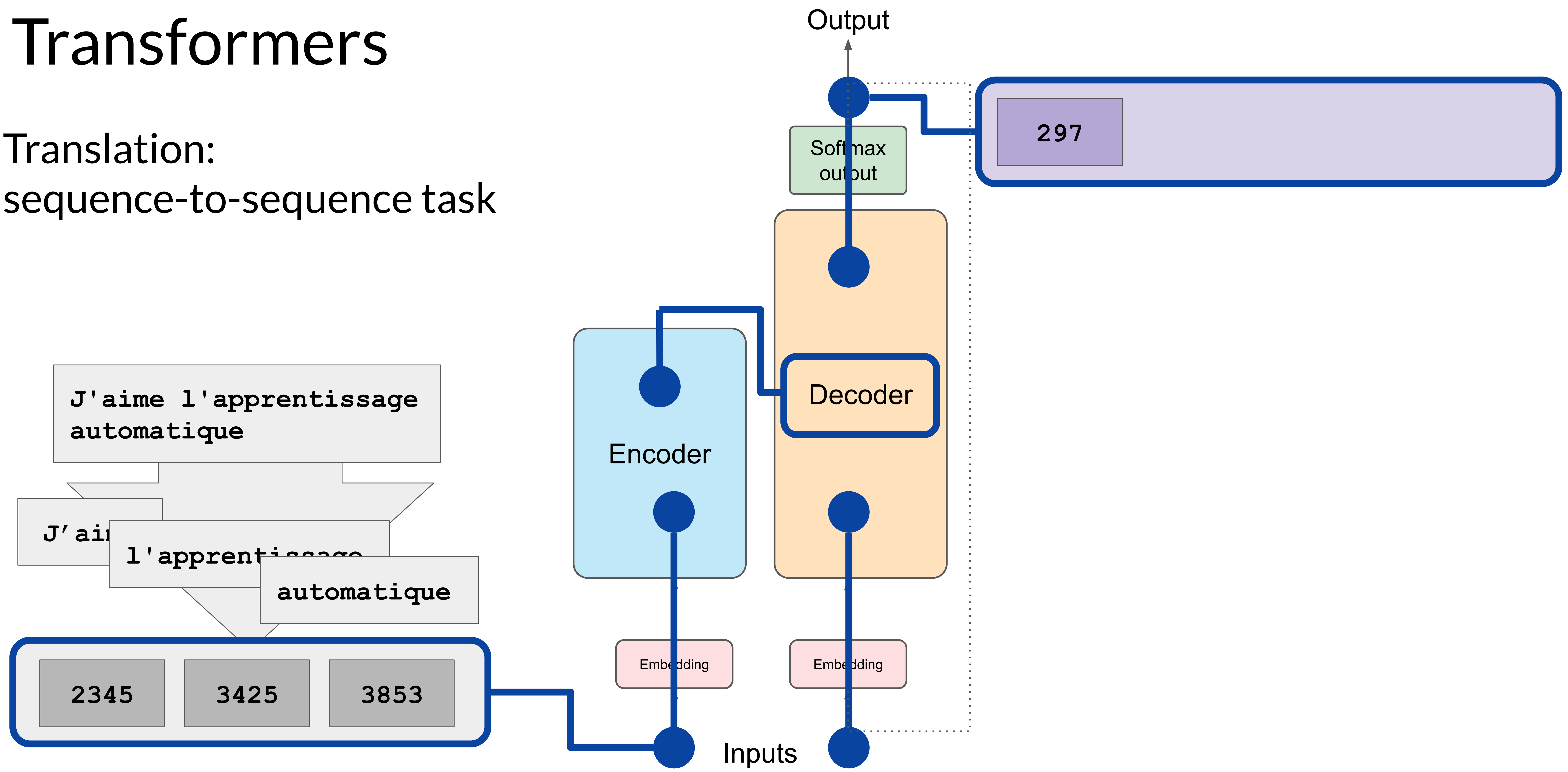
Transformers

Translation:
sequence-to-sequence task



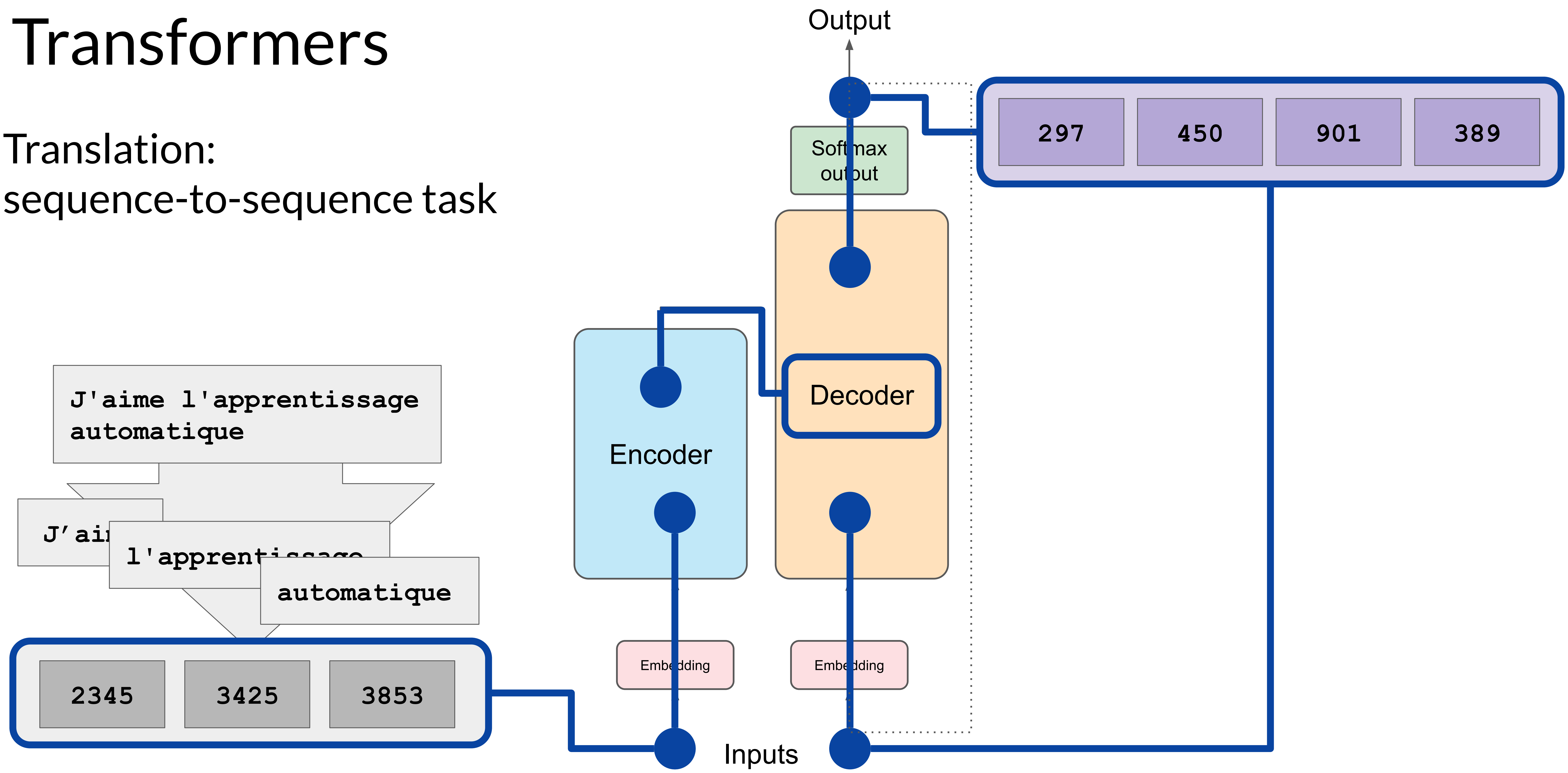
Transformers

Translation:
sequence-to-sequence task



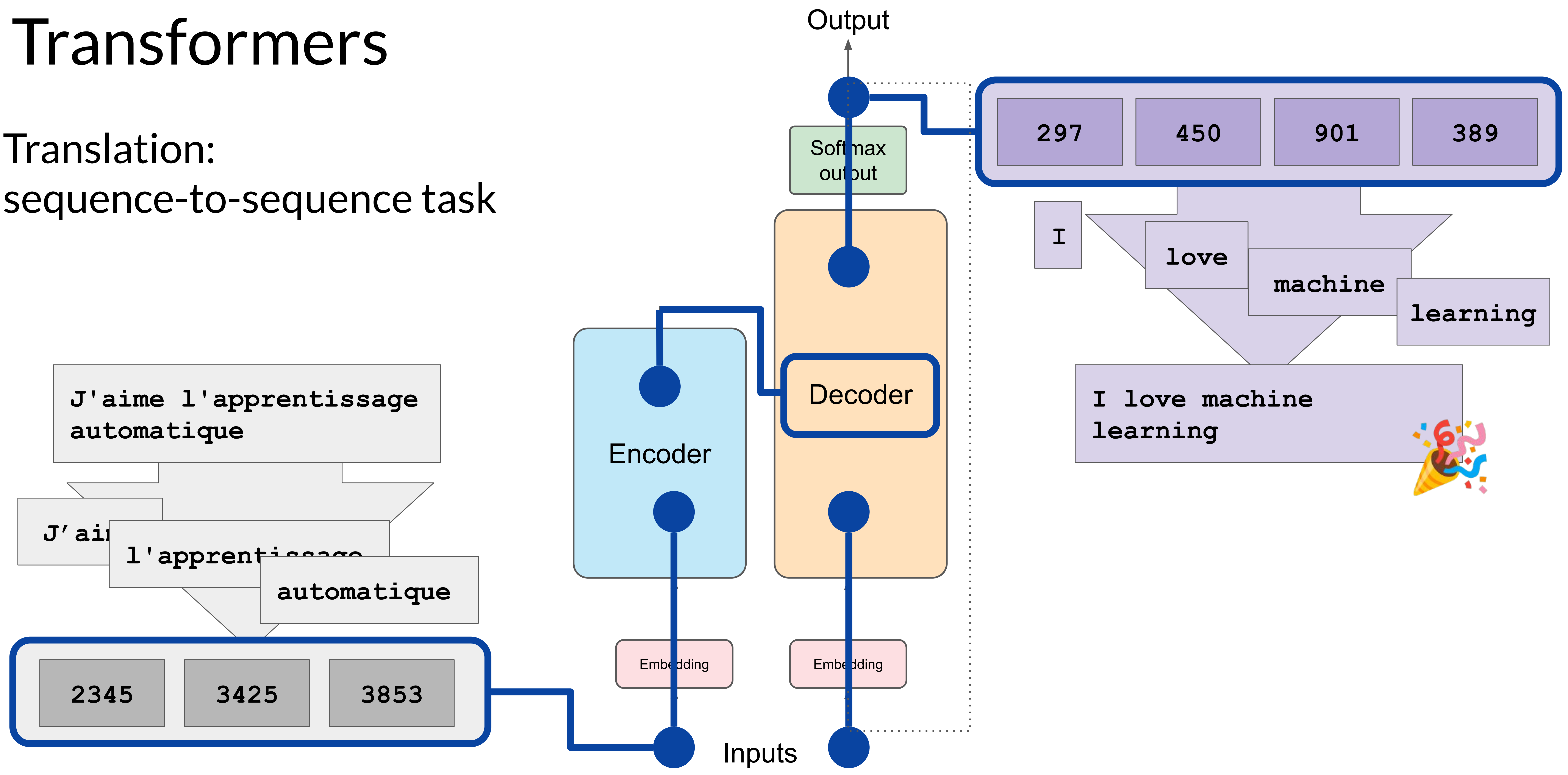
Transformers

Translation:
sequence-to-sequence task

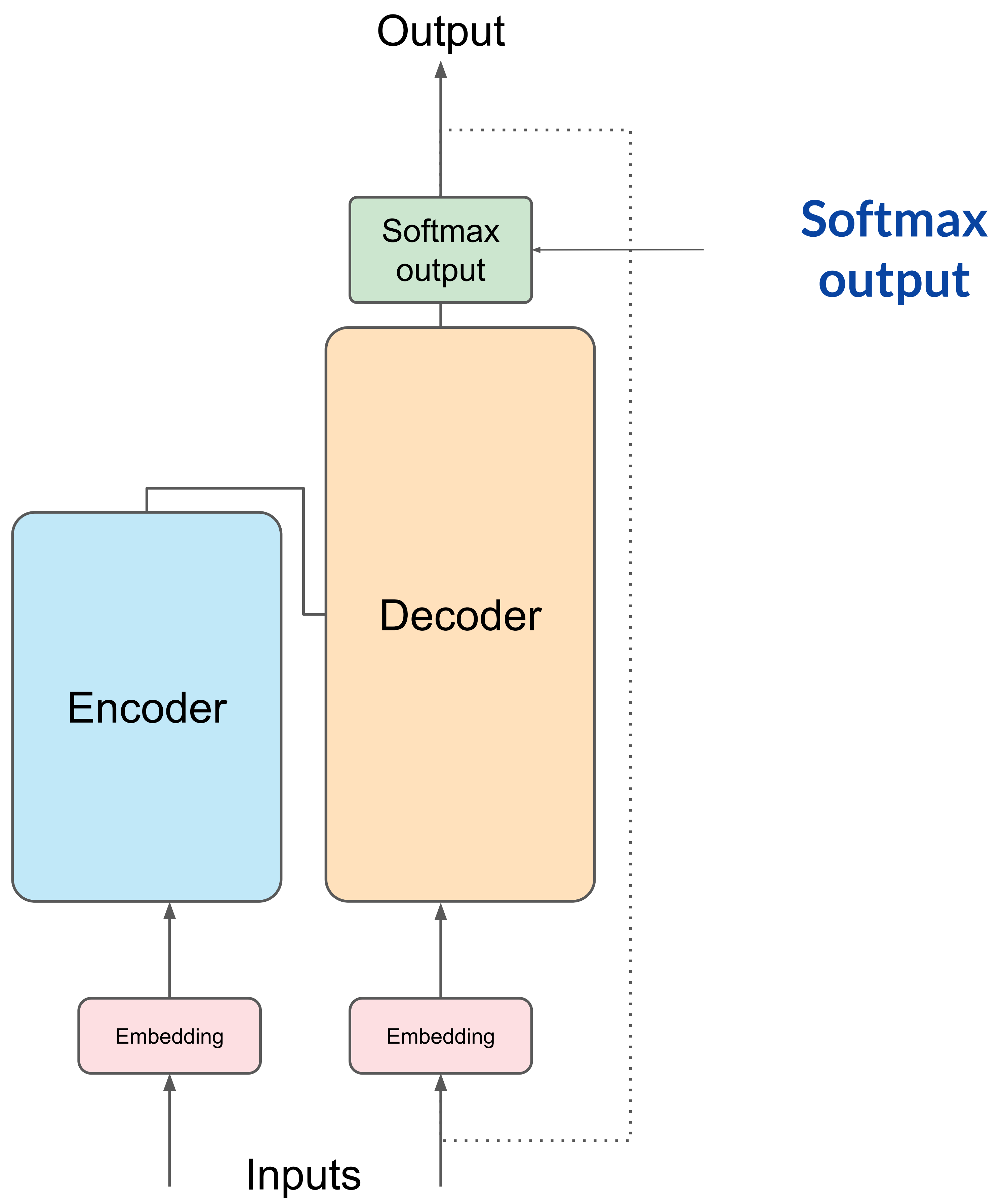


Transformers

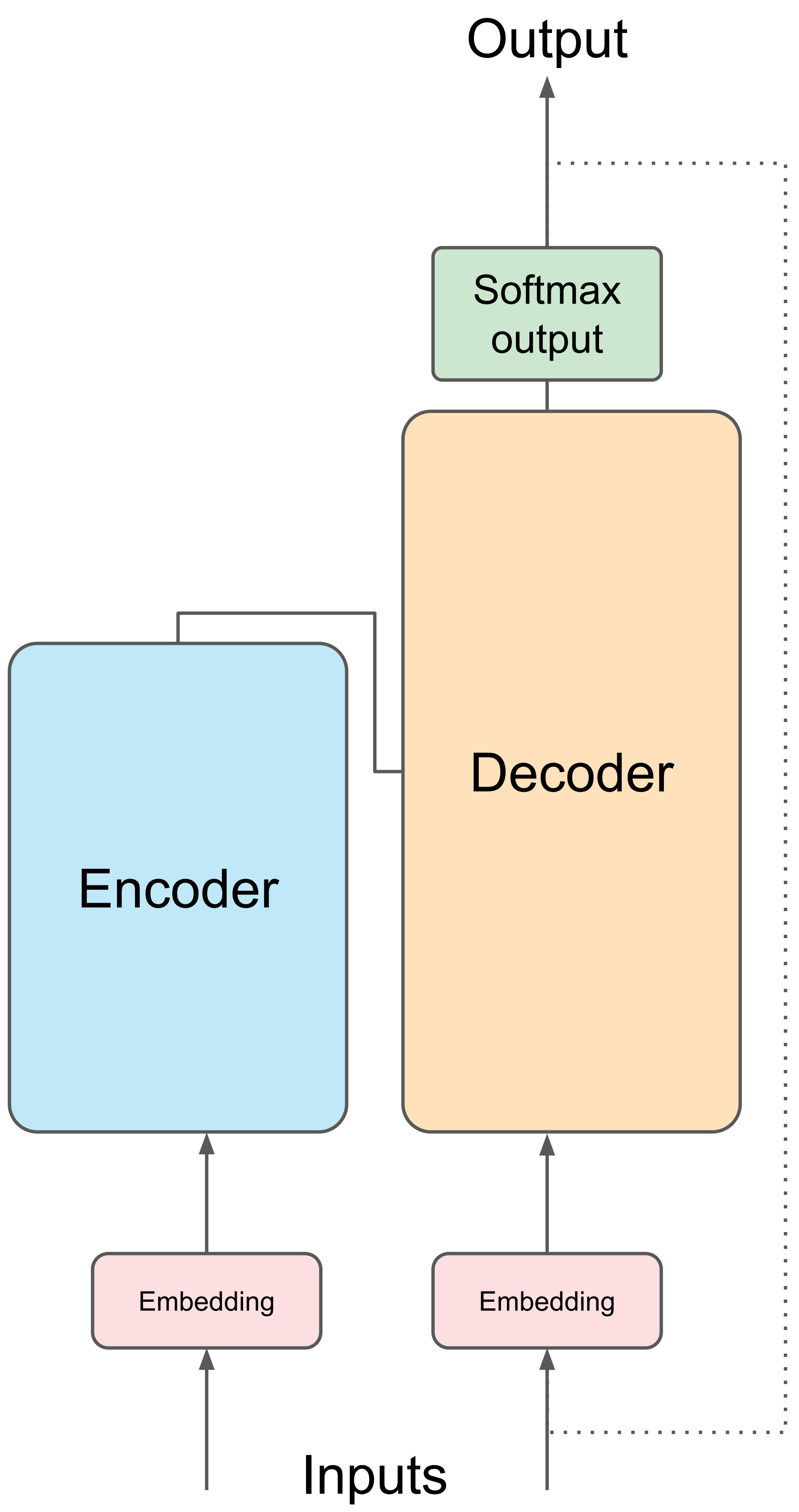
Translation:
sequence-to-sequence task



Transformers



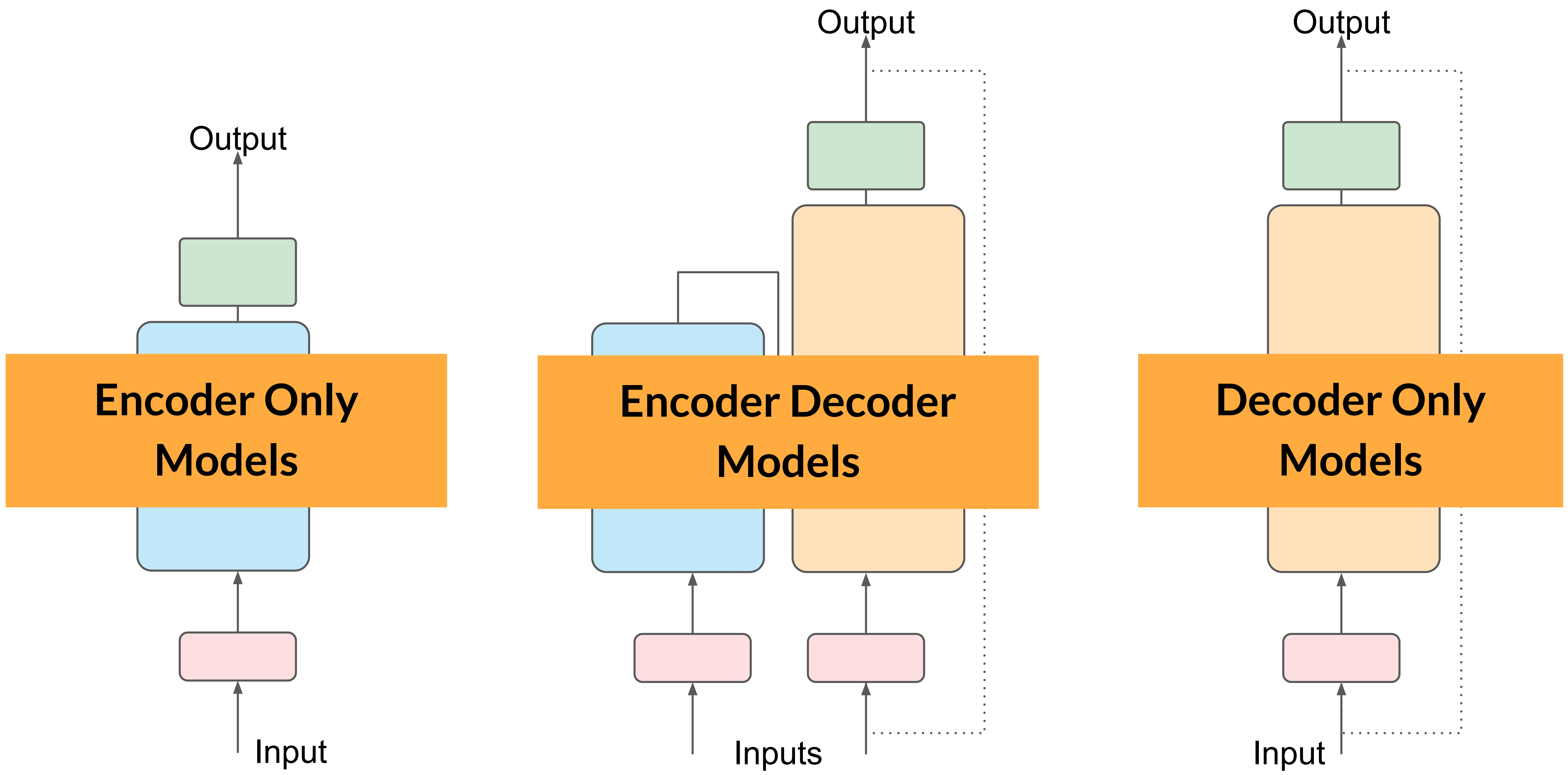
Transformers



Encoder
Encodes inputs ("prompts") with contextual understanding and produces one vector per input token.

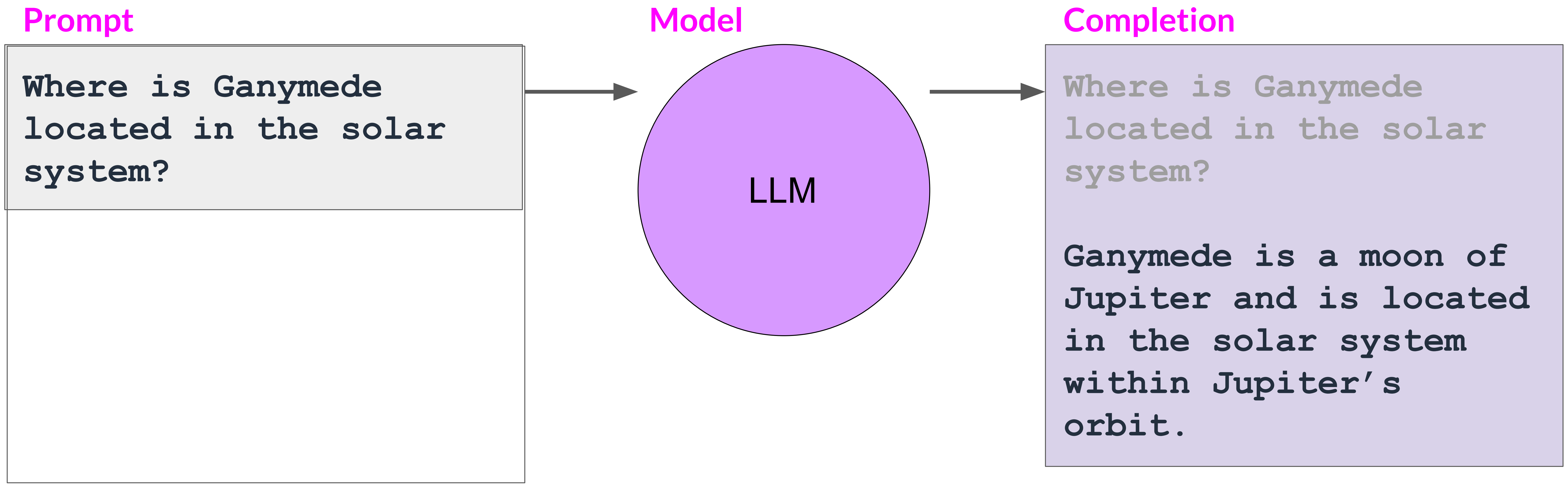
Decoder
Accepts input tokens and generates new tokens.

Transformers



Prompting and prompt engineering

Prompting and prompt engineering



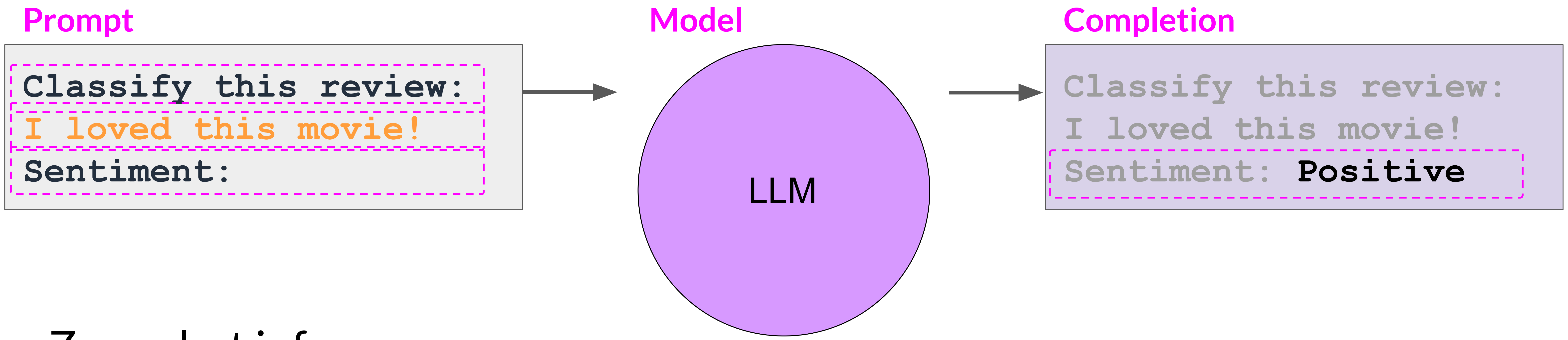
Where is Ganymede located in the solar system?

Where is Ganymede located in the solar system?

Ganymede is a moon of Jupiter and is located in the solar system within Jupiter's orbit.

Context window: typically a few thousand words

In-context learning (ICL) - zero shot inference

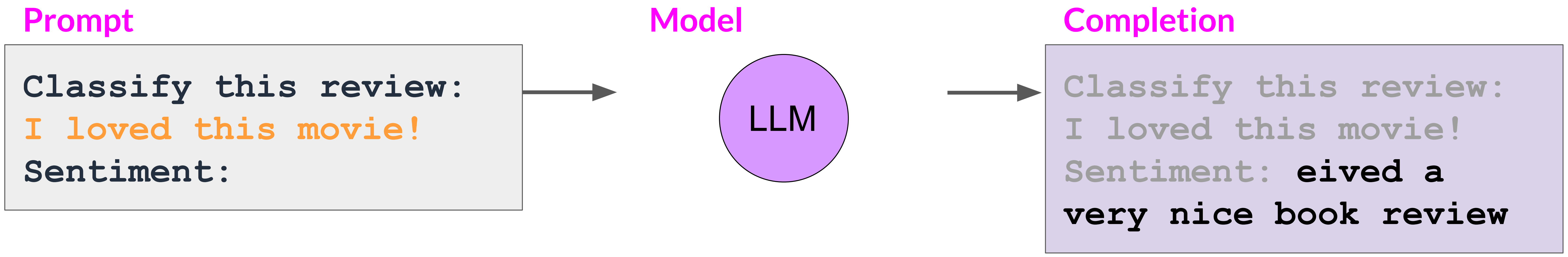


```
Classify this review:
I loved this movie!
Sentiment:
```

```
Classify this review:
I loved this movie!
Sentiment: Positive
```

Zero-shot inference

In-context learning (ICL) - zero shot inference



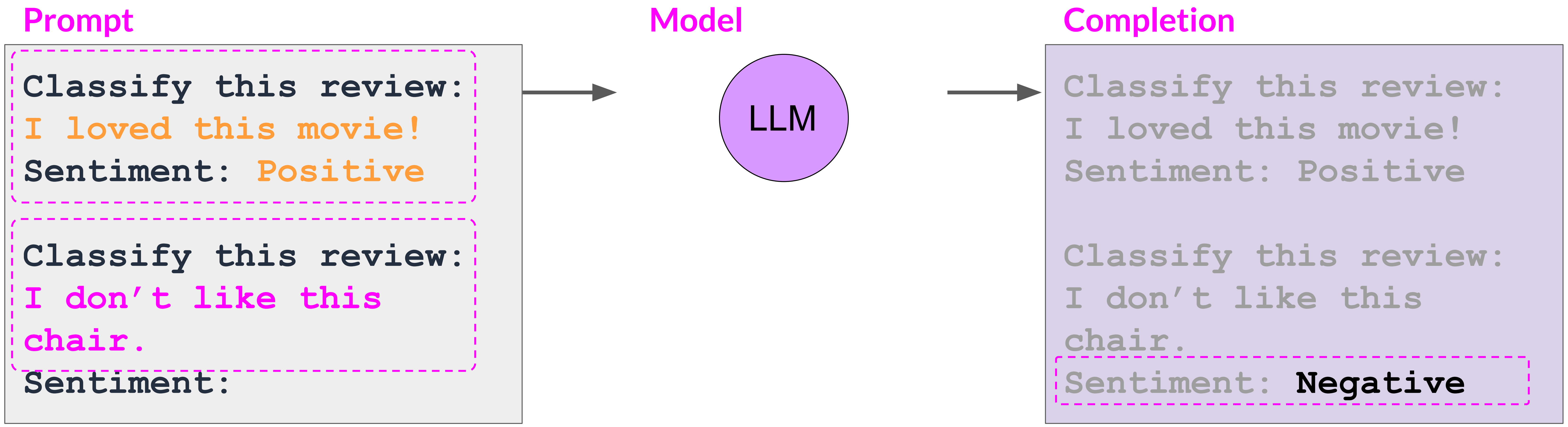
Prompt:

```
Classify this review:
I loved this movie!
Sentiment:
```

Completion:

```
Classify this review:
I loved this movie!
Sentiment: eived a
very nice book review
```

In-context learning (ICL) - one shot inference

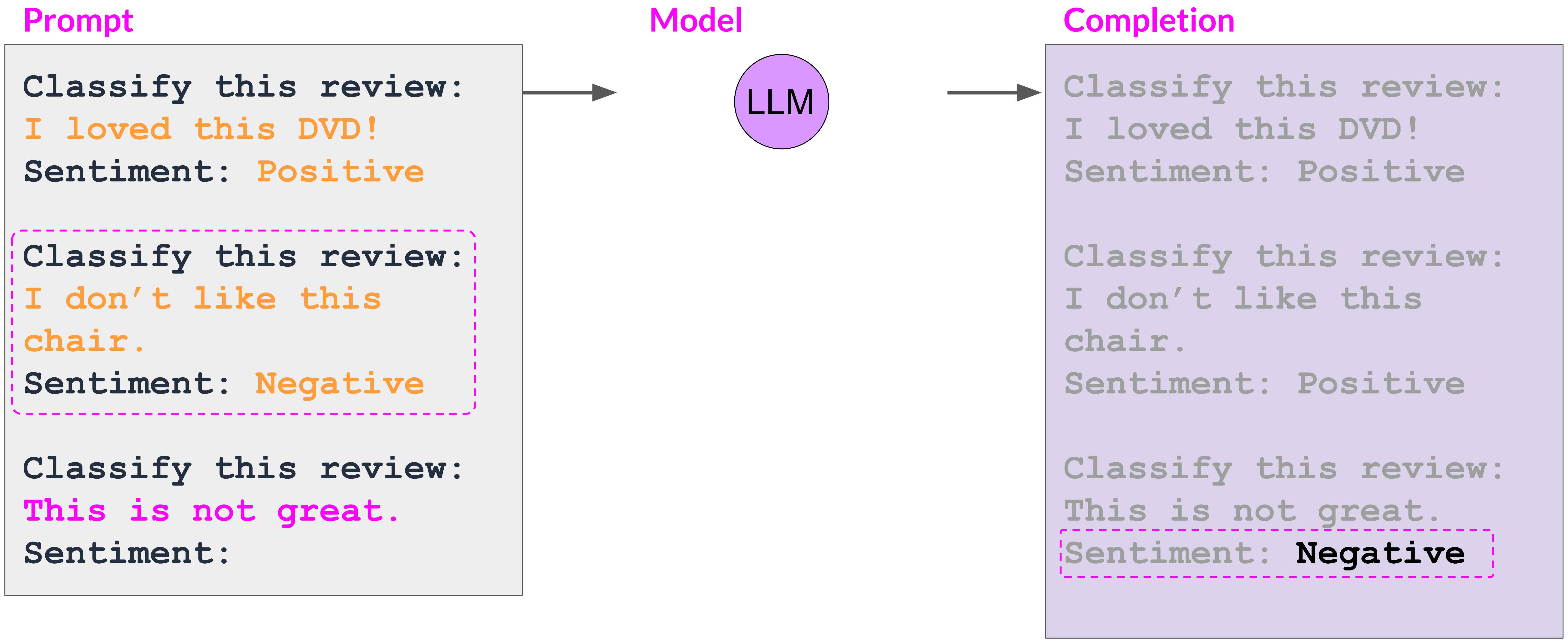


```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment:
```

```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment: Negative
```

One-shot inference

In-context learning (ICL) - few shot inference



Prompt:

```
Classify this review:
I loved this DVD!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment: Negative

Classify this review:
This is not great.
Sentiment:
```

Completion:

```
Classify this review:
I loved this DVD!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment: Positive

Classify this review:
This is not great.
Sentiment: Negative
```

Summary of in-context learning (ICL)

Prompt // Zero Shot

```
Classify this review:
I loved this movie!
Sentiment:
```

Context Window
(few thousand words)

Prompt // One Shot

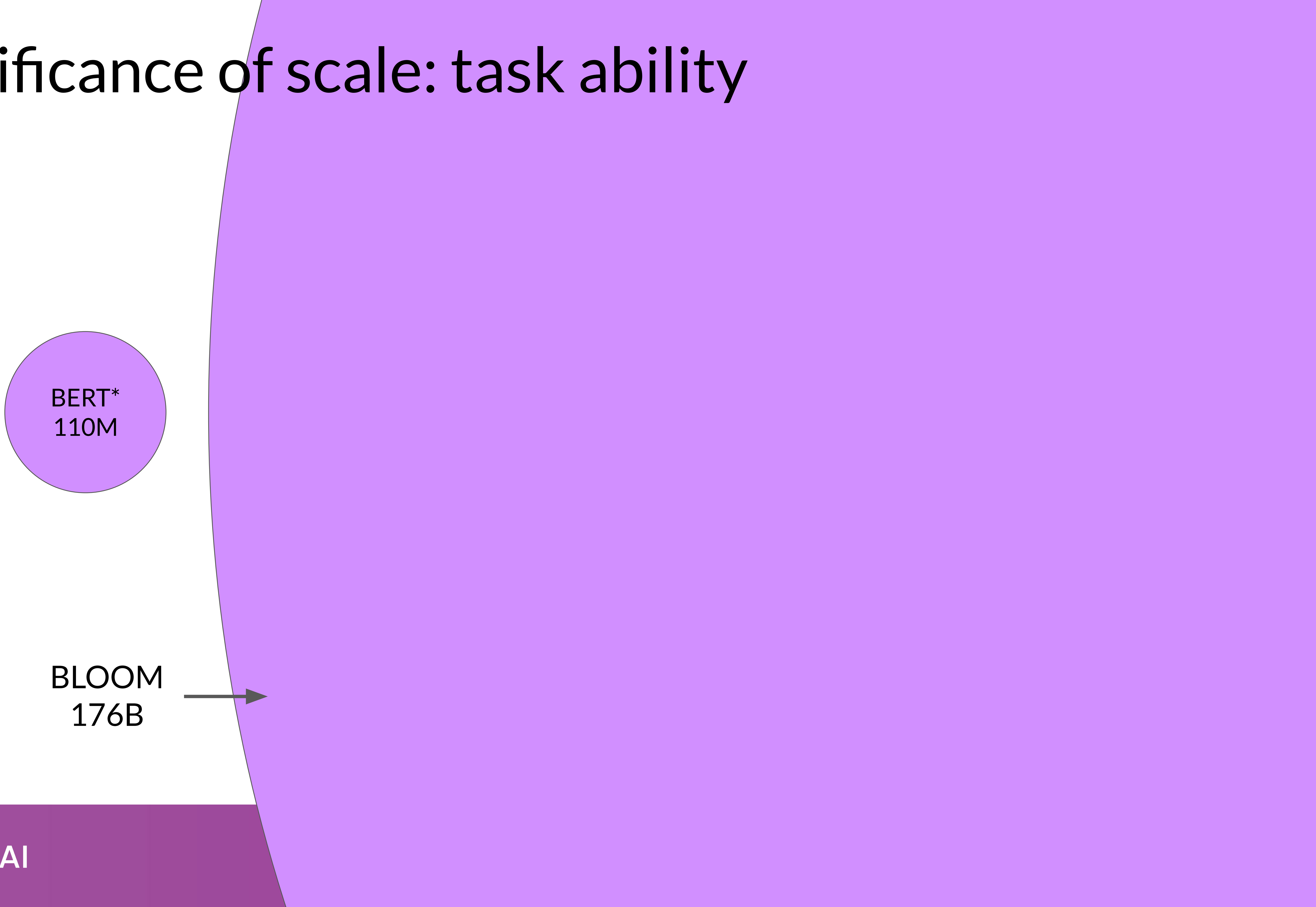
```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment:
```

Prompt // Few Shot

>5 or 6 examples

```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment: Negative

Classify this review:
Who would use this
product?
Sentiment:
```

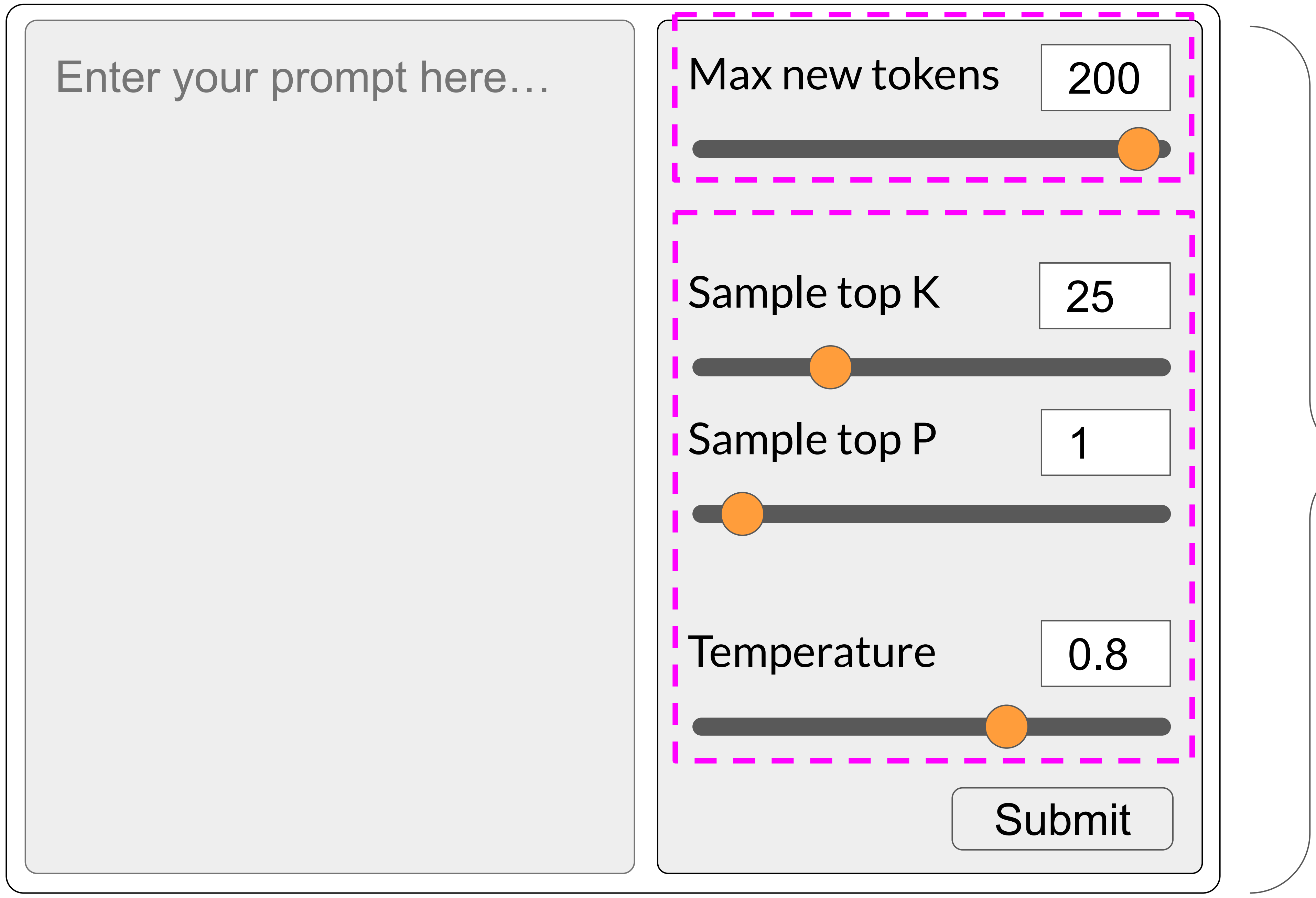
The significance of scale: task ability



*Bert-base

Generative configuration parameters for inference

Generative configuration - inference parameters



Enter your prompt here…

Max new tokens 200

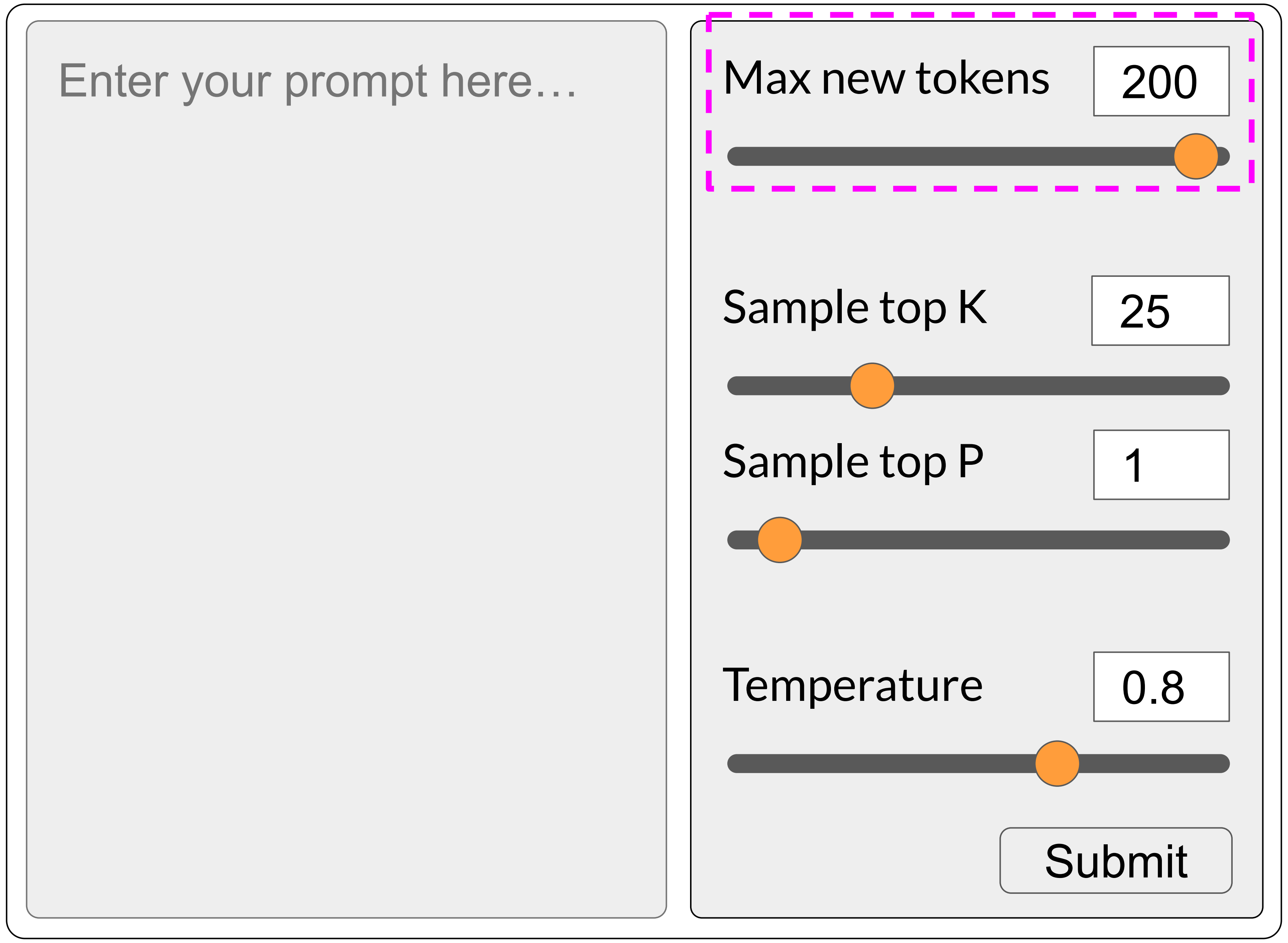
Sample top K 25

Sample top P 1

Temperature 0.8

Inference configuration parameters

Generative configuration - max new tokens



Enter your prompt here…

Max new tokens 200

Sample top K 25

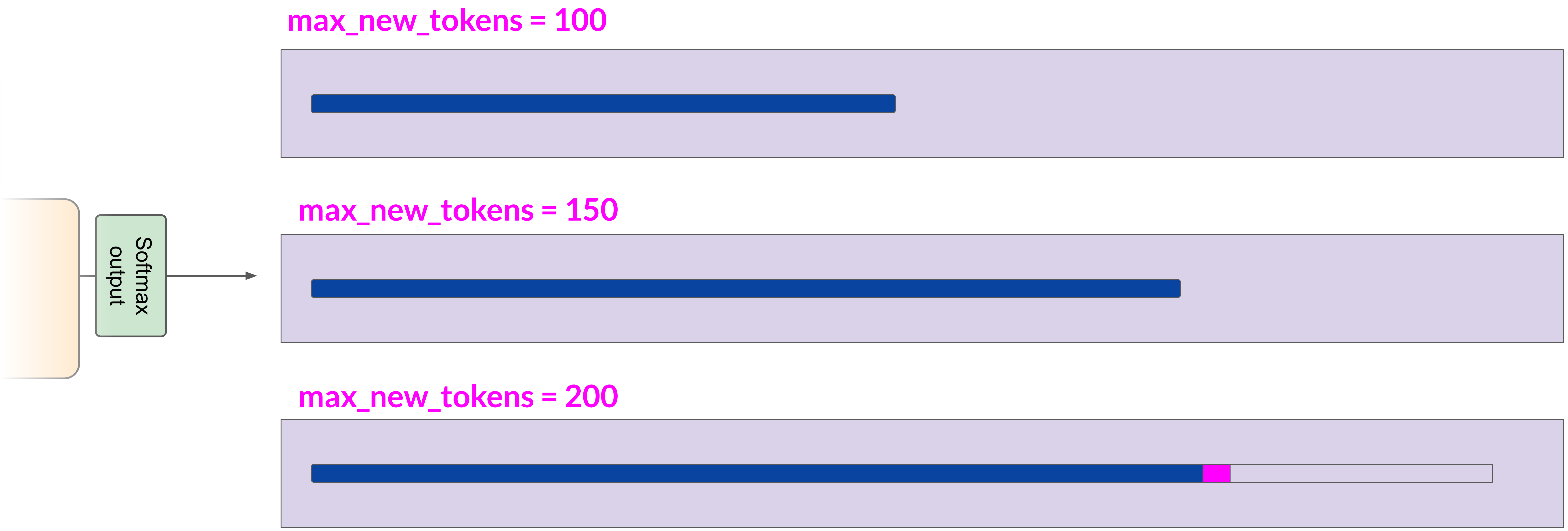
Sample top P 1

Temperature 0.8

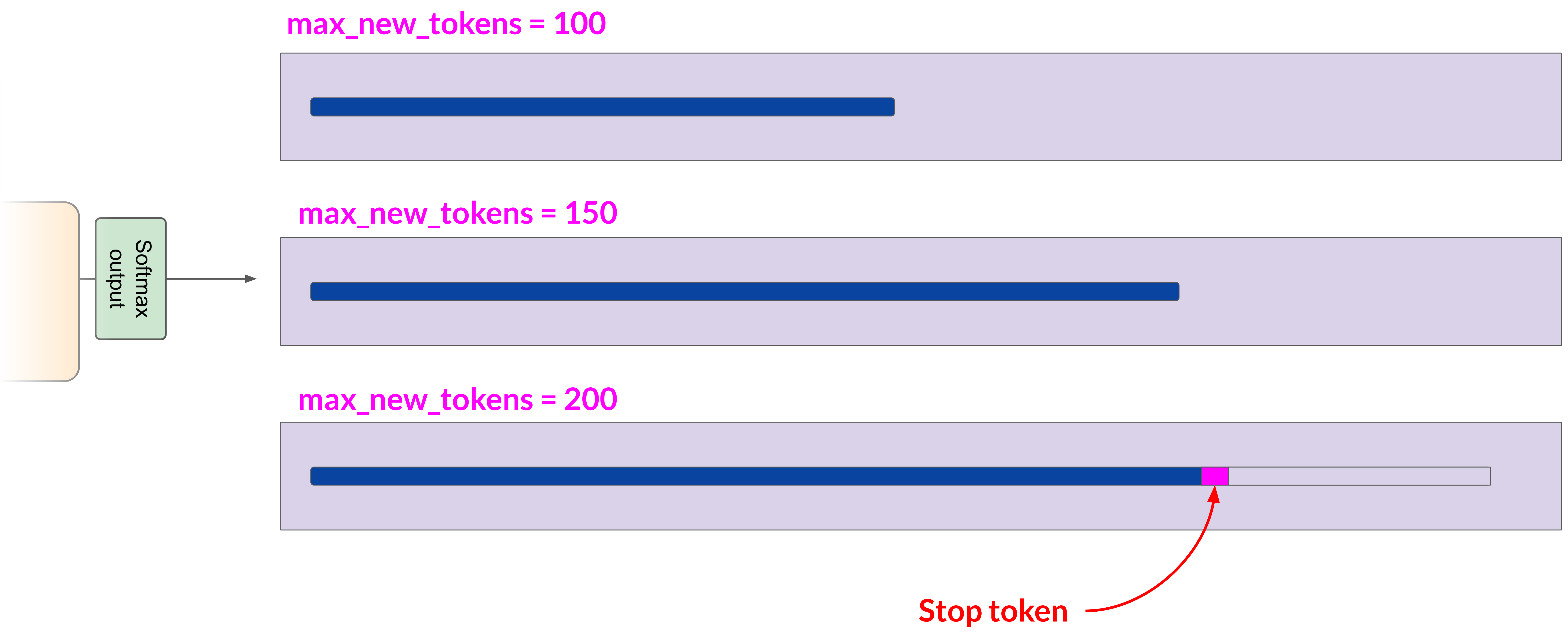
Max new tokens

規定最長的生成Token單字量

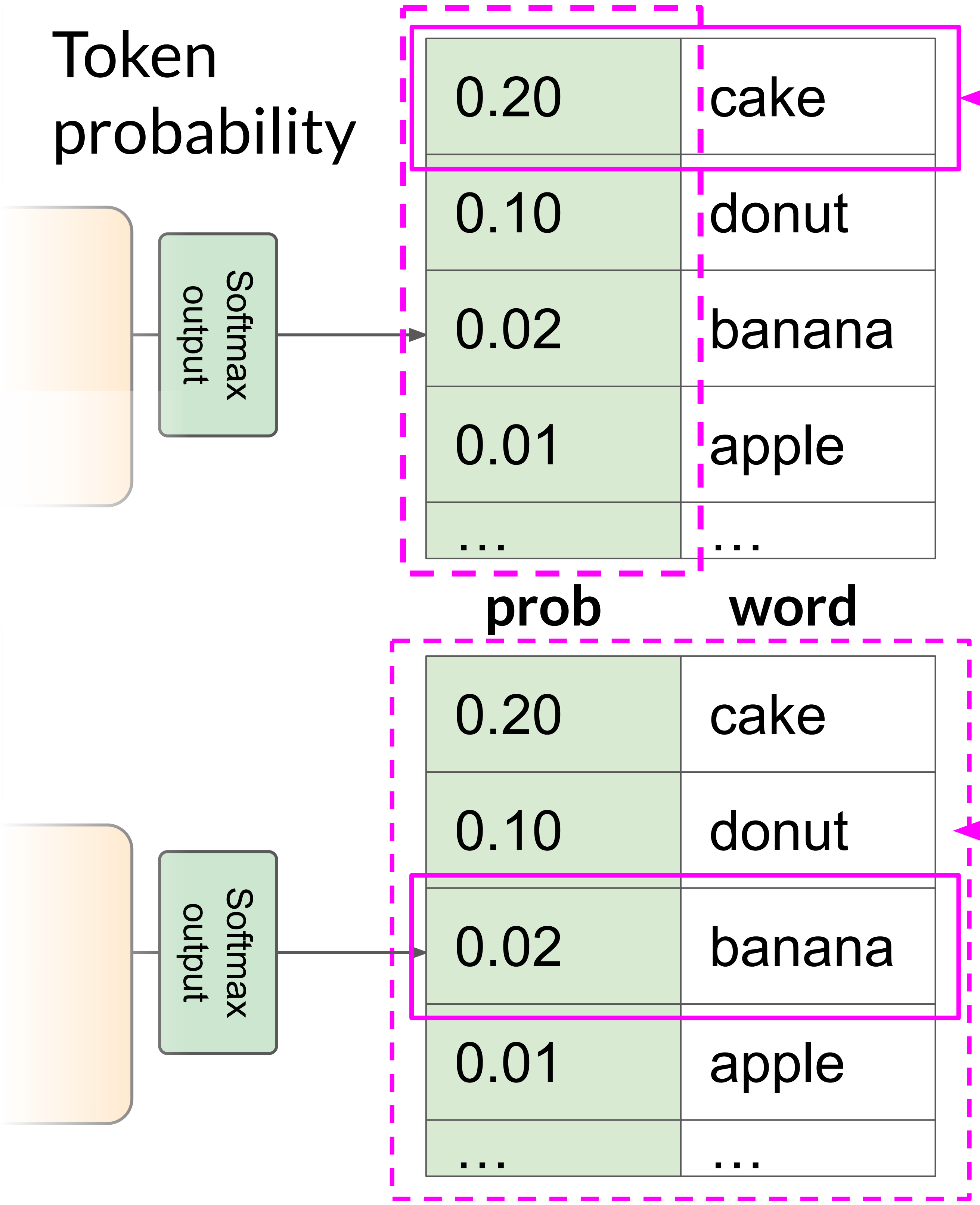
Generative config - max new tokens



Generative config - max new tokens



Generative config - greedy vs. random sampling



prob	word
0.20	cake
0.10	donut
0.02	banana
0.01	apple
…	…

prob	word
0.20	cake
0.10	donut
0.02	banana
0.01	apple
…	…

greedy: The word/token with the highest probability is selected.

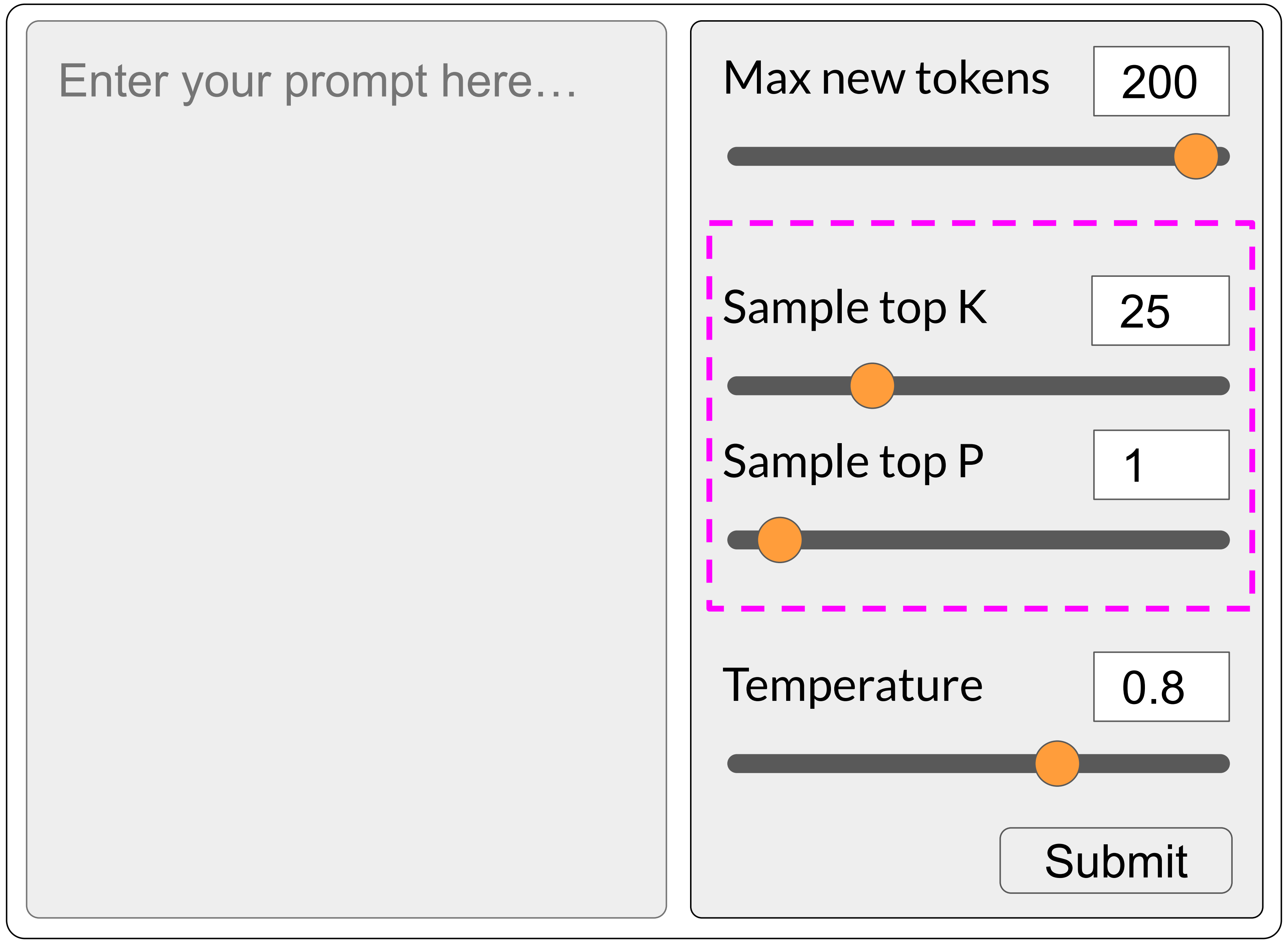
每次都找機率最大的token去做機率預測
- 短文本會有好的效果
-長文本會受到重複單詞的影響

把預測機率值當做被抽到的機率
- 會比較有創意
(do sample=True)

random(-weighted) sampling: select a token using a random-weighted strategy across the probabilities of all tokens.

Here, there is a 20% chance that 'cake' will be selected, but 'banana' was actually selected.

Generative configuration - top-k and top-p



Enter your prompt here…

Max new tokens 200

Sample top K 25

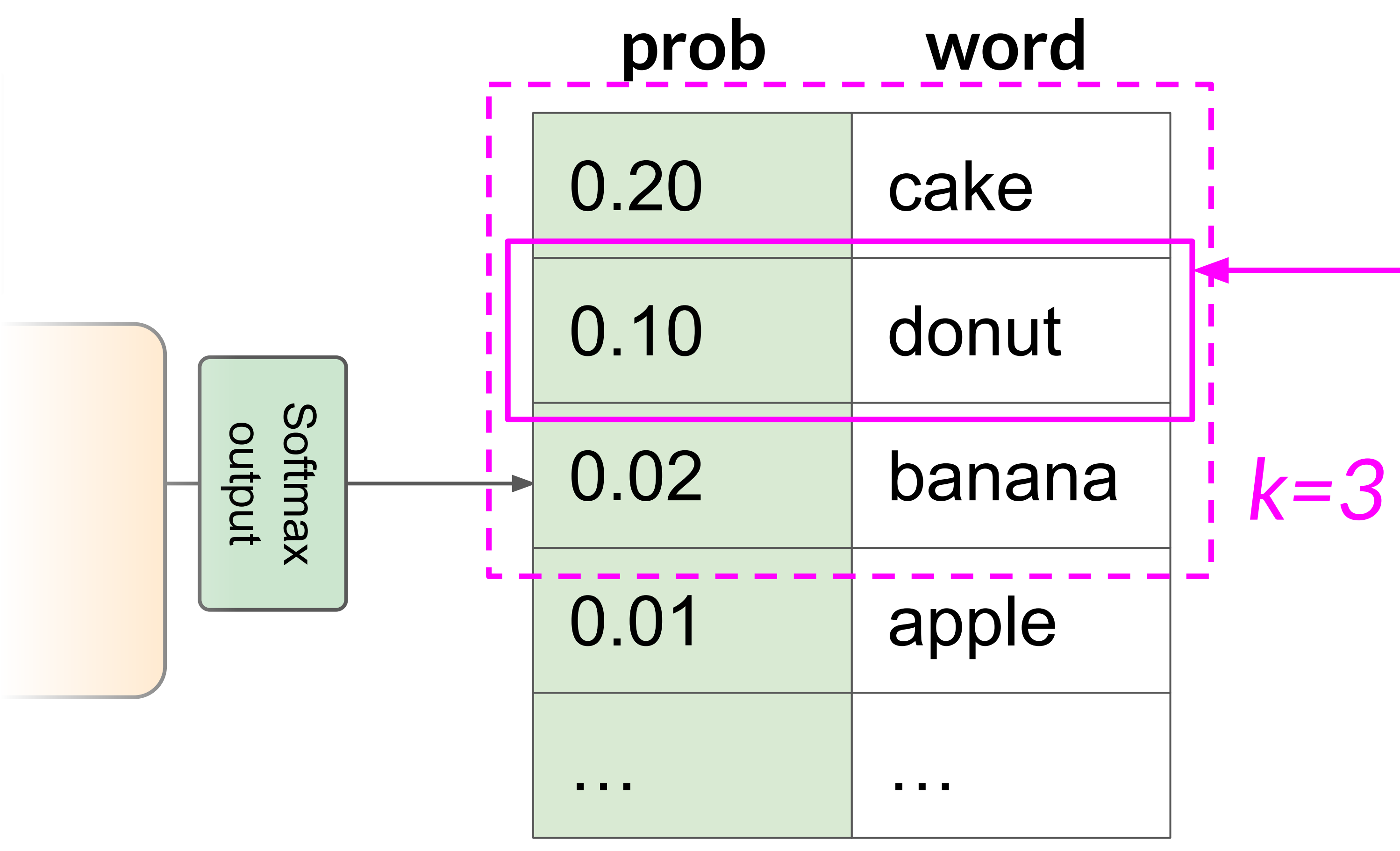
Sample top P 1

Temperature 0.8

Top-k and top-p sampling

只從幾個機率最大值做抽樣

Generative config - top-k sampling

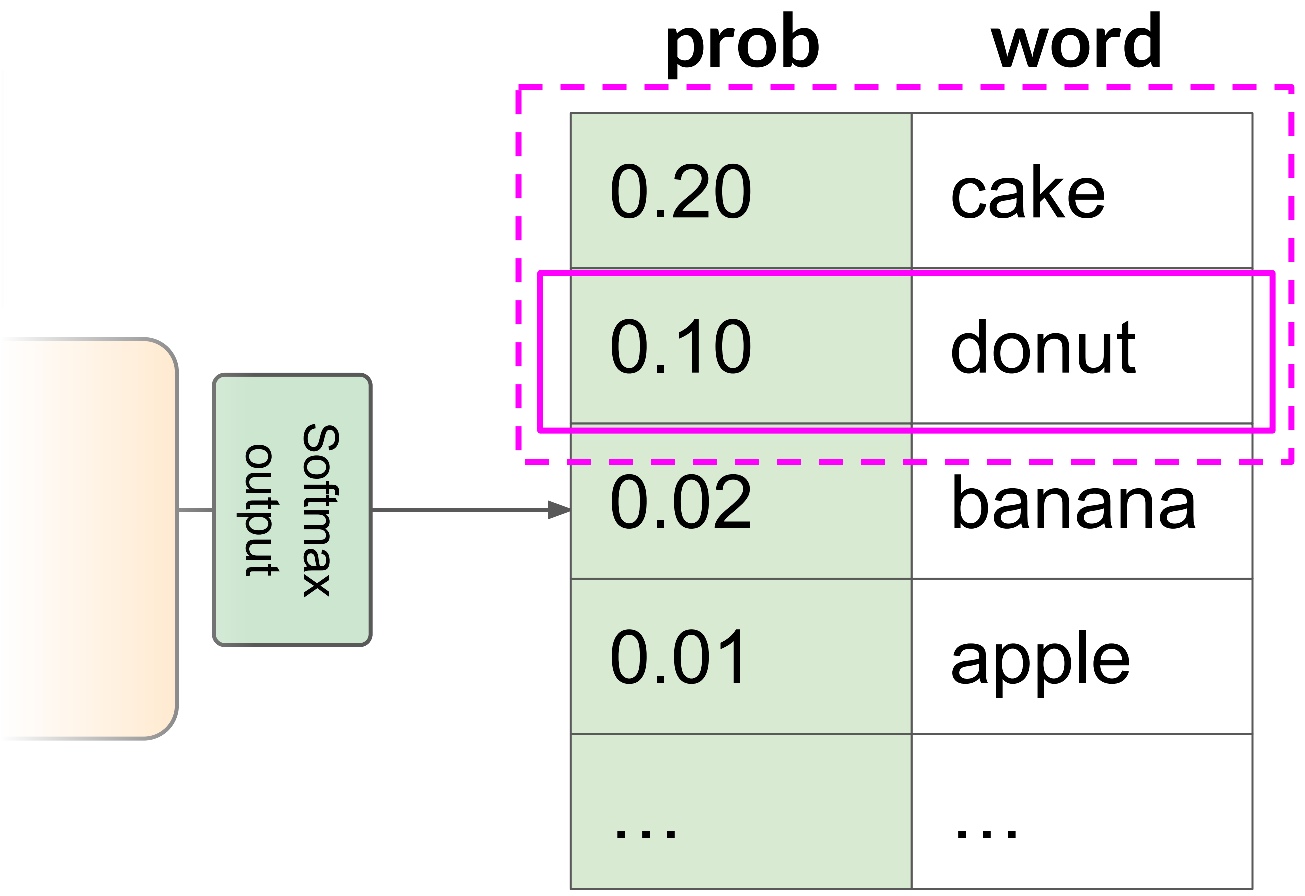


prob	word
0.20	cake
0.10	donut
0.02	banana
0.01	apple
…	…

top-k: select an output from the top-k results after applying random-weighted strategy using the probabilities

只從幾個機率最大值做抽樣

Generative config - top-p sampling



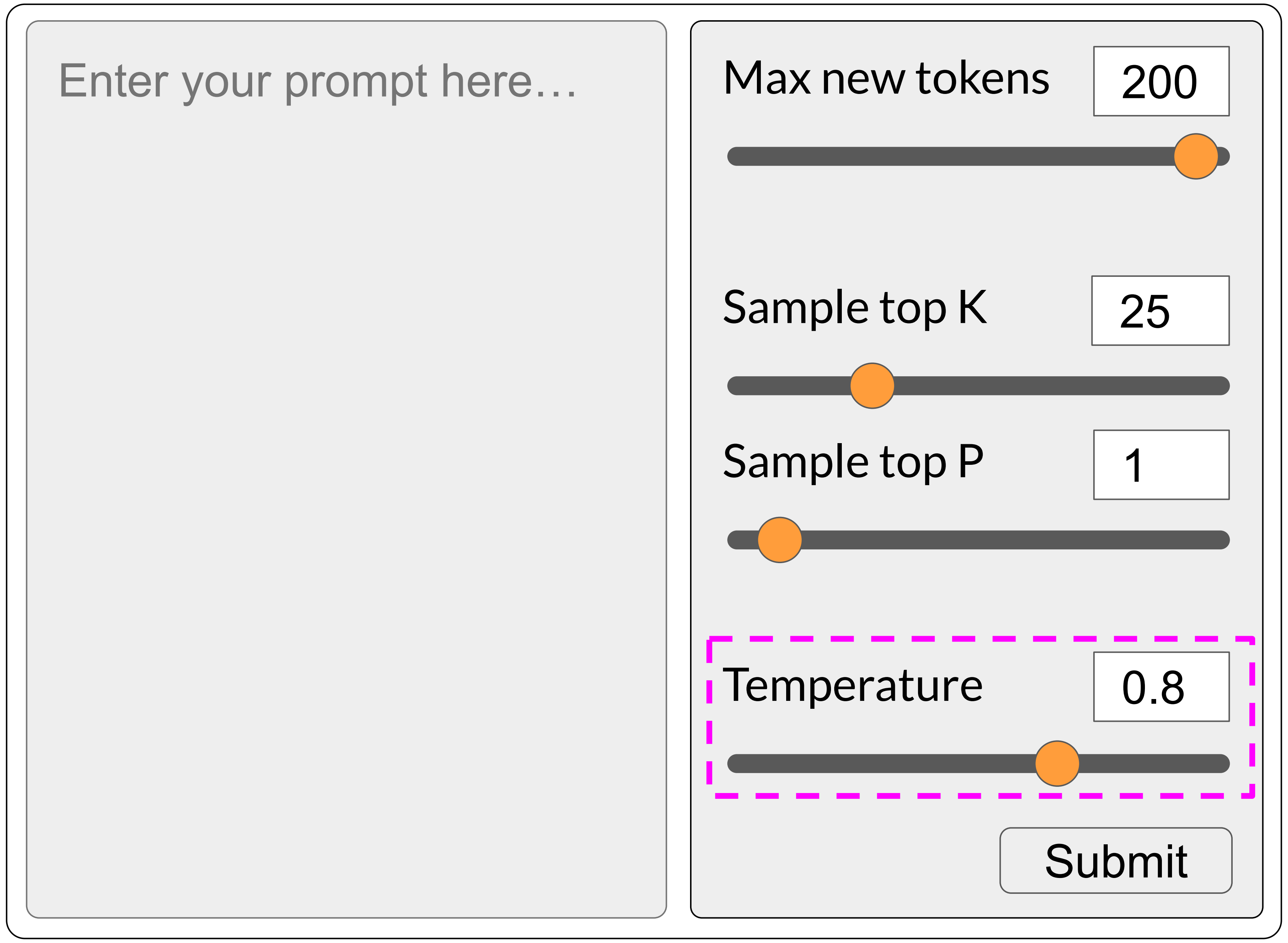
prob	word
0.20	cake
0.10	donut
0.02	banana
0.01	apple
...	...

top-p: select an output using the random-weighted strategy with the top-ranked consecutive results by probability and with a cumulative probability <= *p*.

p = 0.30

任意組合單詞的機率sum p=0.3,
再從這裡面做字詞選擇

Generative configuration - temperature



Enter your prompt here…

Max new tokens 200

Sample top K 25

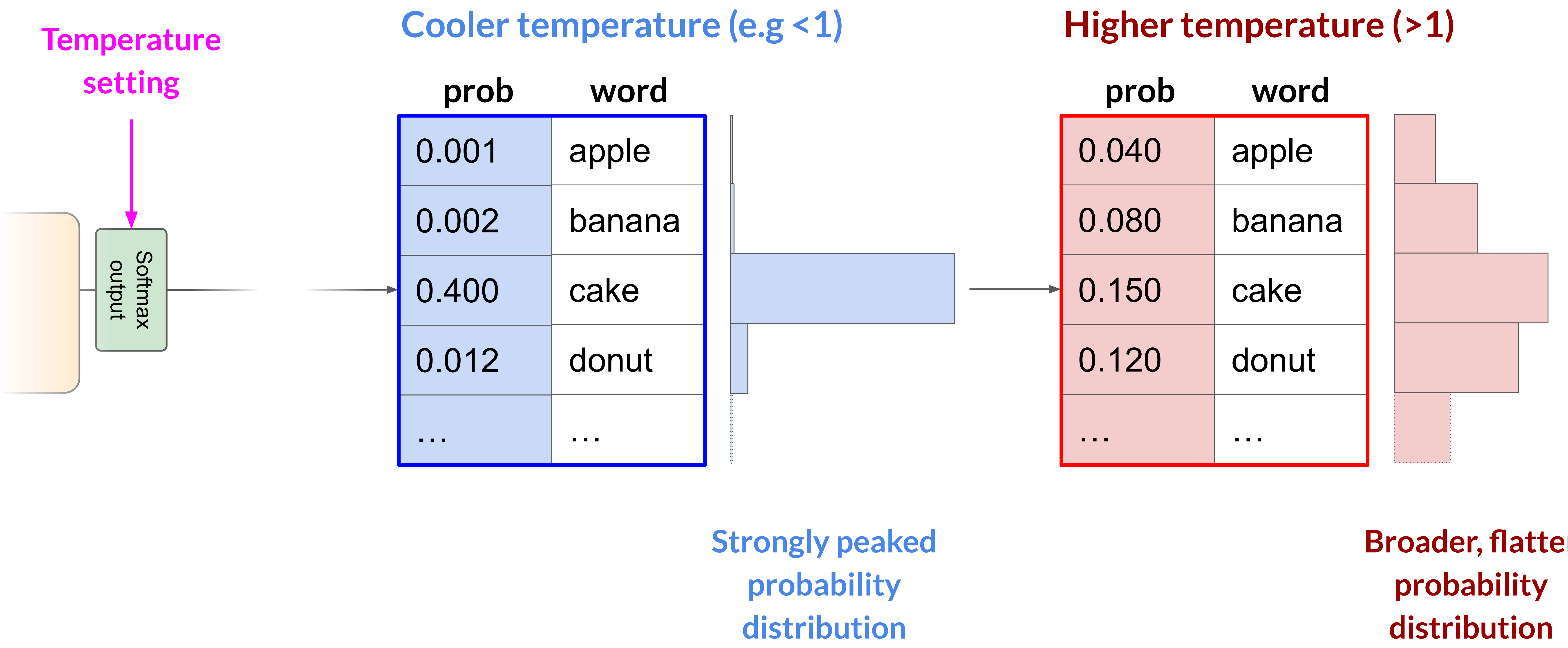
Sample top P 1

Temperature 0.8

softmax的機率值分佈極端性
- 越小(<1)越集中於高機率
- 越大(>1)越分散於各機率

Temperature

Generative config - temperature



Temperature setting
Softmax output
Cooler temperature (e.g <1)
Higher temperature (>1)
Strongly peaked probability distribution
Broader, flatter probability distribution

Cooler temperature (e.g <1)

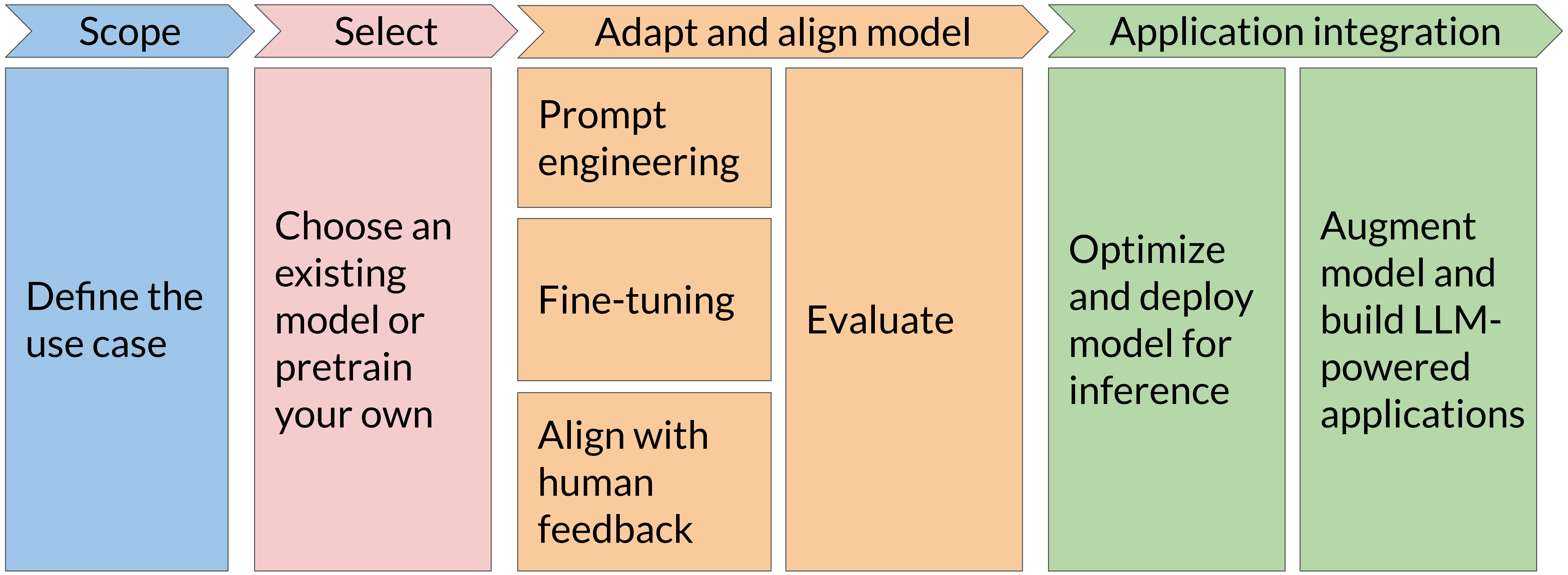
prob	word
0.001	apple
0.002	banana
0.400	cake
0.012	donut
…	…

Higher temperature (>1)

prob	word
0.040	apple
0.080	banana
0.150	cake
0.120	donut
…	…

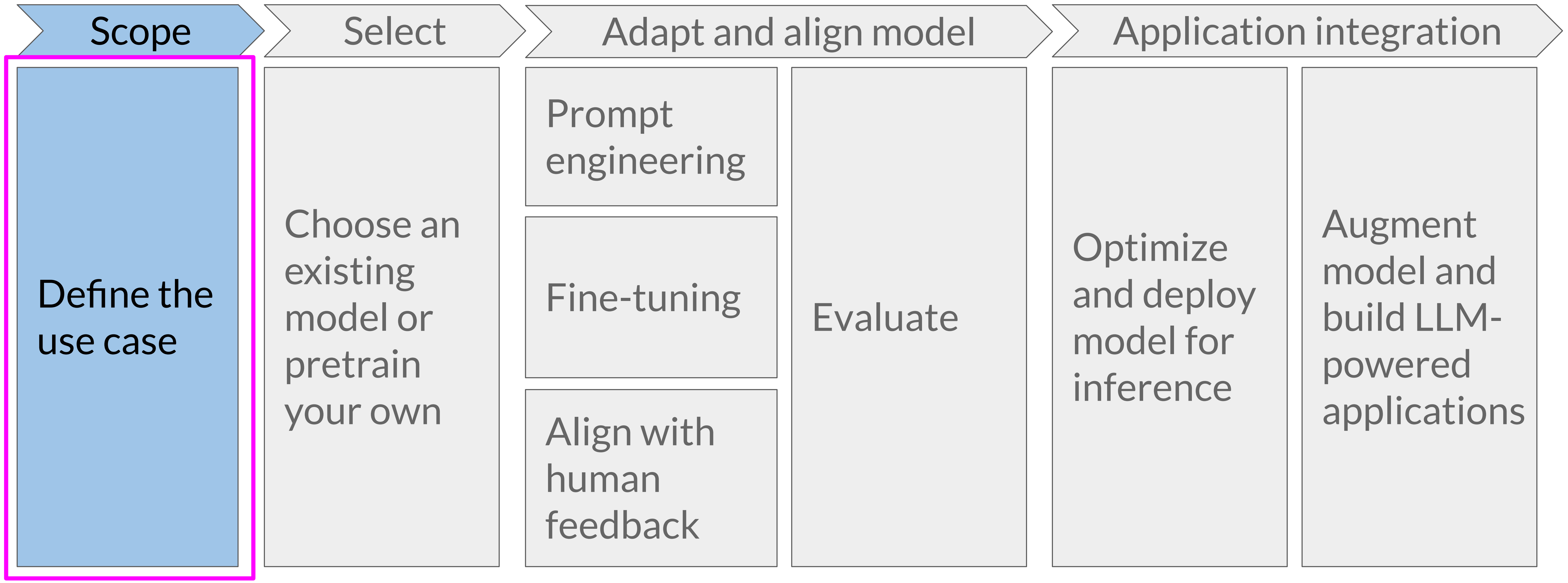
Generative AI project lifecycle

Generative AI project lifecycle



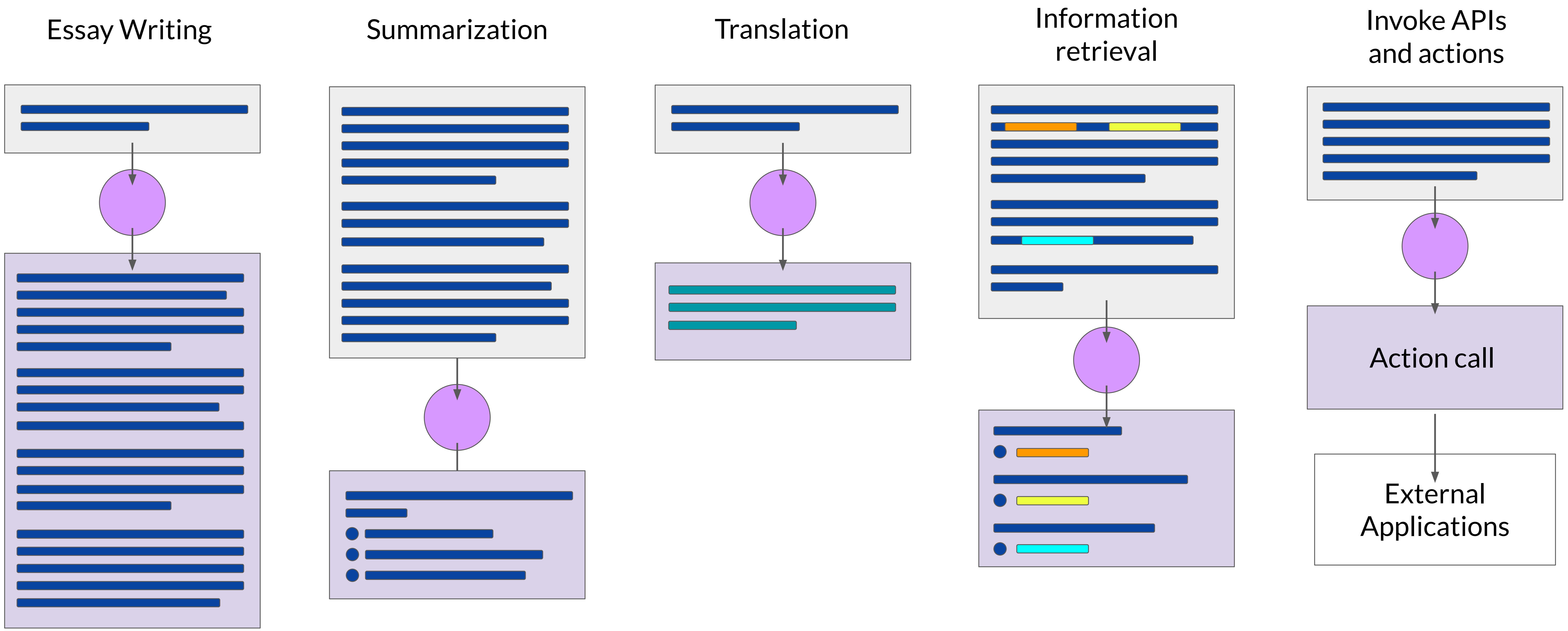
Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

Generative AI project lifecycle

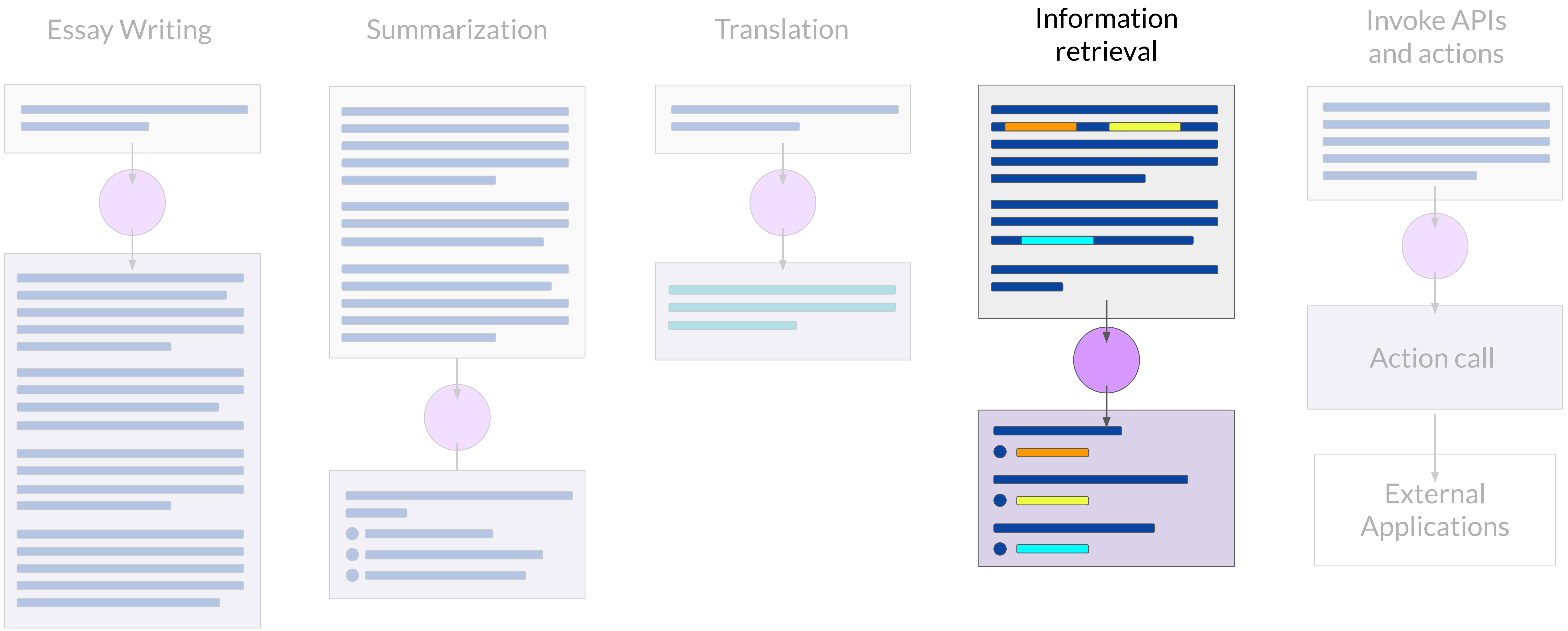


Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

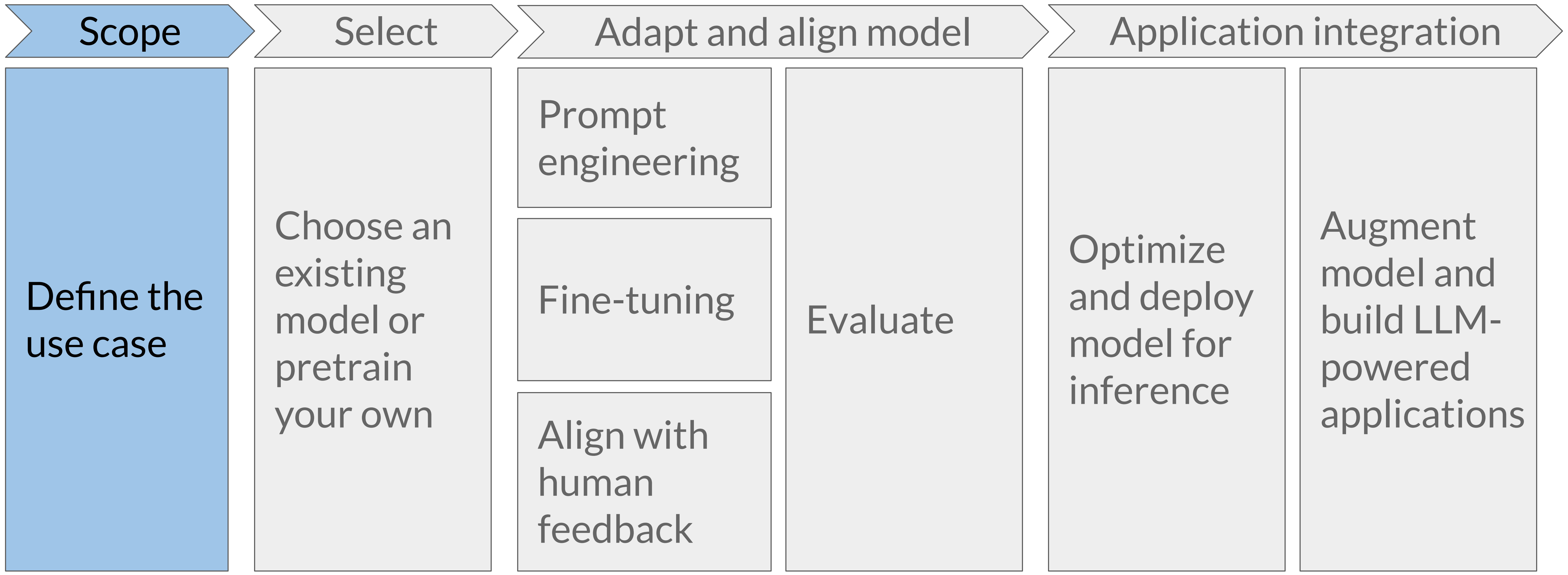
Good at many tasks?



Or good at a single task?

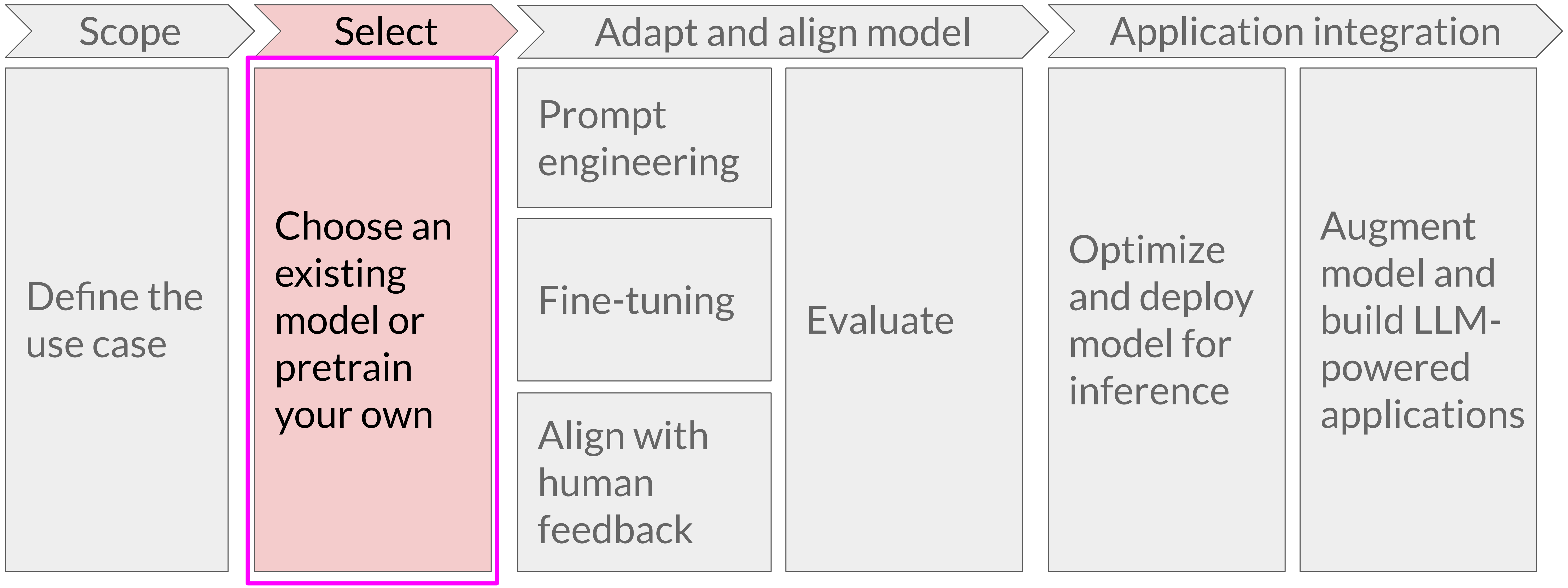


Generative AI project lifecycle

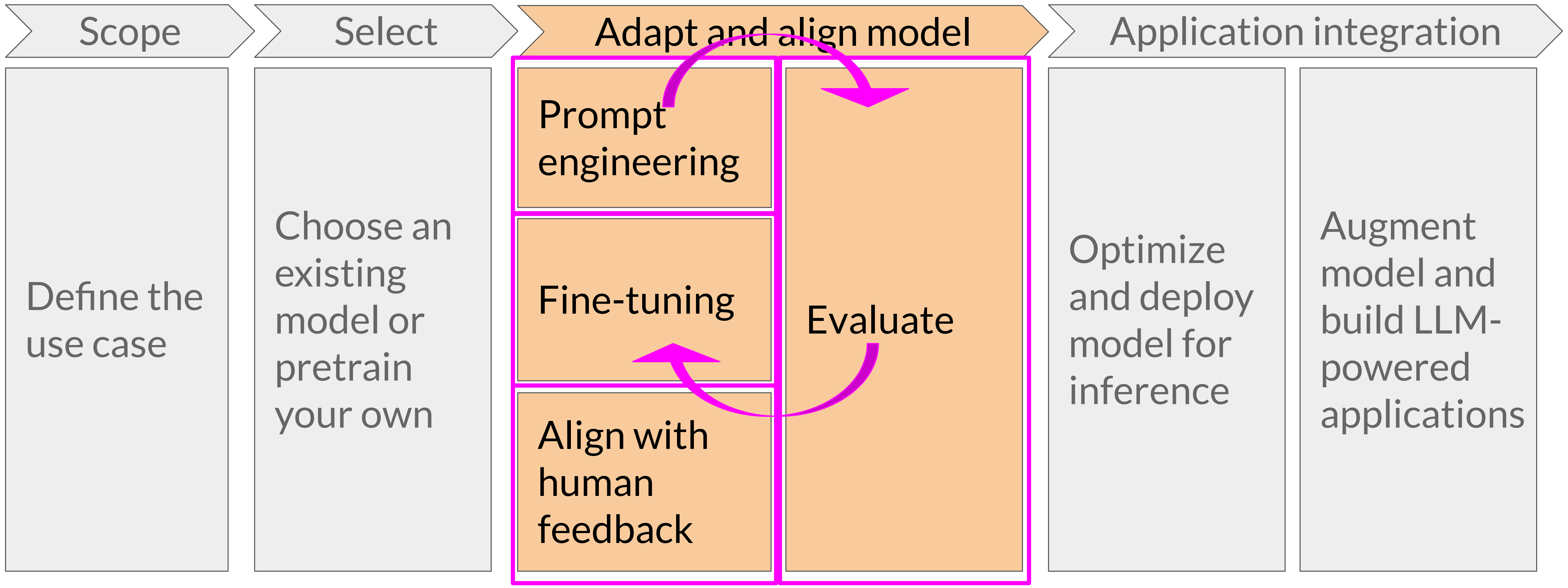


Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

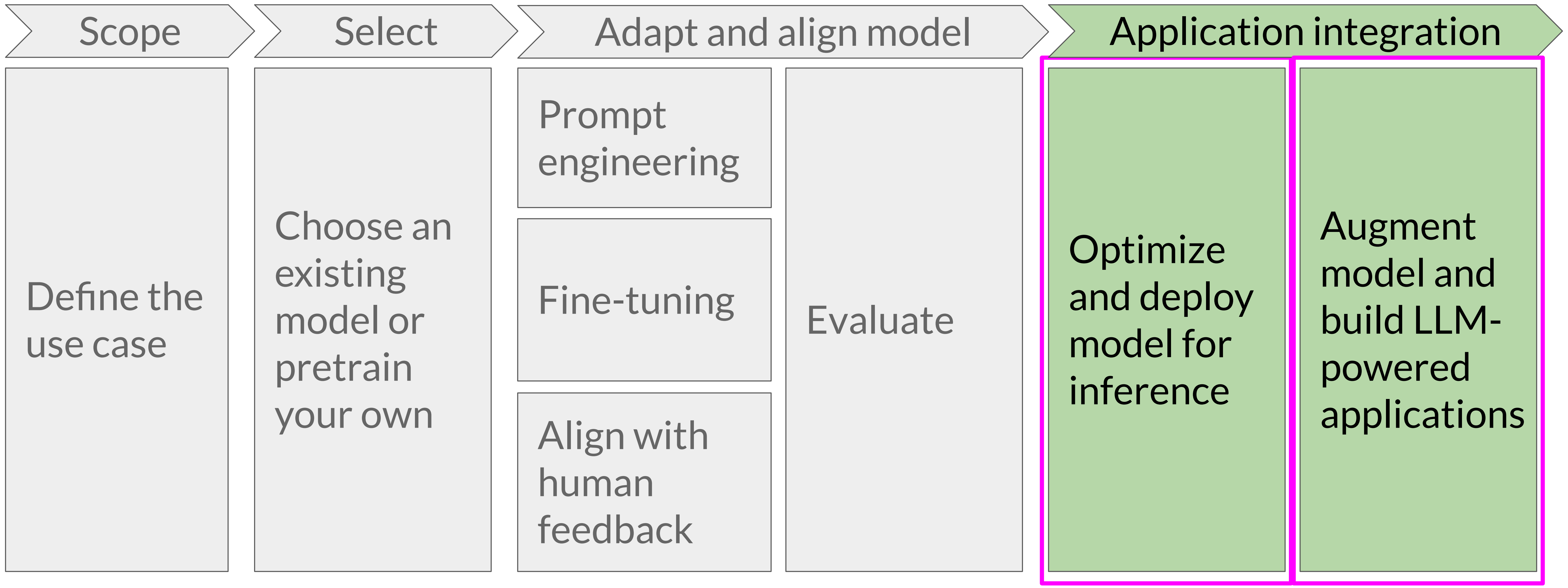
Generative AI project lifecycle


Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

Generative AI project lifecycle

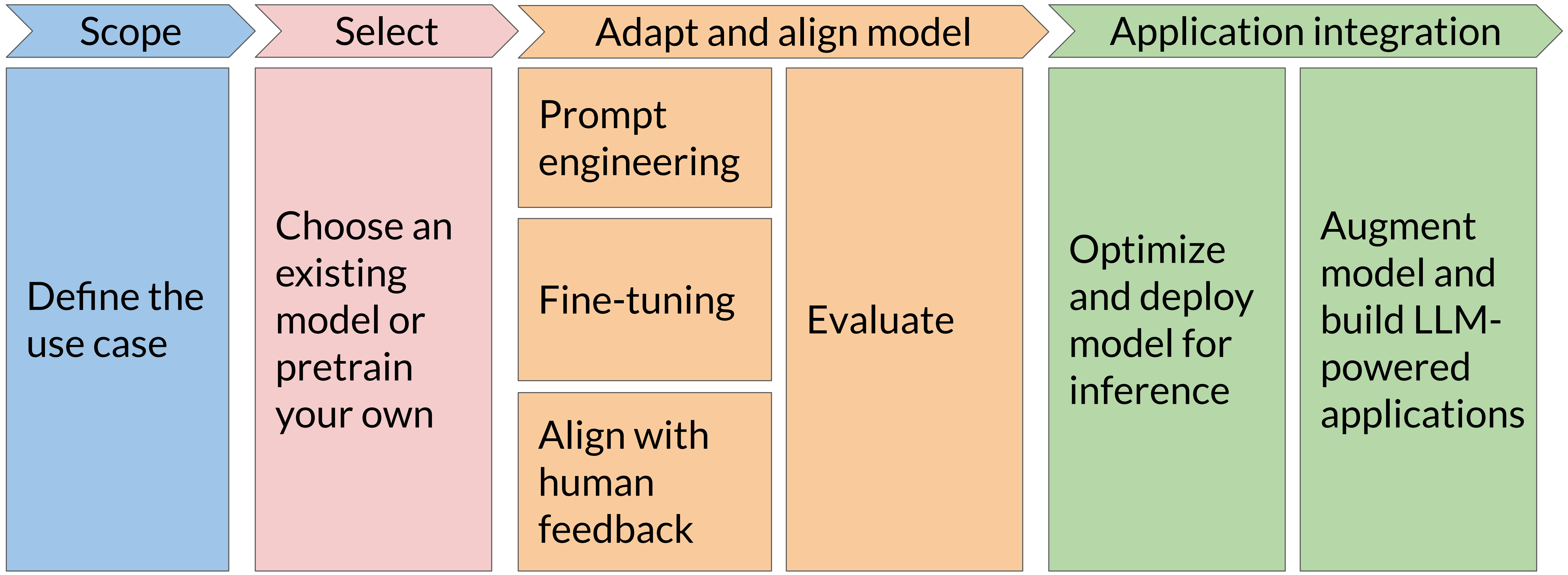

Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

Generative AI project lifecycle

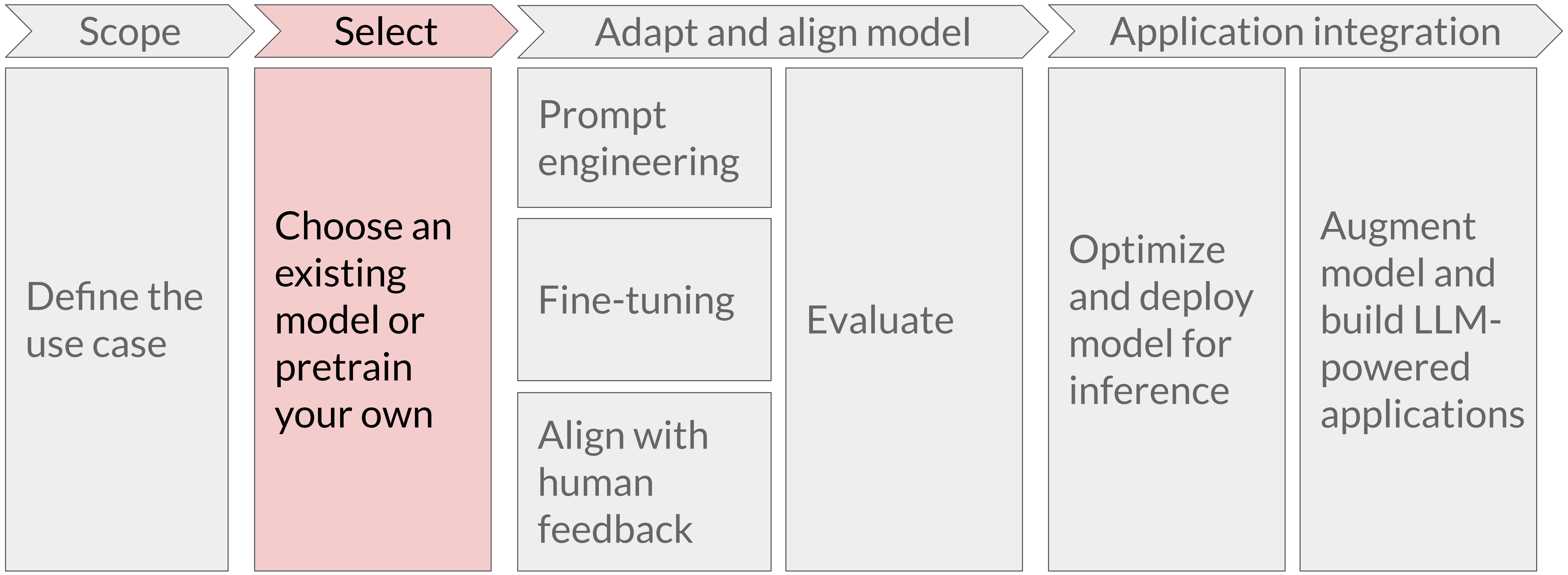

Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

Pre-training and scaling laws

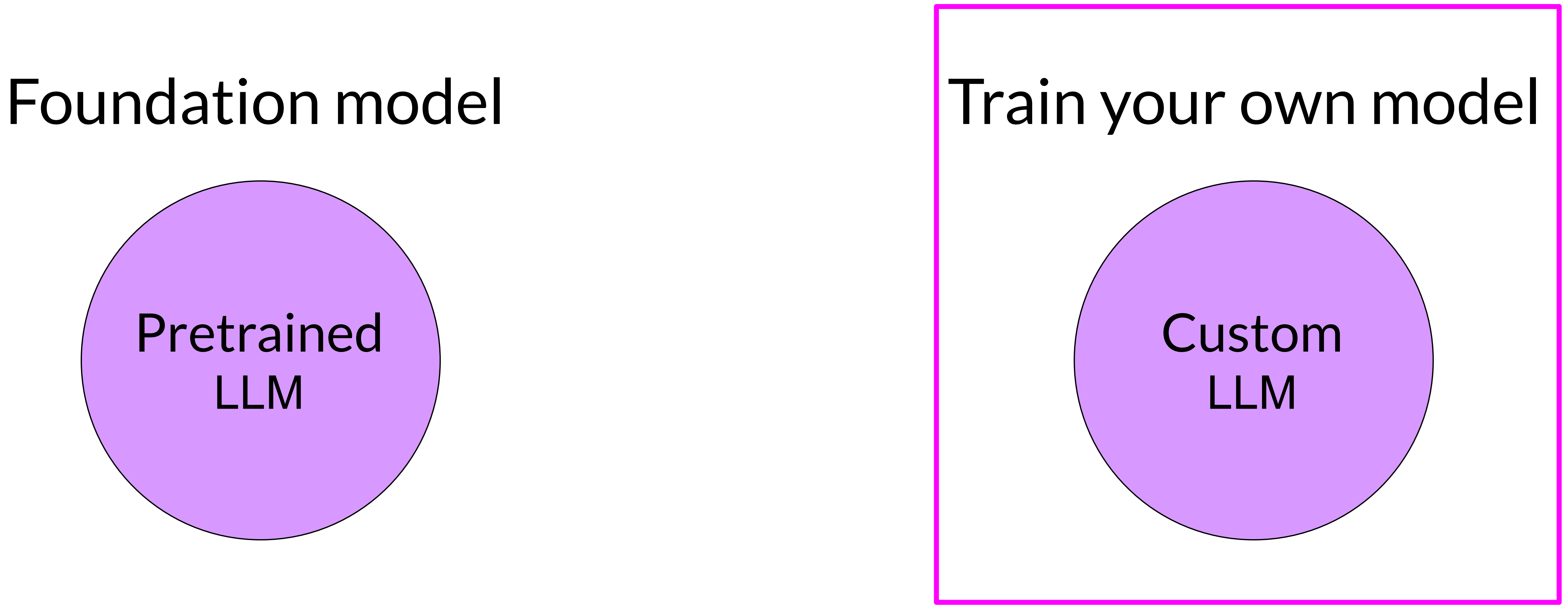
Generative AI project lifecycle


Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

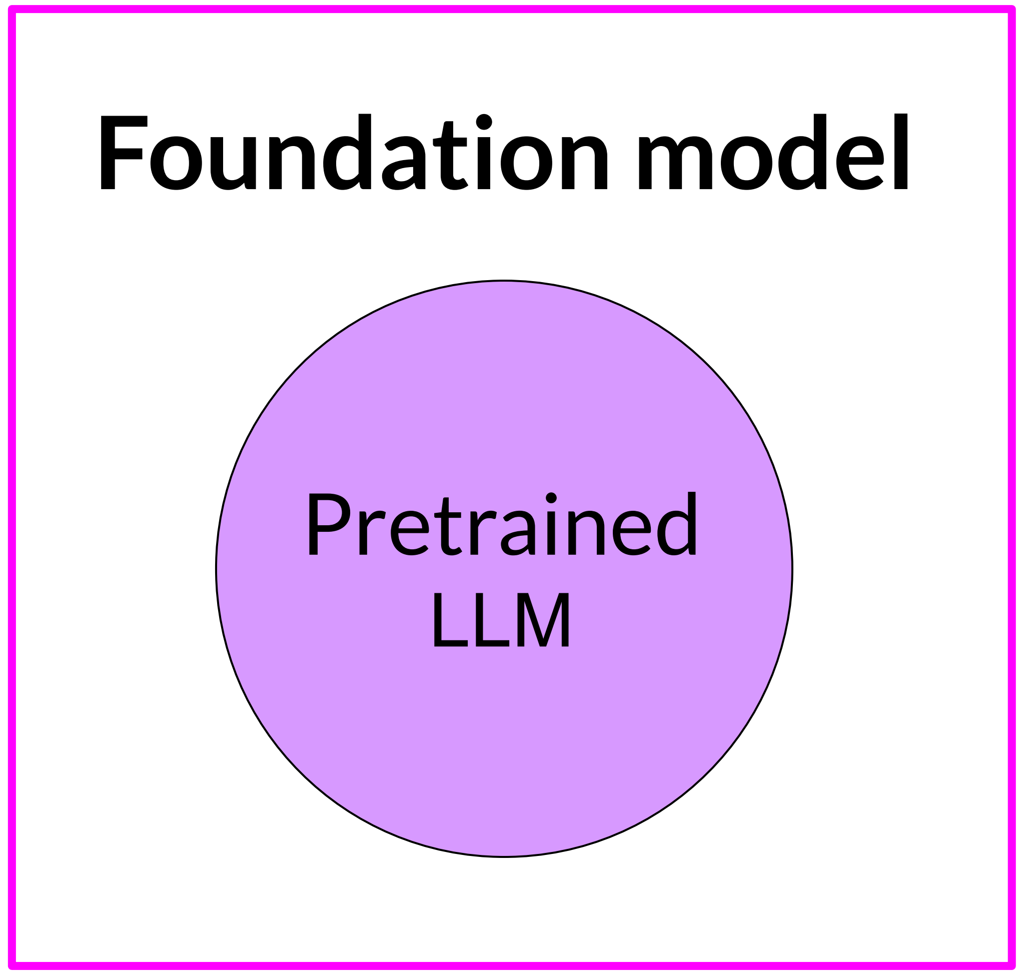
Generative AI project lifecycle

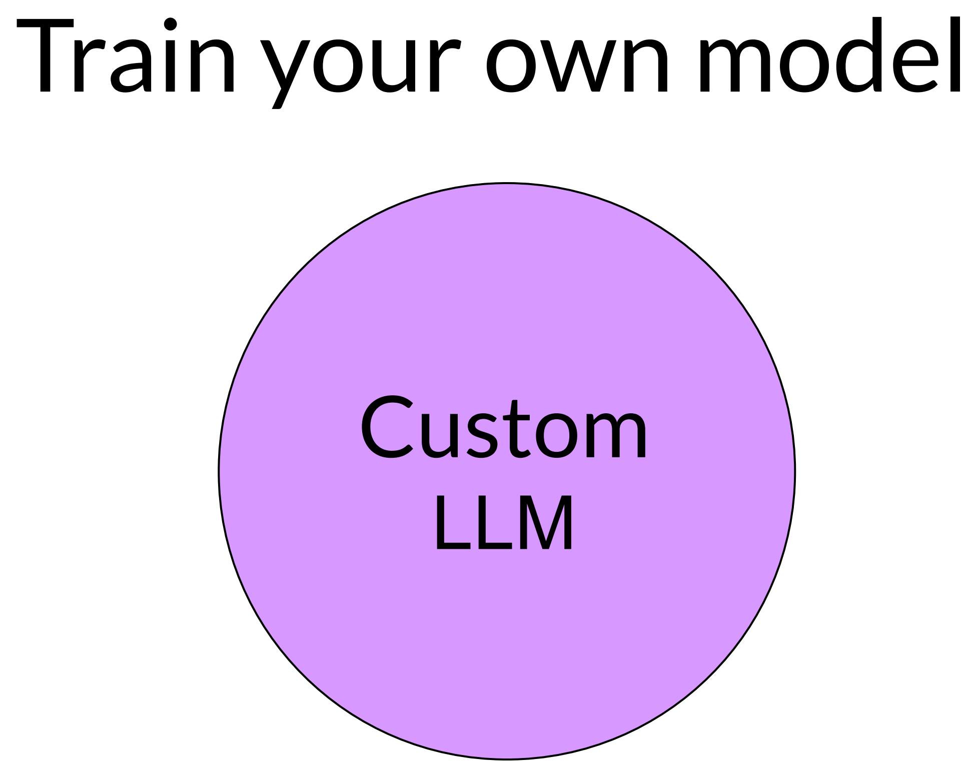

Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

Considerations for choosing a model



Considerations for choosing a model





Model hubs

Model Card for T5 Large

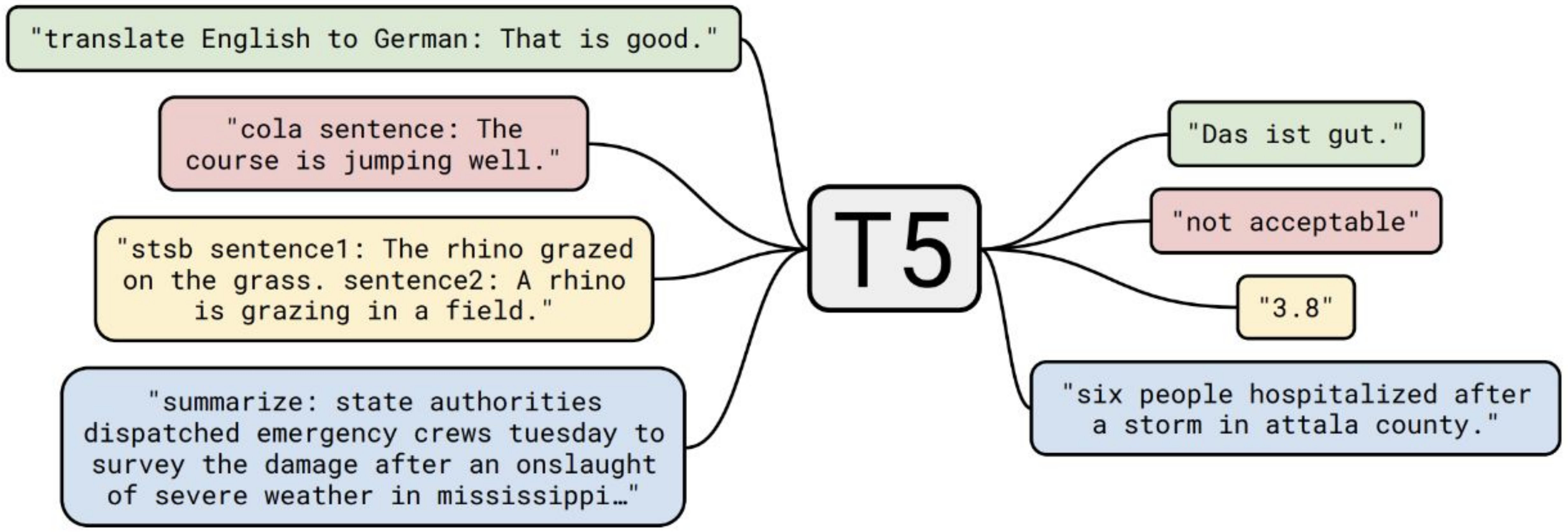
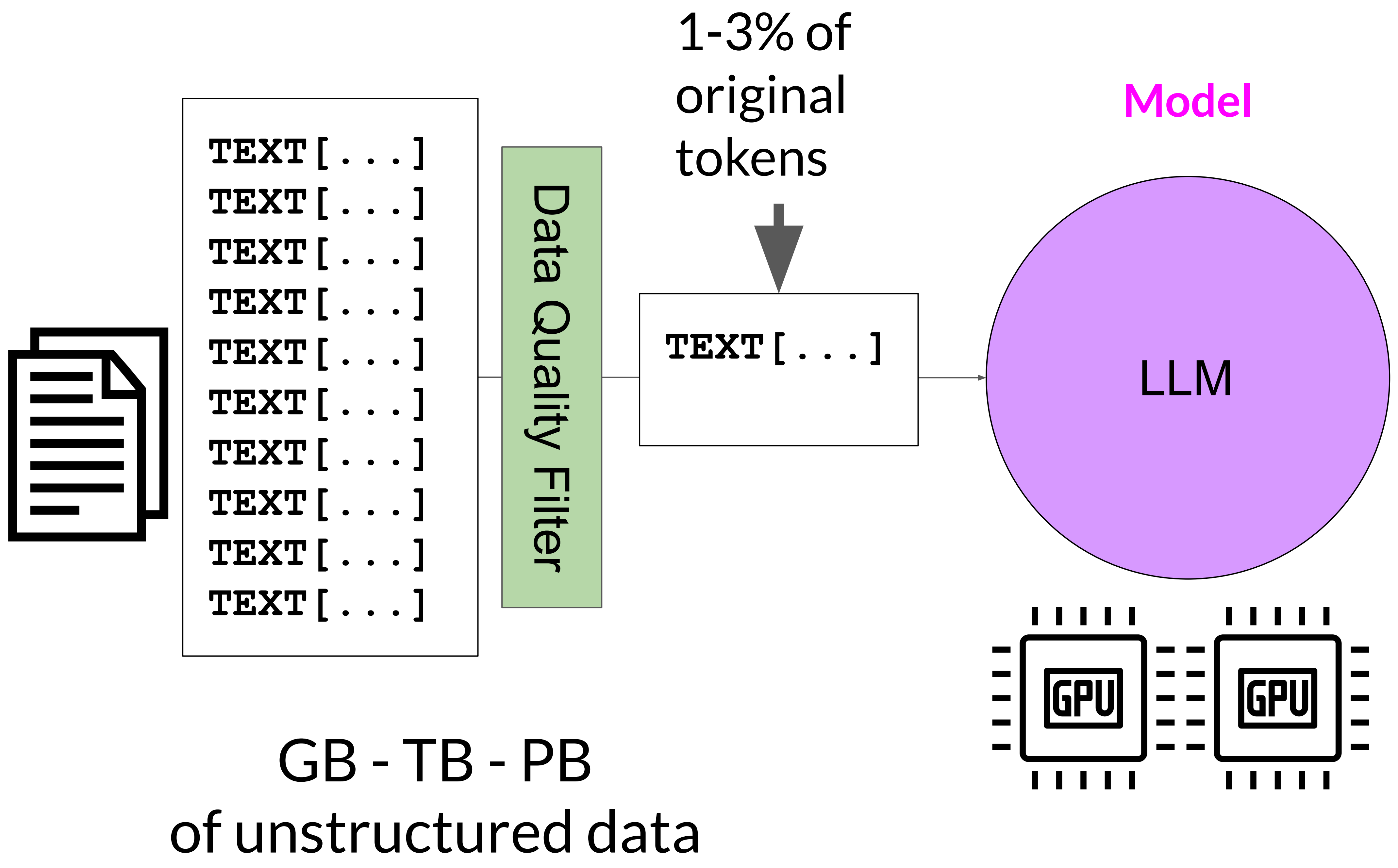

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Table of Contents

Model architectures and pre-training objectives

LLM pre-training at a high level



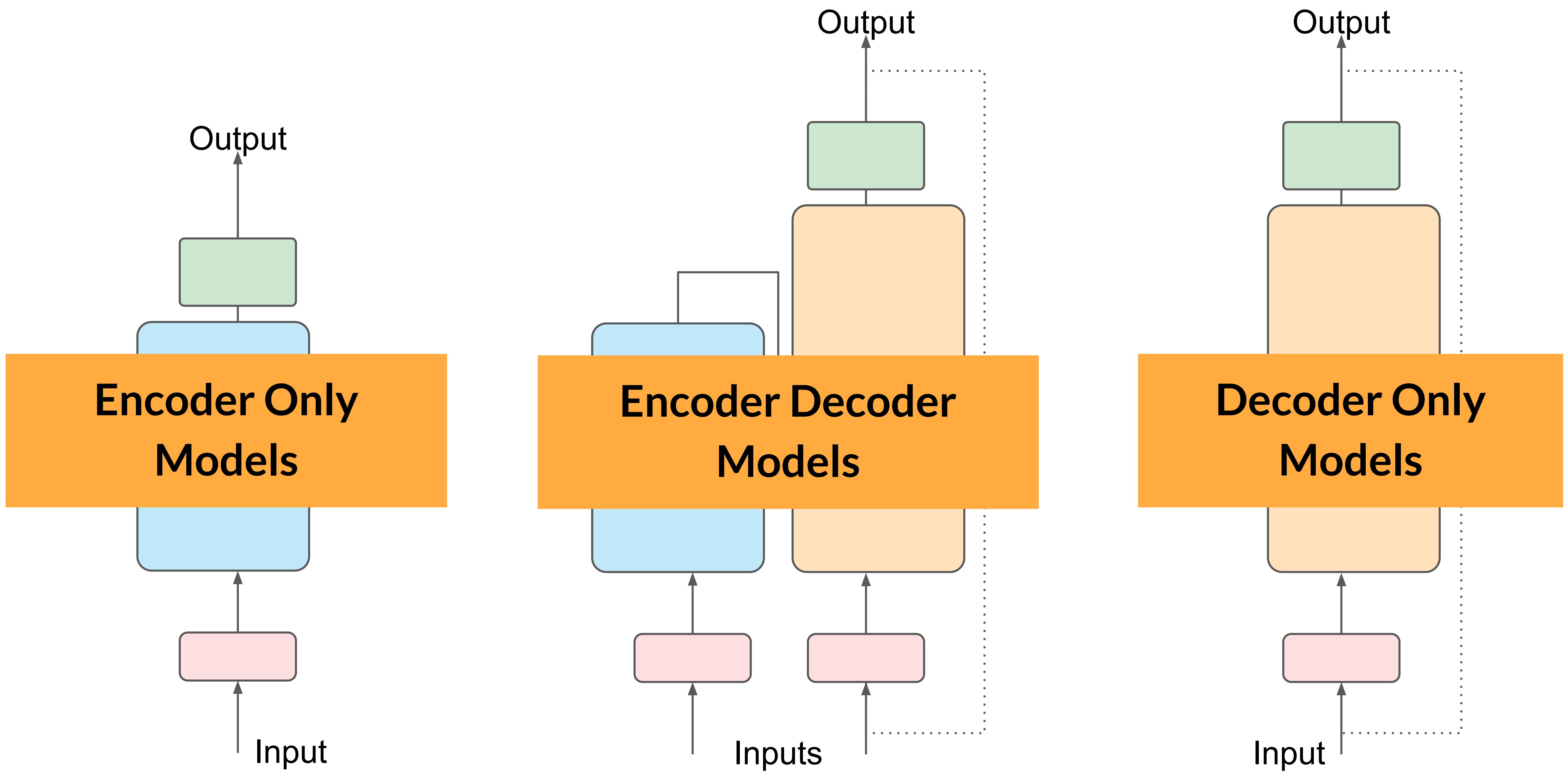
Token String	Token ID	Embedding / Vector Representation
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...	...	...

Vocabulary

DeepLearning.AI

aws

Transformers



Autoencoding models

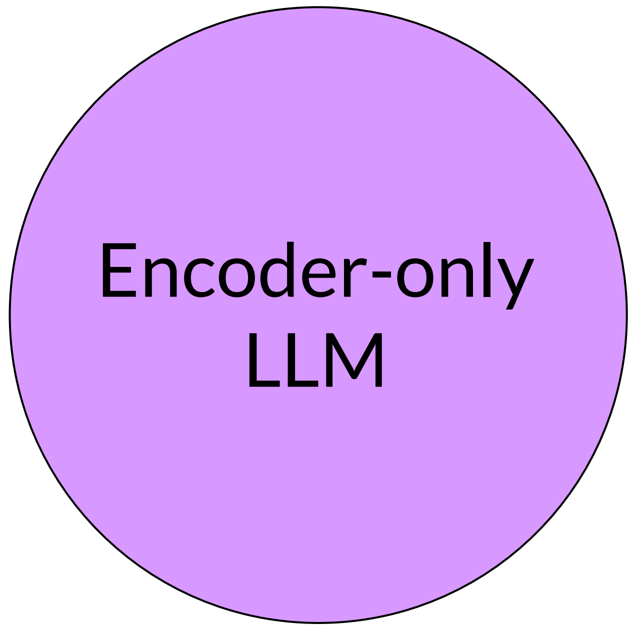
Original text

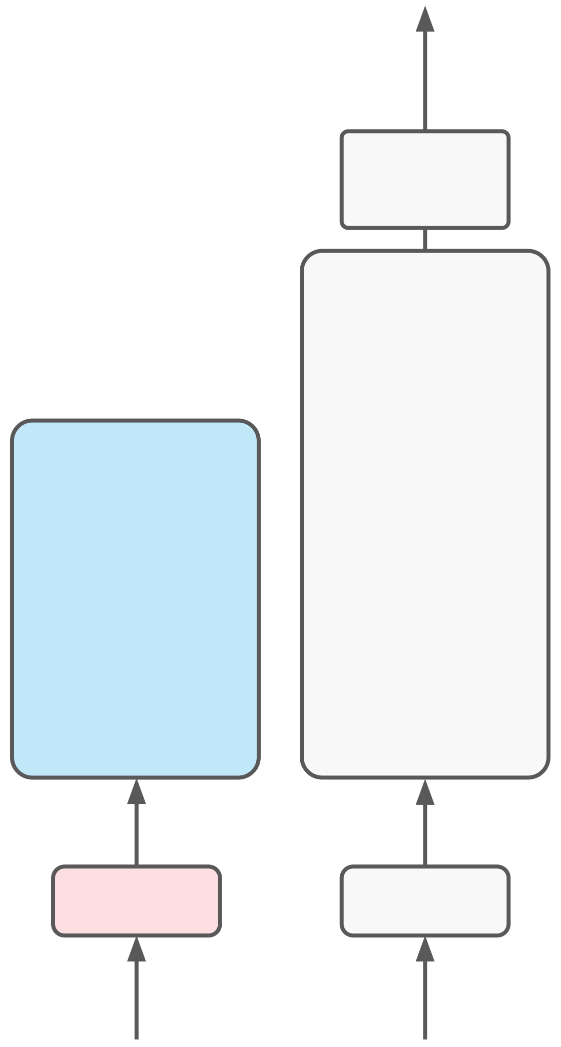
The teacher teaches the student.

[...]

Masked Language Modeling (MLM)

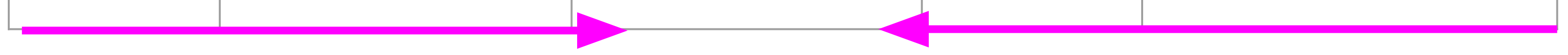
The	teacher	**<MASK>**	the	student





Objective: Reconstruct text ("denoising")

The	teacher	teaches	the	student



Bidirectional context

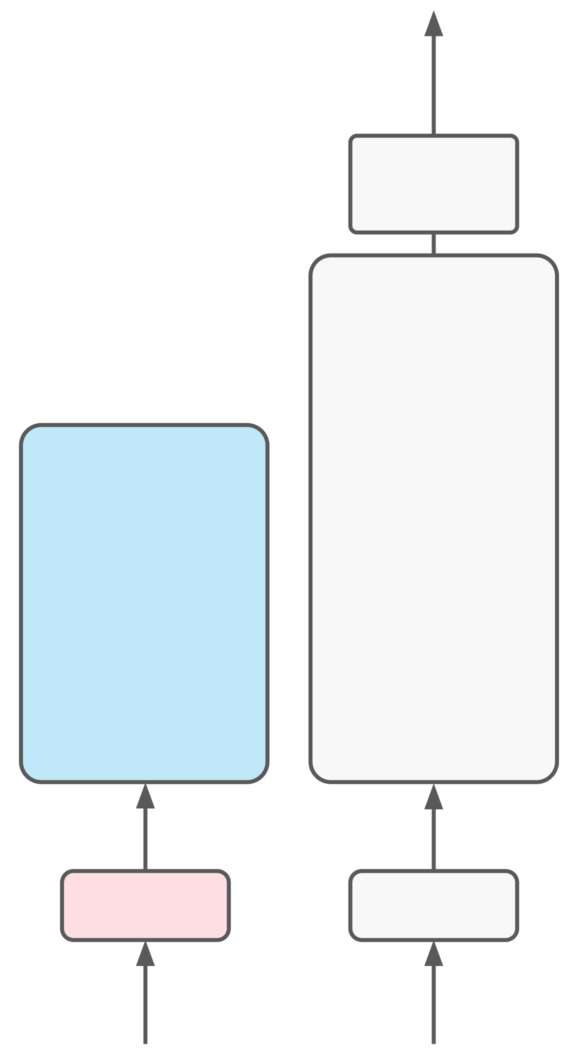
Autoencoding models

Good use cases:

- Sentiment analysis
- Named entity recognition
- Word classification

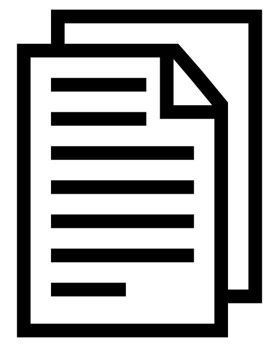
Example models:

- BERT
- ROBERTA



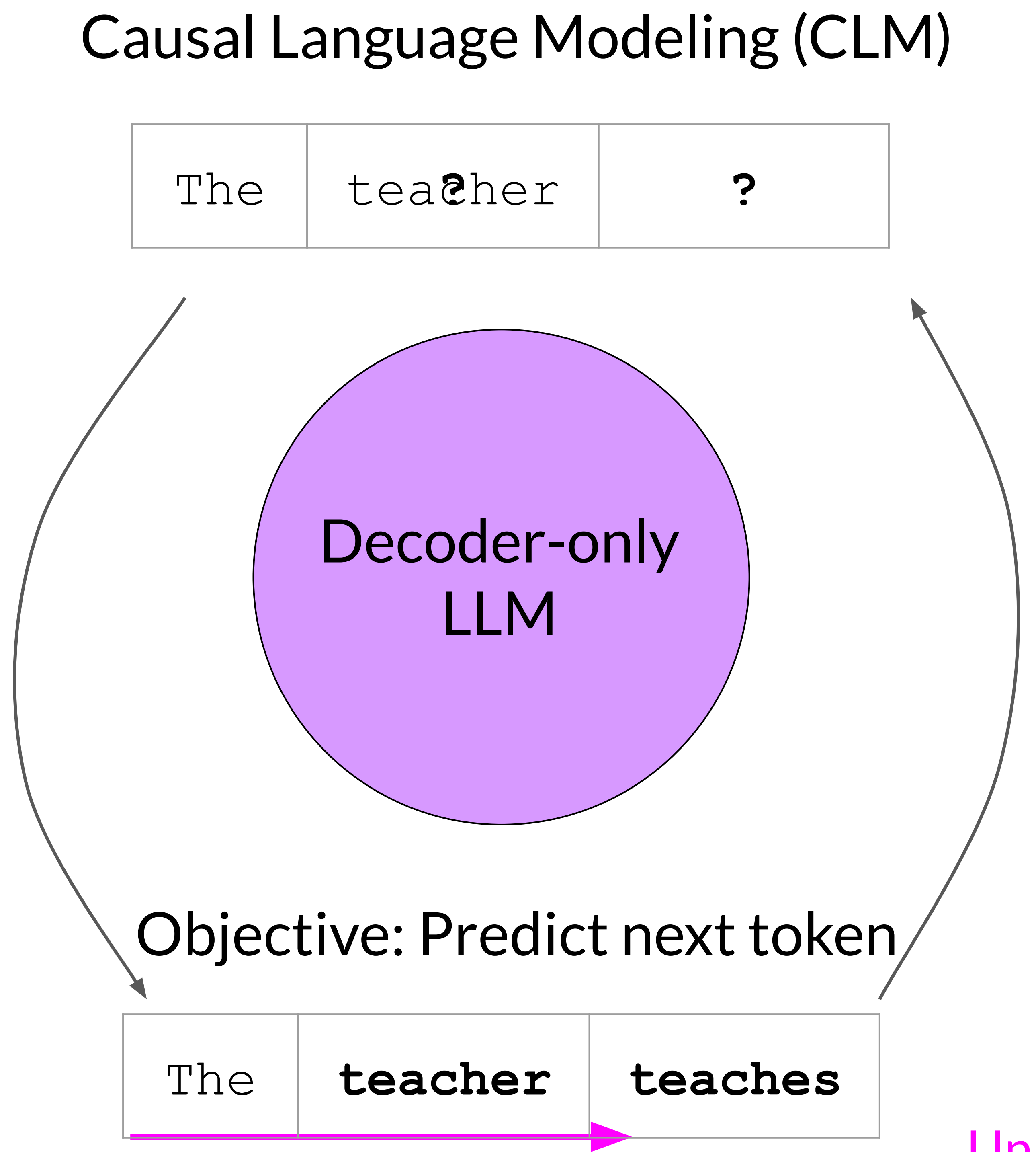
Autoregressive models

Original text

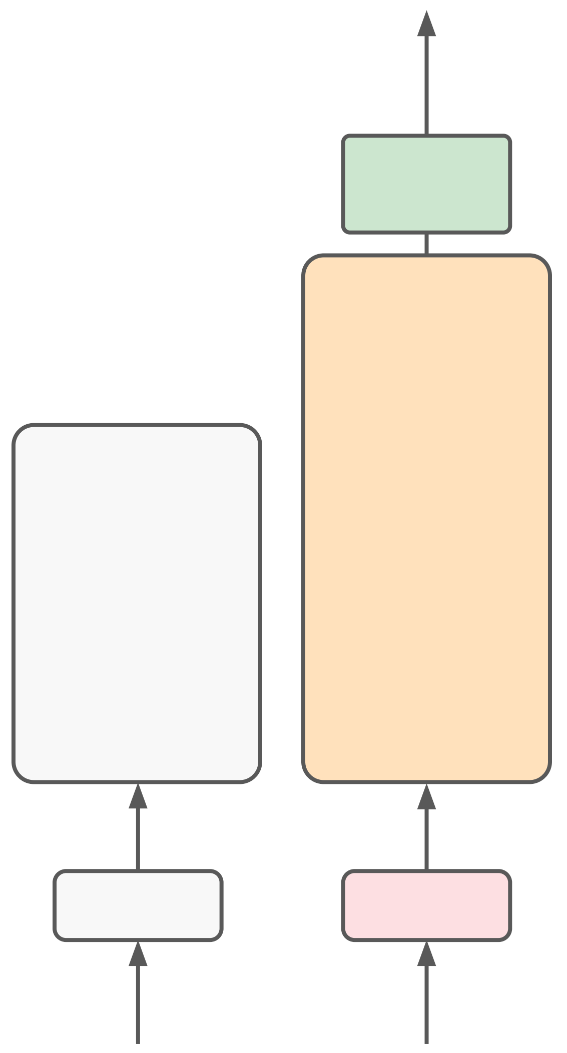


The teacher teaches the student.

[...]



Unidirectional context



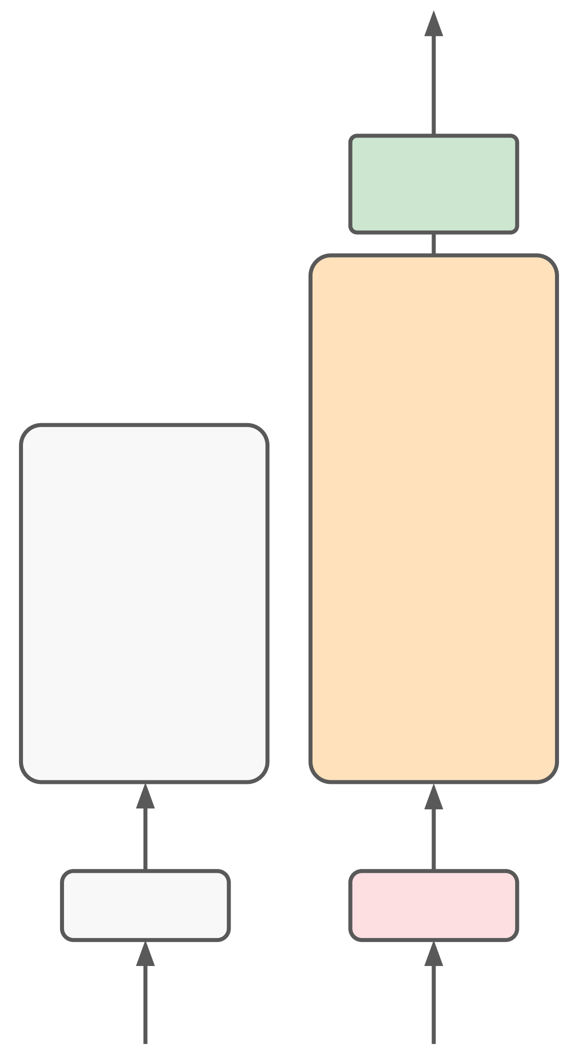
Autoregressive models

Good use cases:

- Text generation
- Other emergent behavior
  - Depends on model size

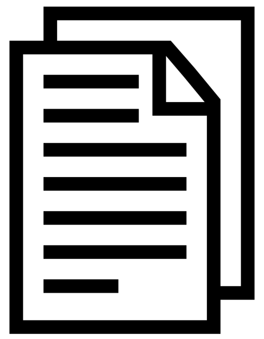
Example models:

- GPT
- BLOOM



Sequence-to-sequence models

Original text

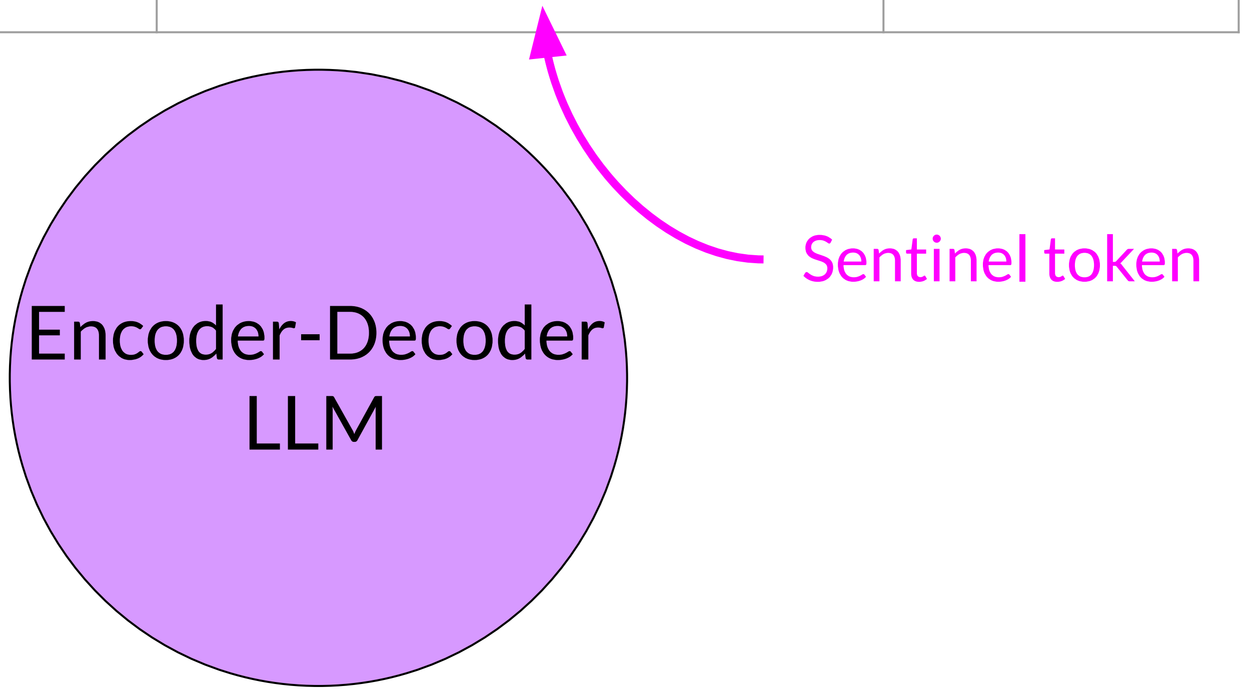


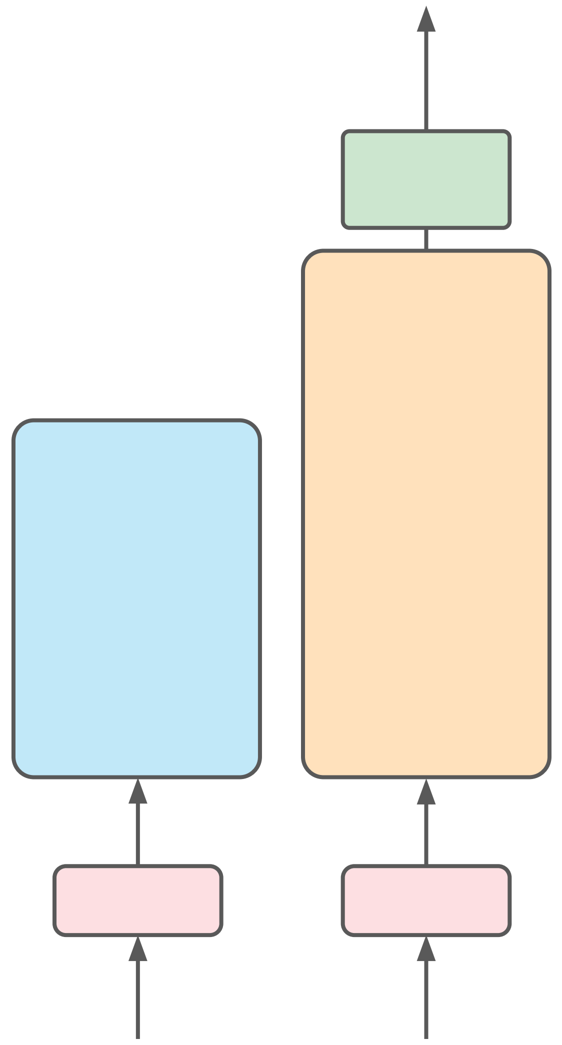
```
The teacher
teaches the
student.

[...]
```

Span Corruption

The	teacher	**<MASK>**	**<MASK>**	student

The	teacher	**<X>**	student





Objective: Reconstruct span

<x>	**teaches**	**the**

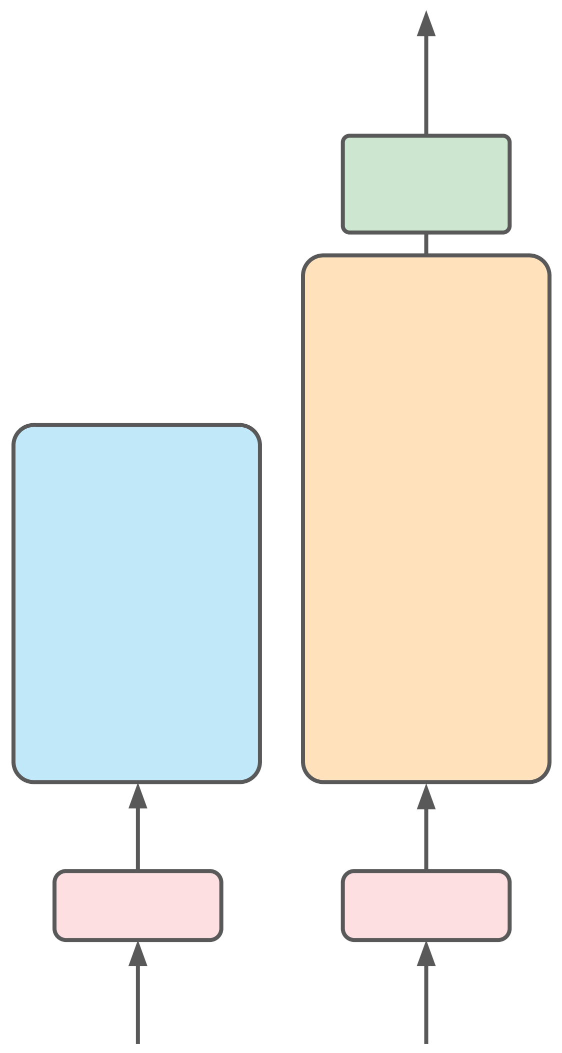
Sequence-to-sequence models

Good use cases:

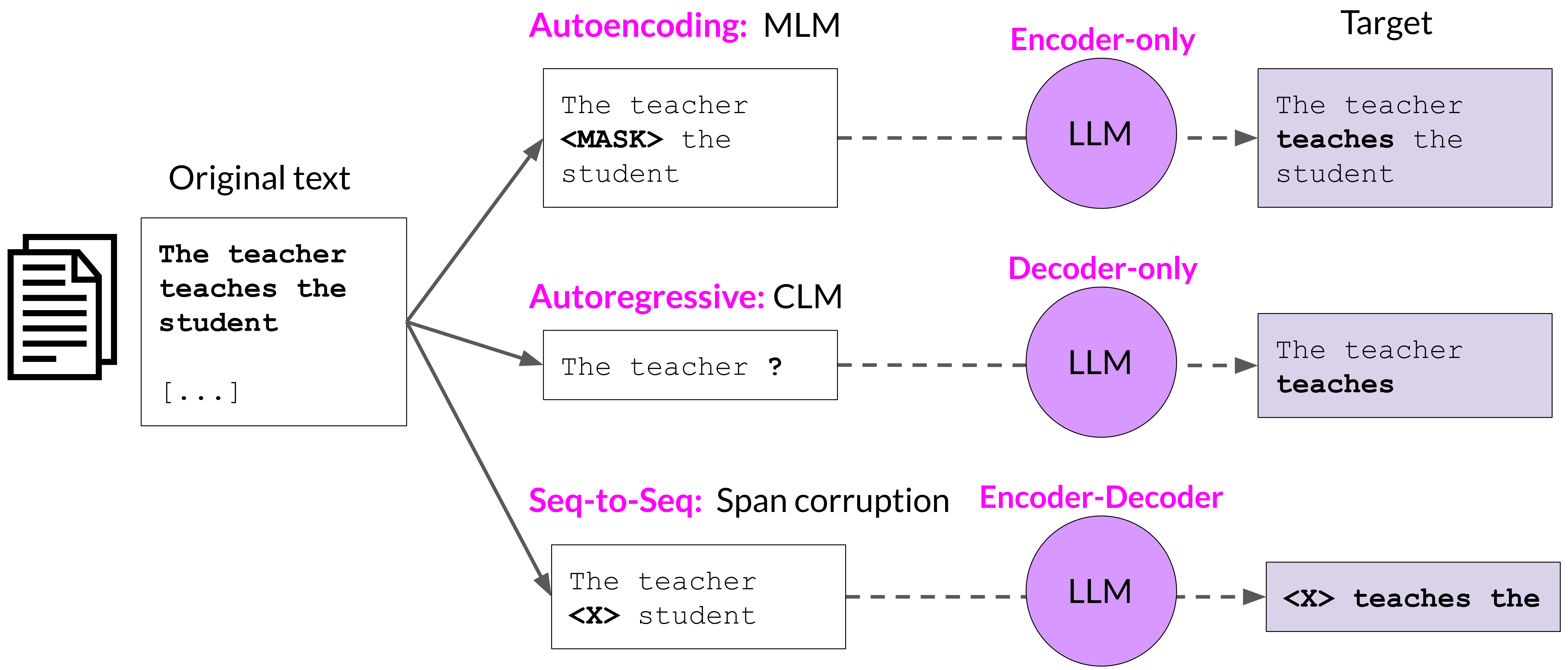
- Translation
- Text summarization
- Question answering

Example models:

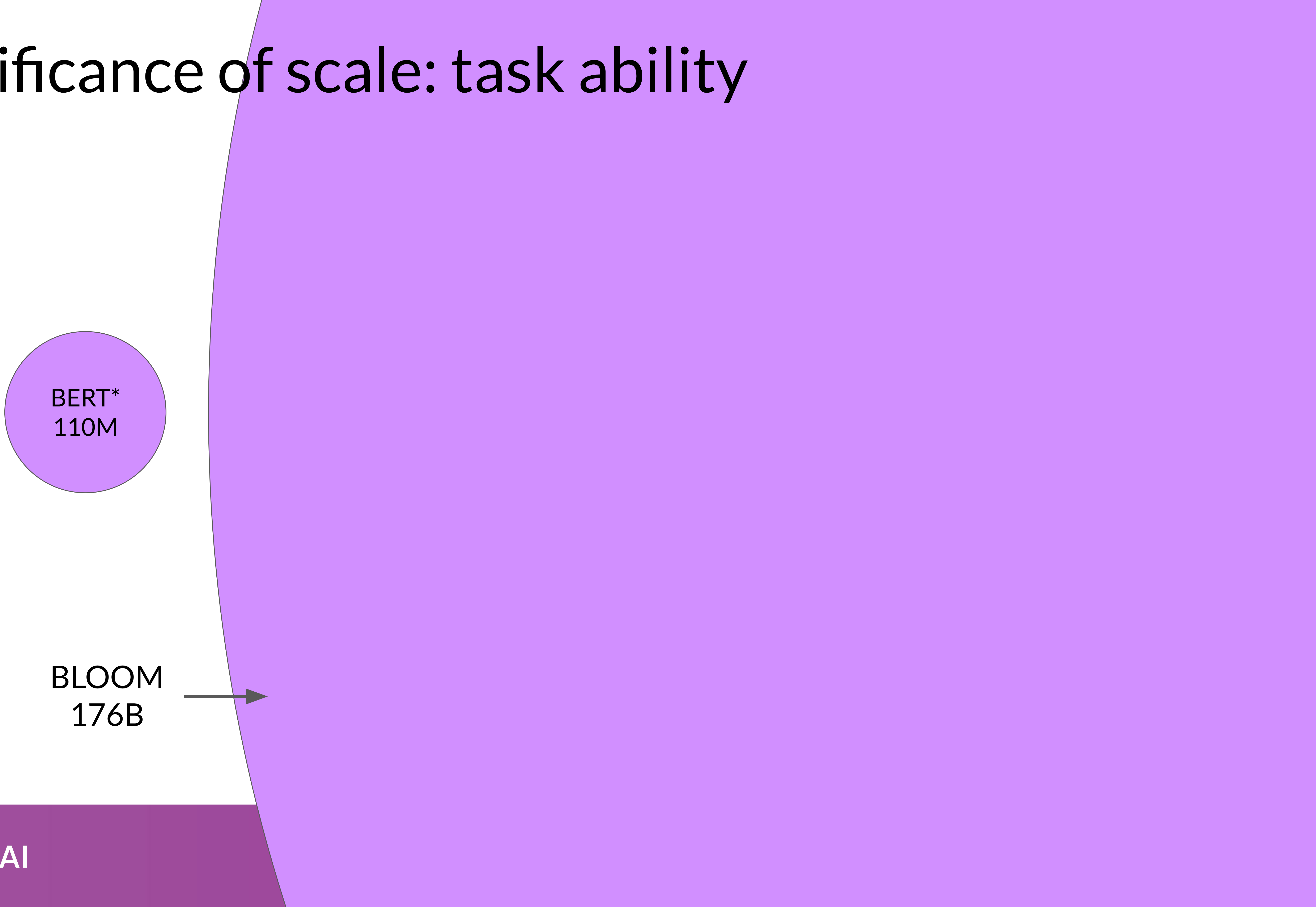
- T5
- BART



Model architectures and pre-training objectives

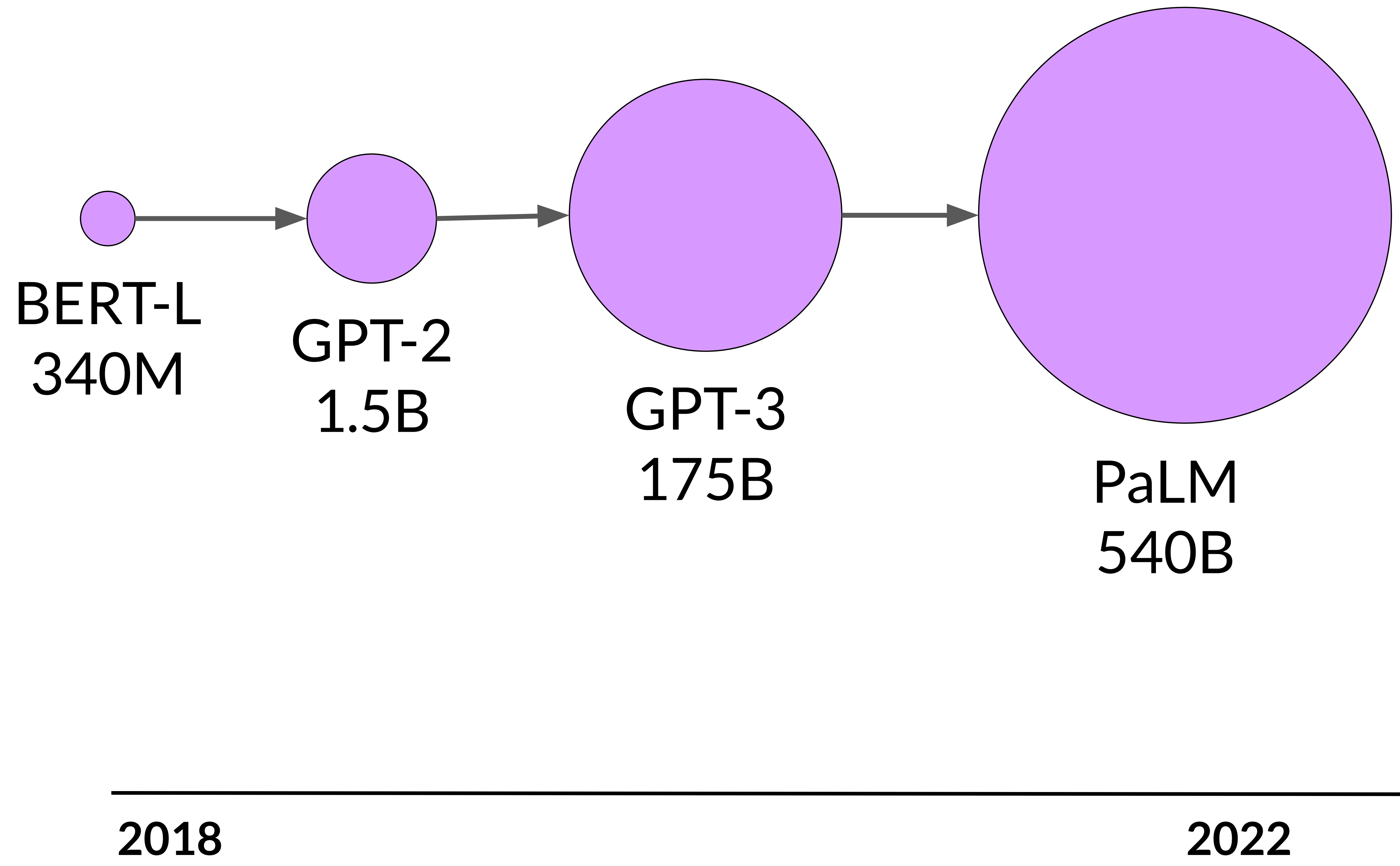


The significance of scale: task ability



*Bert-base

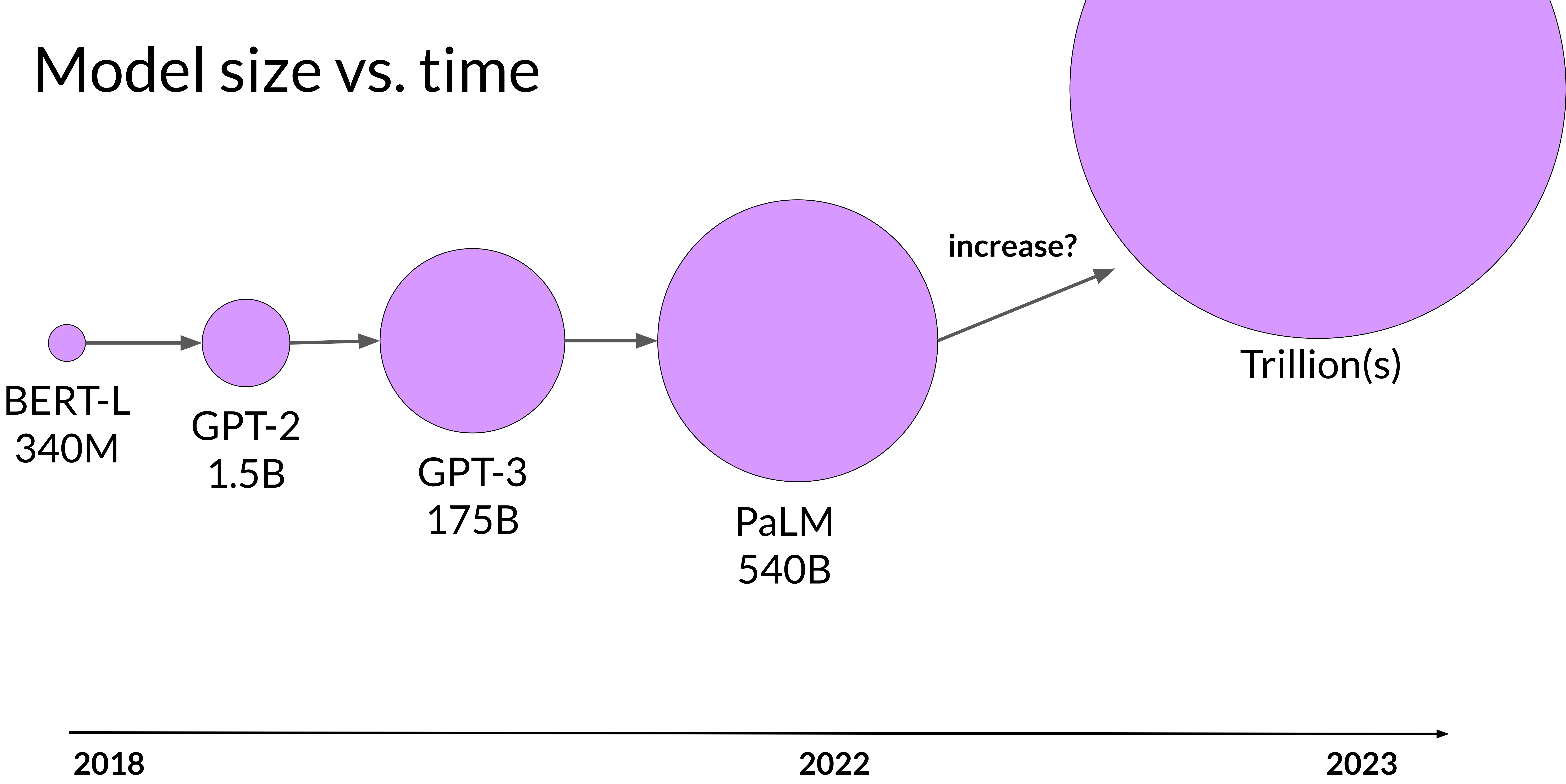
Model size vs. time



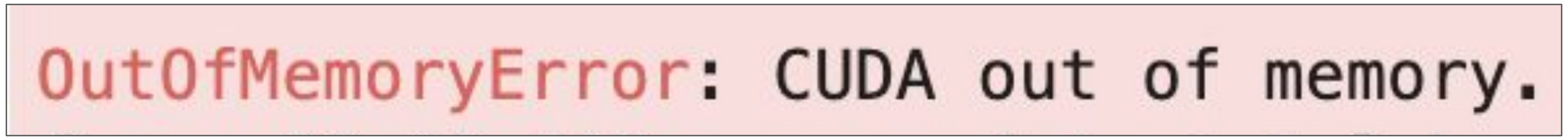
Growth powered by:

- Introduction of transformer
- Access to massive datasets
- More powerful compute resources

Model size vs. time



Computational challenges



```
OutOfMemoryError: CUDA out of memory.
```

Approximate GPU RAM needed to store 1B parameters

1 parameter = 4 bytes (32-bit float)

1B parameters = 4 x 10^9 bytes = 4GB

4GB @ 32-bit full precision

Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

Additional GPU RAM needed to train 1B parameters

	Bytes per parameter
Model Parameters (Weights)	4 bytes per parameter

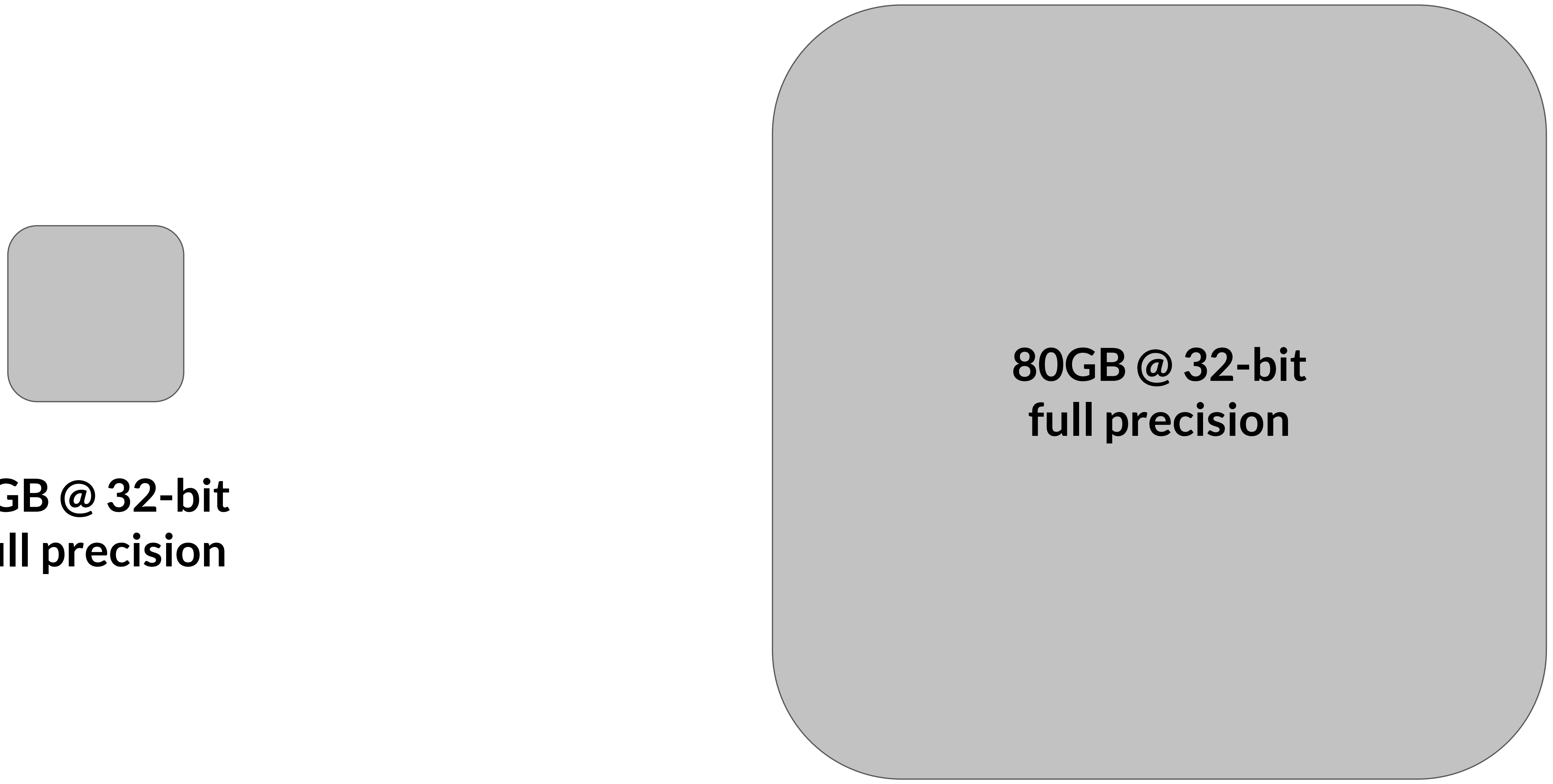
~20 extra bytes per parameter

Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

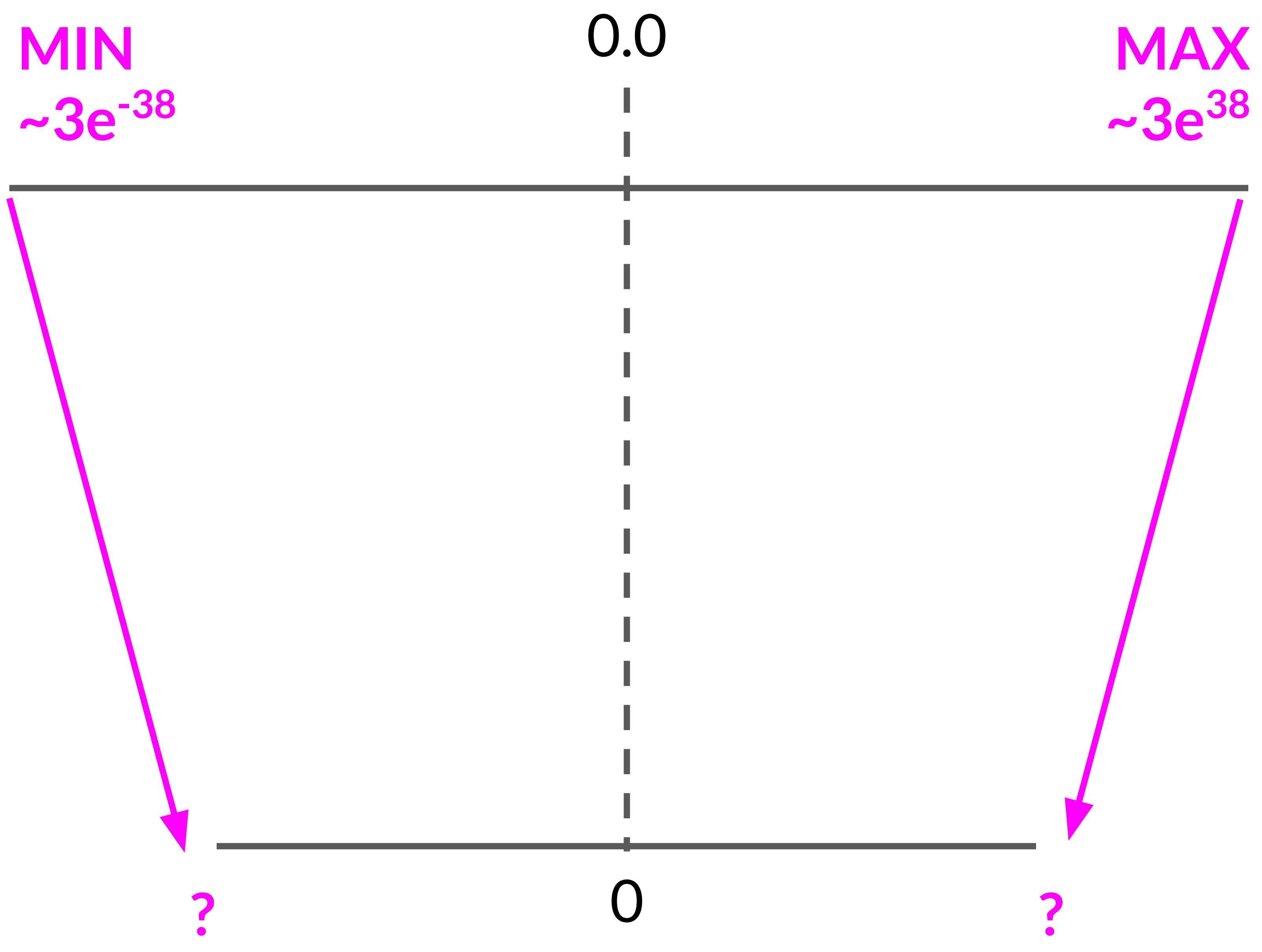
Approximate GPU RAM needed to train 1B-params

Memory needed to store model

Memory needed to train model



Quantization



FP32

32-bit floating point

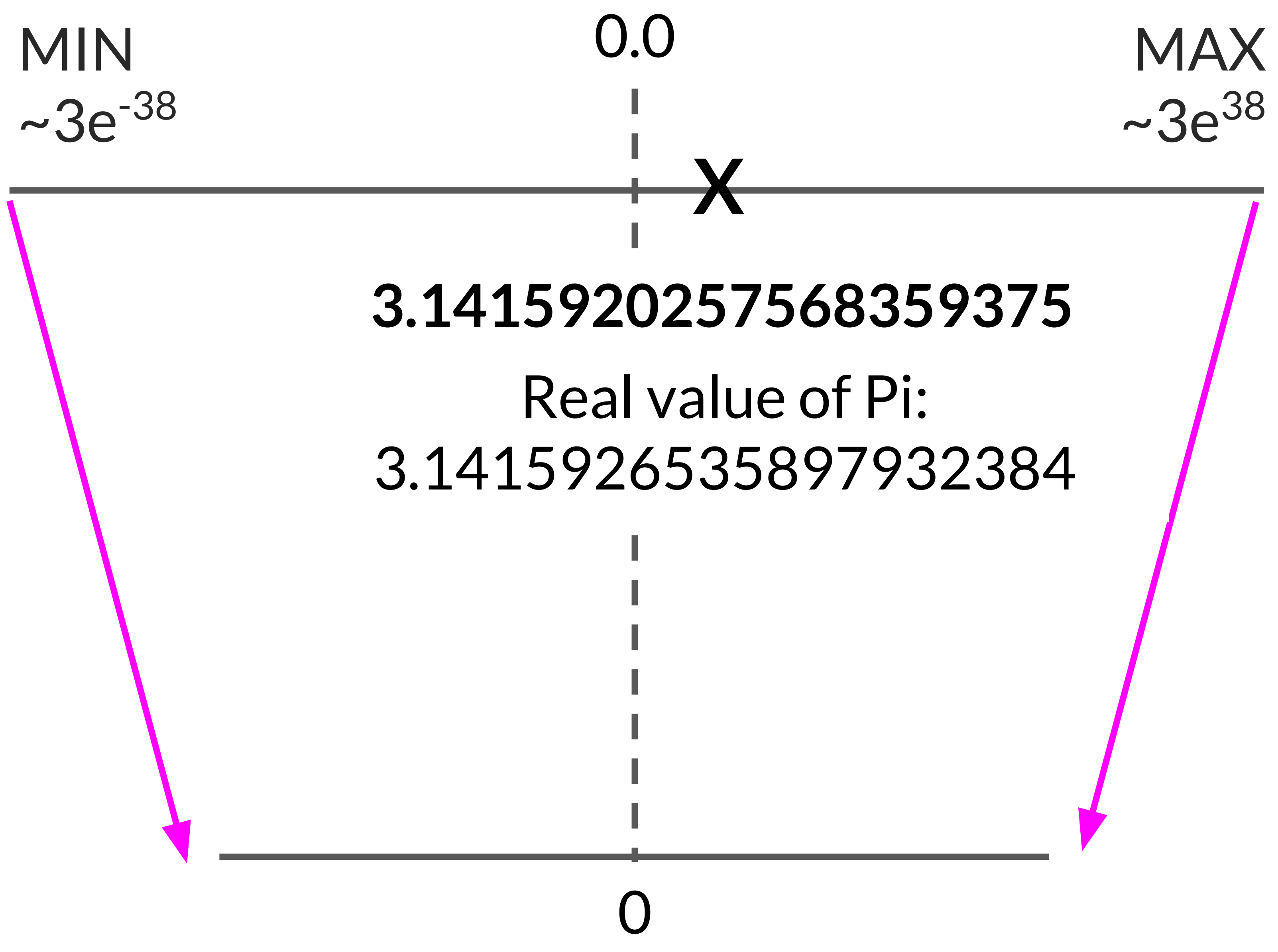
Range:

From **~3e^{-38}** to **~3e^{38}**

FP16 | BFLOAT16 | INT8

16-bit floating point | 8-bit integer

Quantization: FP32



3.1415920257568359375

Real value of Pi:
3.14159265358979323

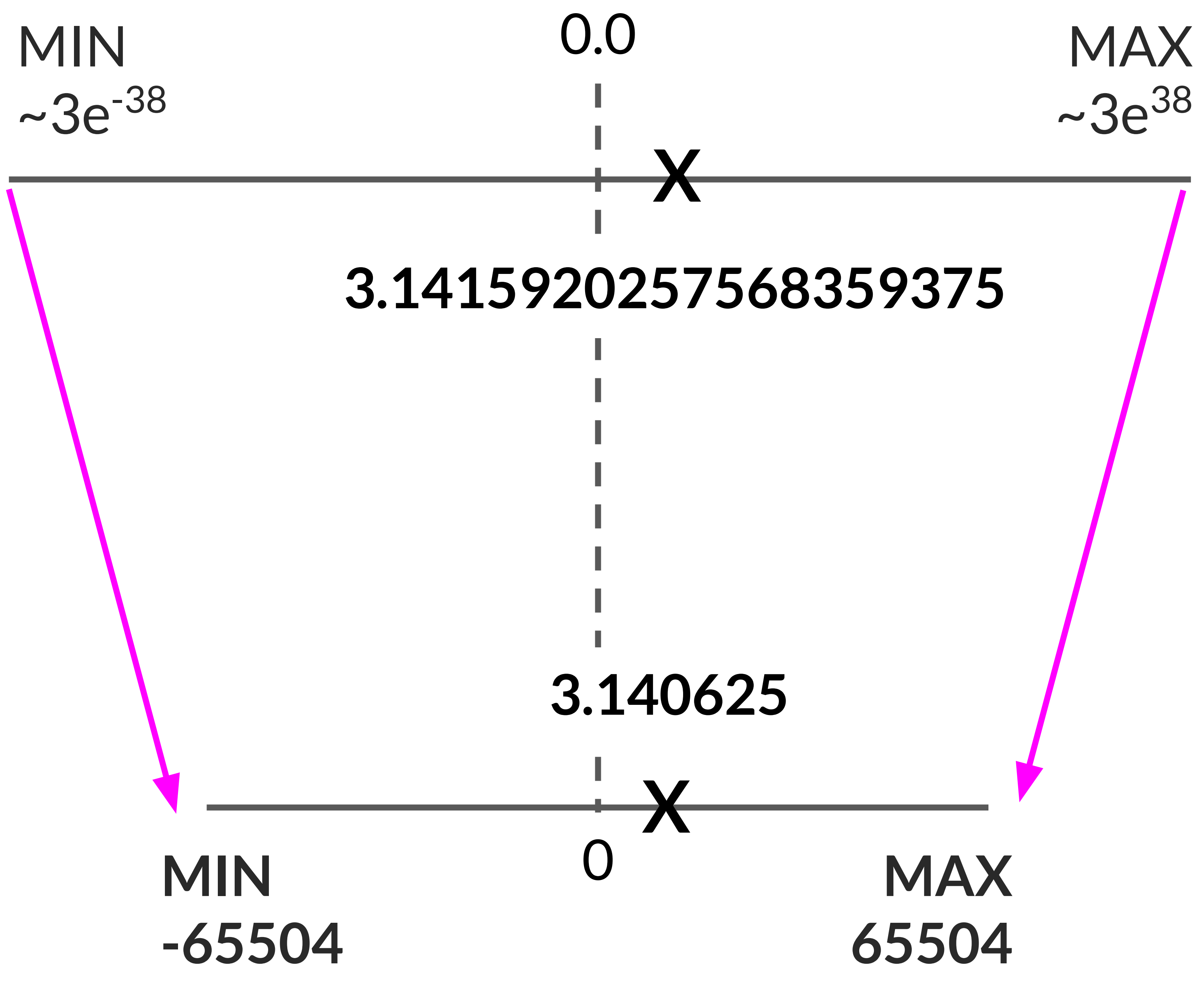
Let's store Pi: 3.141592

FP32

$$\underbrace{\mathbf{0}}_{\substack{\textbf{Sign}\\ \text{1 bit}}}\quad \underbrace{\mathbf{10000000}}_{\substack{\textbf{Exponent}\\ \text{8 bits}}}\quad \underbrace{\mathbf{10010010000111111011000}}_{\substack{\textbf{Fraction}\\ \text{23 bits}}}$$

Mantissa / Significand
= Precision

Quantization: FP16



Let's store Pi: 3.141592

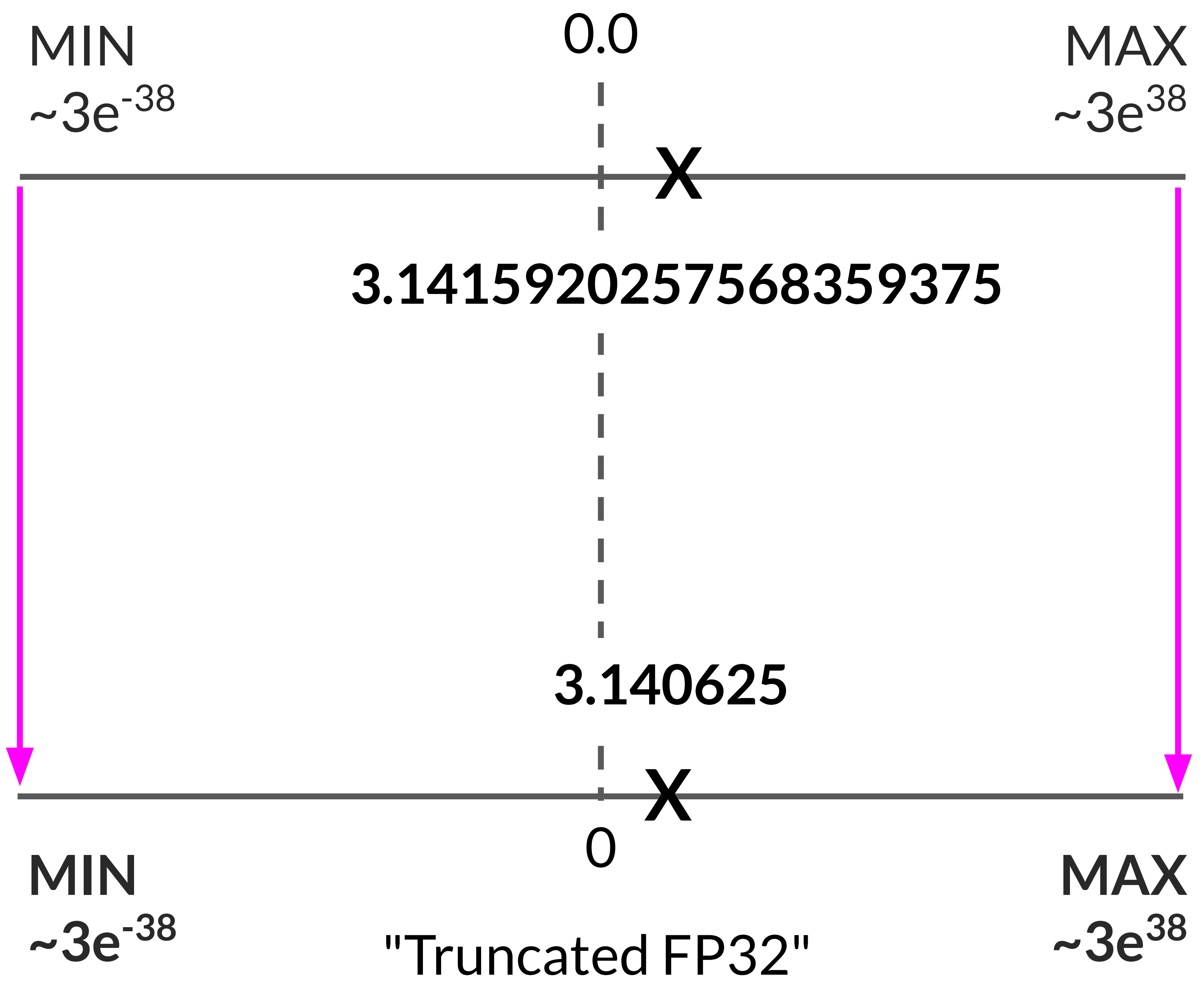
FP32 **4 bytes memory**

$$\underbrace{0}_{\text{Sign, 1 bit}}\quad\underbrace{10000000}_{\text{Exponent, 8 bits}}\quad\underbrace{10010010000111111011000}_{\text{Fraction, 23 bits}}$$

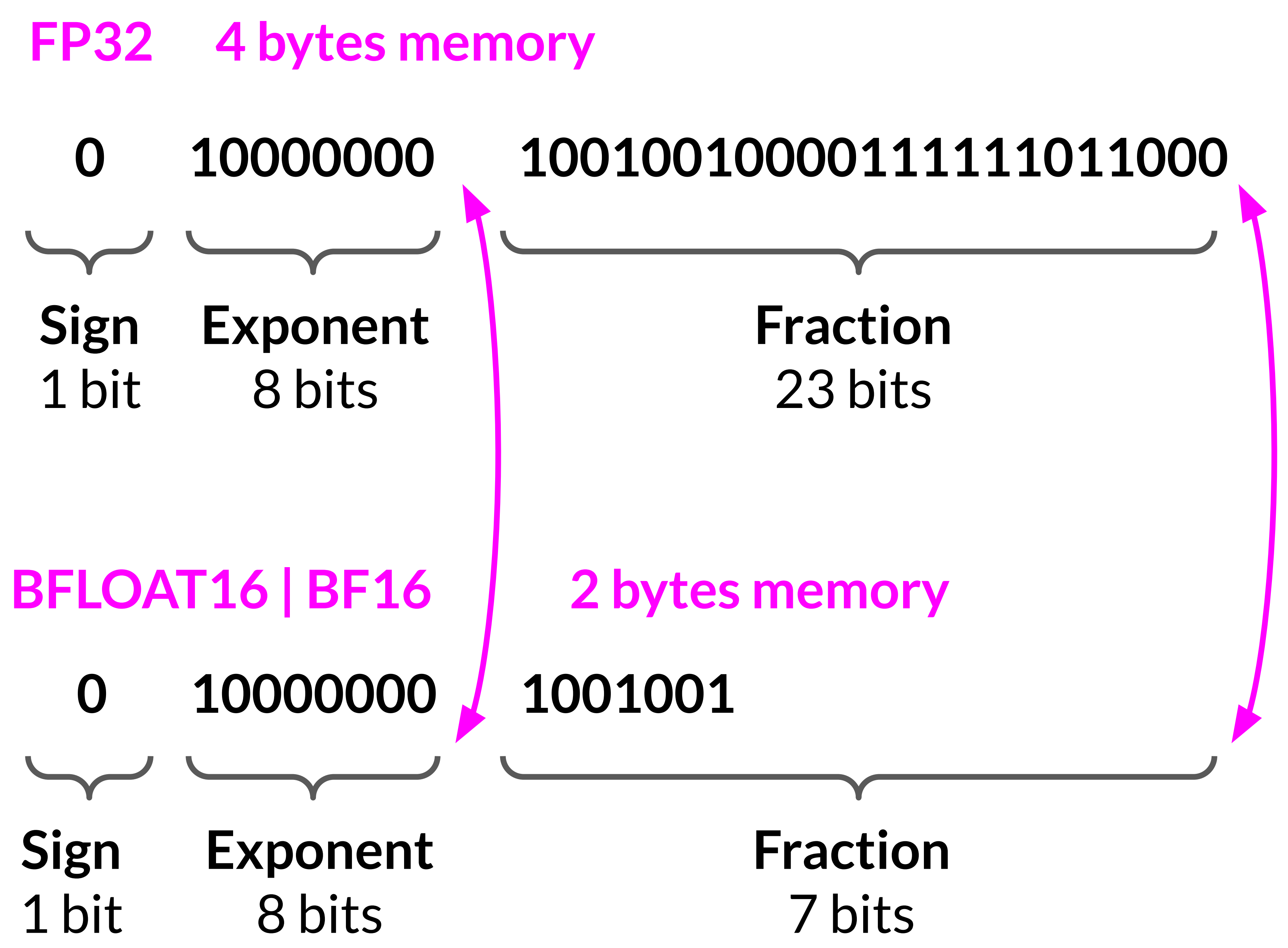
FP16 **2 bytes memory**

$$\underbrace{0}_{\text{Sign, 1 bit}}\quad\underbrace{10000}_{\text{Exponent, 5 bits}}\quad\underbrace{1001001000}_{\text{Fraction, 10 bits}}$$

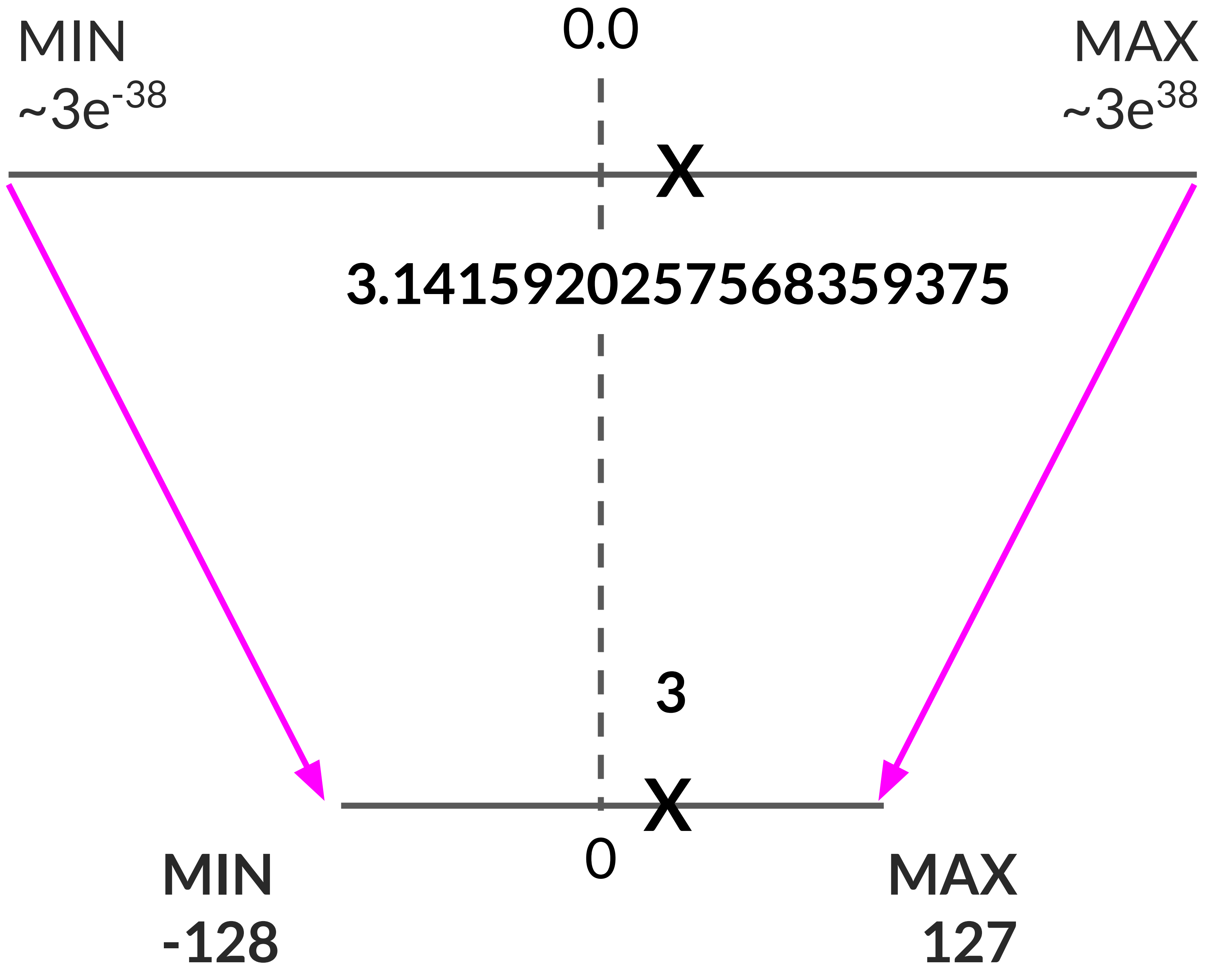
Quantization: BFLOAT16



Let's store Pi: 3.141592



Quantization: INT8



Let's store Pi: 3.141592

FP32 **4 bytes memory**

$$\underbrace{0}_{\text{Sign, 1 bit}}\quad\underbrace{10000000}_{\text{Exponent, 8 bits}}\quad\underbrace{10010010000111111011000}_{\text{Fraction, 23 bits}}$$

INT8 **1 byte memory**

$$\underbrace{0}_{\text{Sign, 1 bit}}\quad\underbrace{0000011}_{\text{Fraction, 7 bits}}$$

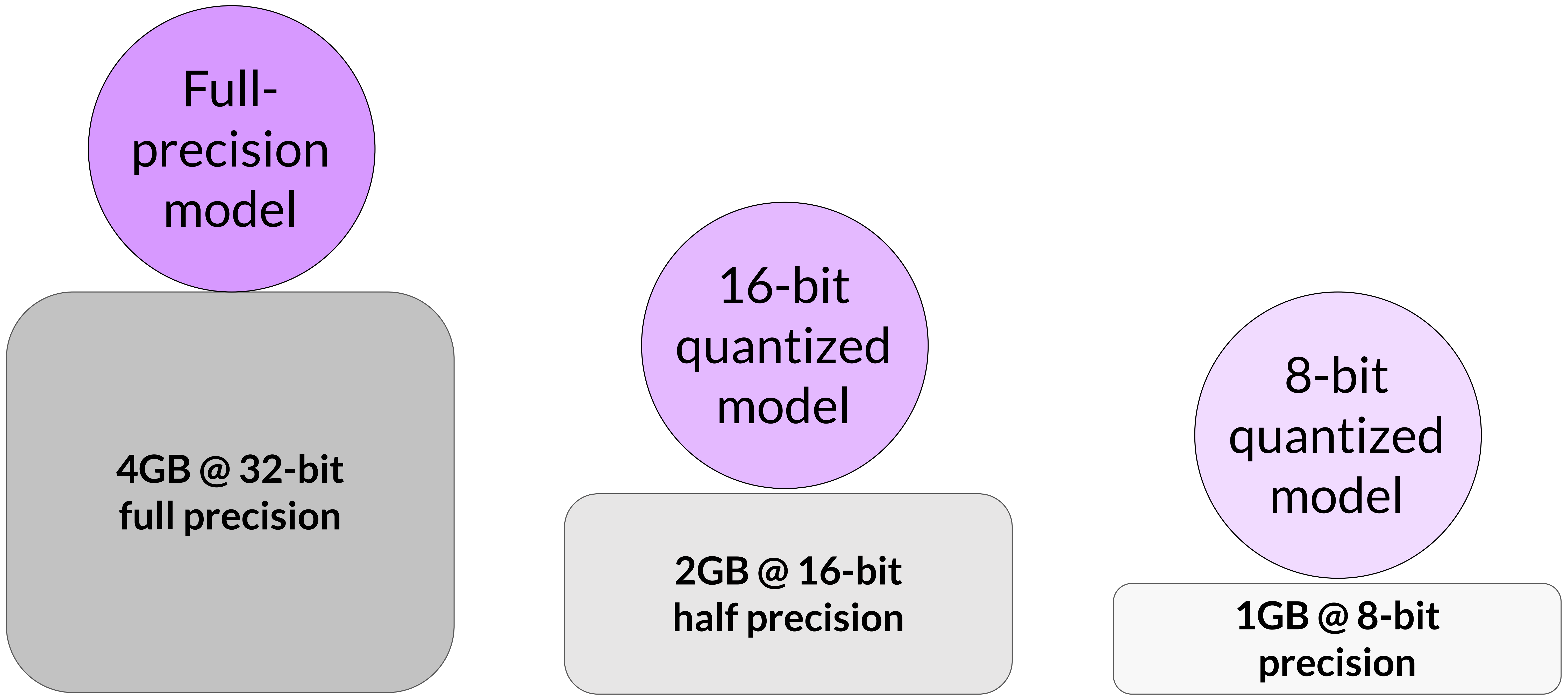
Quantization: Summary

	Bits	Exponent	Fraction	Memory needed to store one value
FP32	32	8	23	4 bytes
FP16	16	5	10	2 bytes
BFLOAT16	16	8	7	2 bytes
INT8	8	–/–	7	1 byte

FLAN T5

- Reduce required memory to store and train models
- Projects original 32-bit floating point numbers into lower precision spaces
- Quantization-aware training (QAT) learns the quantization scaling factors during training
- BFLOAT16 is a popular choice

Approximate GPU RAM needed to store 1B parameters



Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

Approximate GPU RAM needed to train 1B-params

80GB @ 32-bit full precision

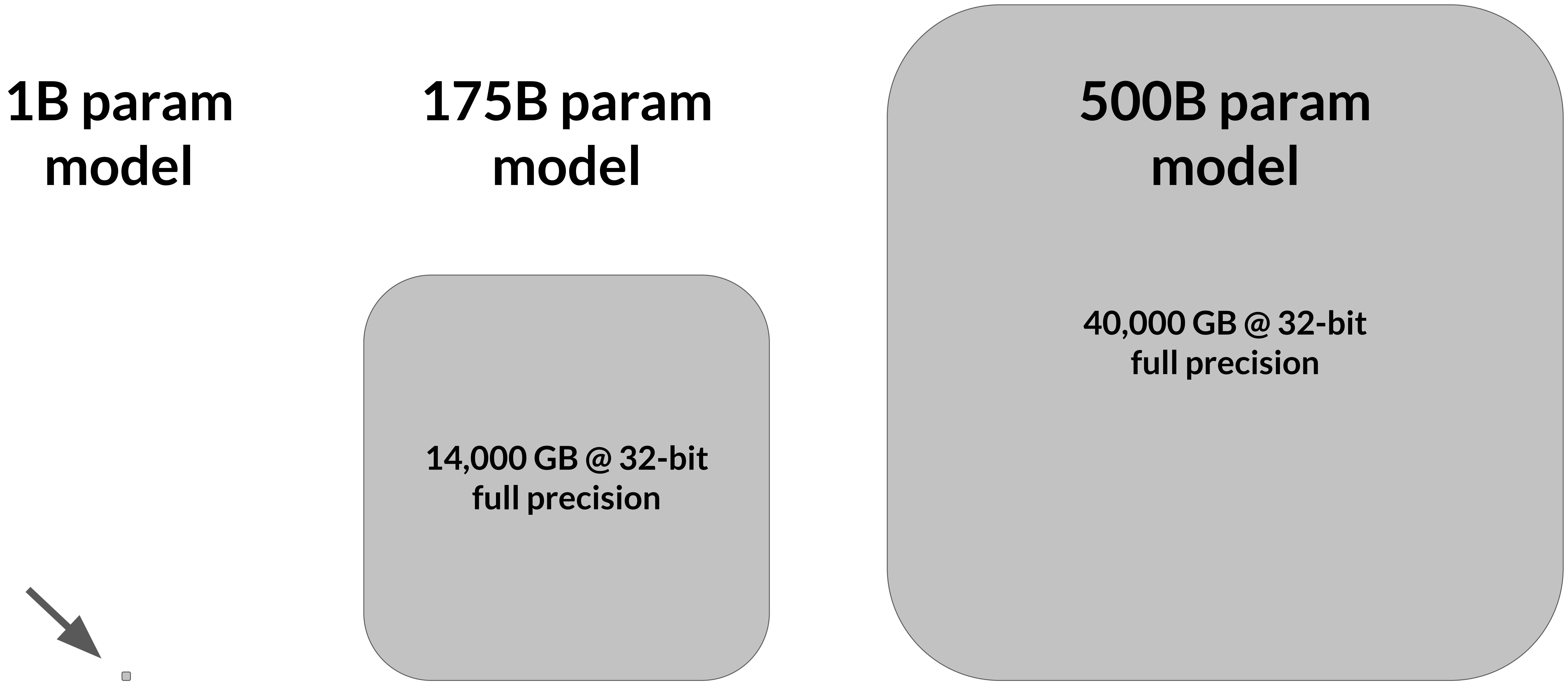
40GB @ 16-bit half precision

20GB @ 8-bit precision

80GB is the maximum memory for the Nvidia A100 GPU, so to keep the model on a single GPU, you need to use 16-bit or 8-bit quantization.

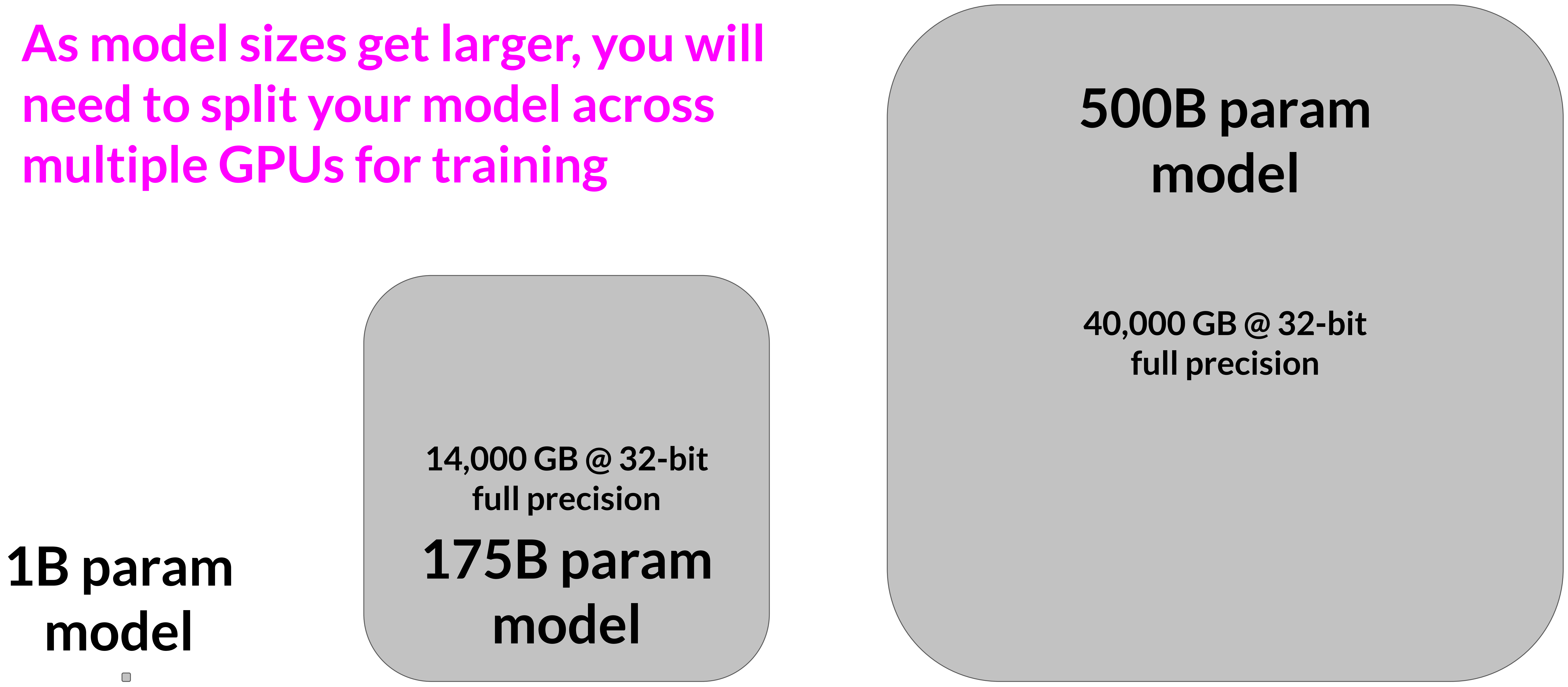
Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

GPU RAM needed to train larger models


1B param model
175B param model
500B param model
14,000 GB @ 32-bit full precision
40,000 GB @ 32-bit full precision

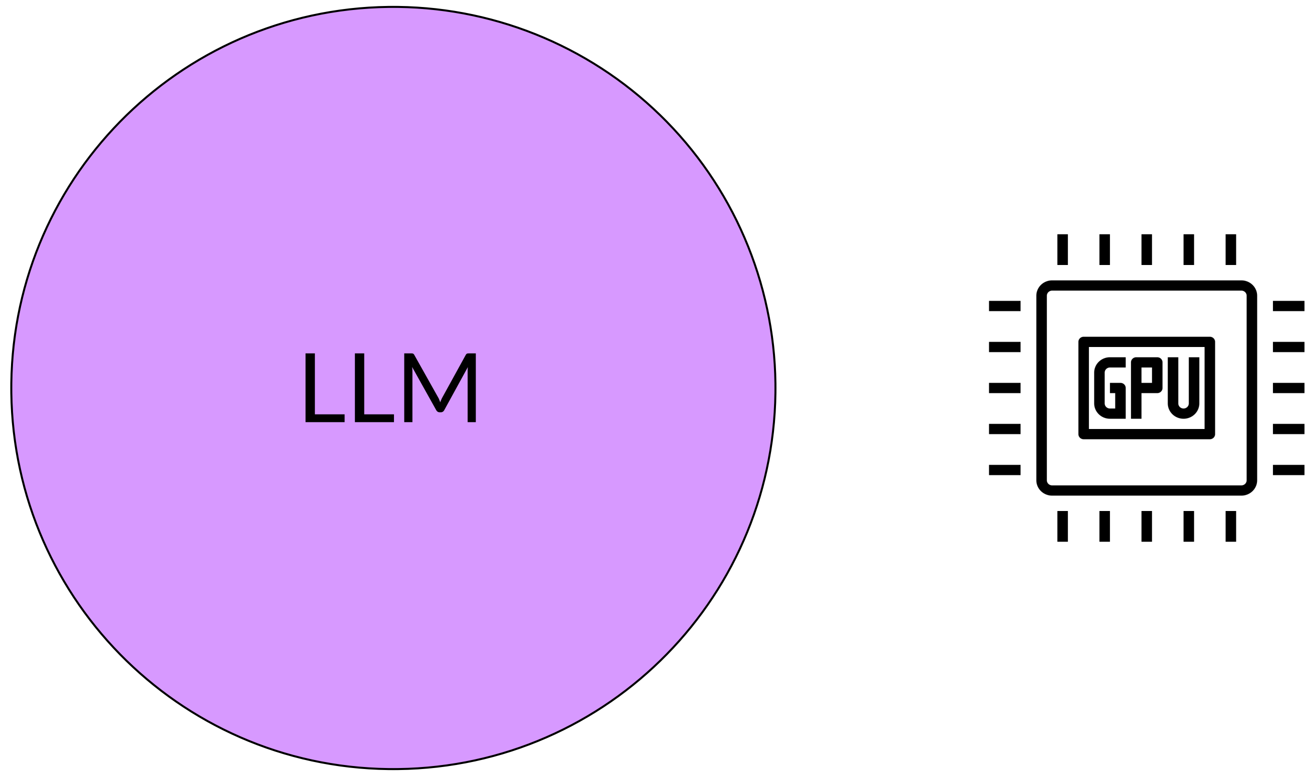
GPU RAM needed to train larger models

As model sizes get larger, you will need to split your model across multiple GPUs for training

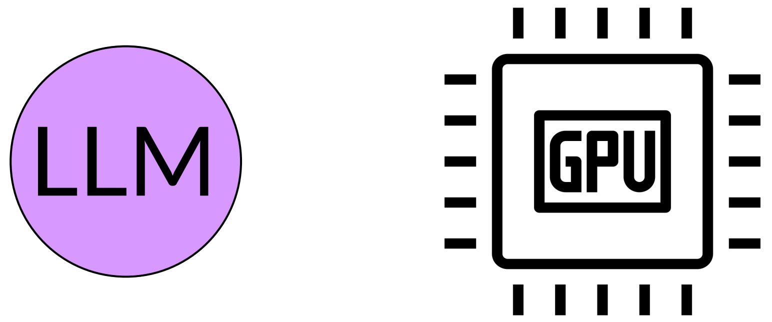

1B param model
14,000 GB @ 32-bit full precision
175B param model
500B param model
40,000 GB @ 32-bit full precision

Efficient Multi-GPU Compute Strategies

When to use distributed compute

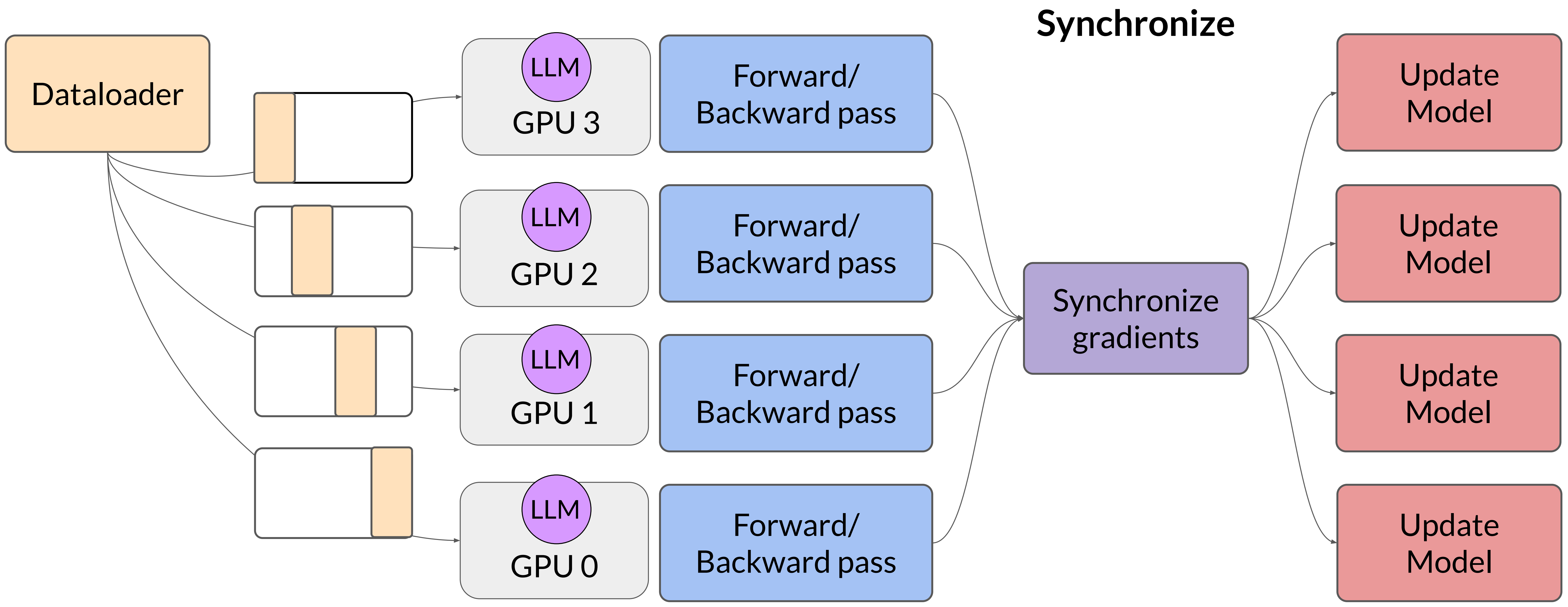


Model too big for single GPU



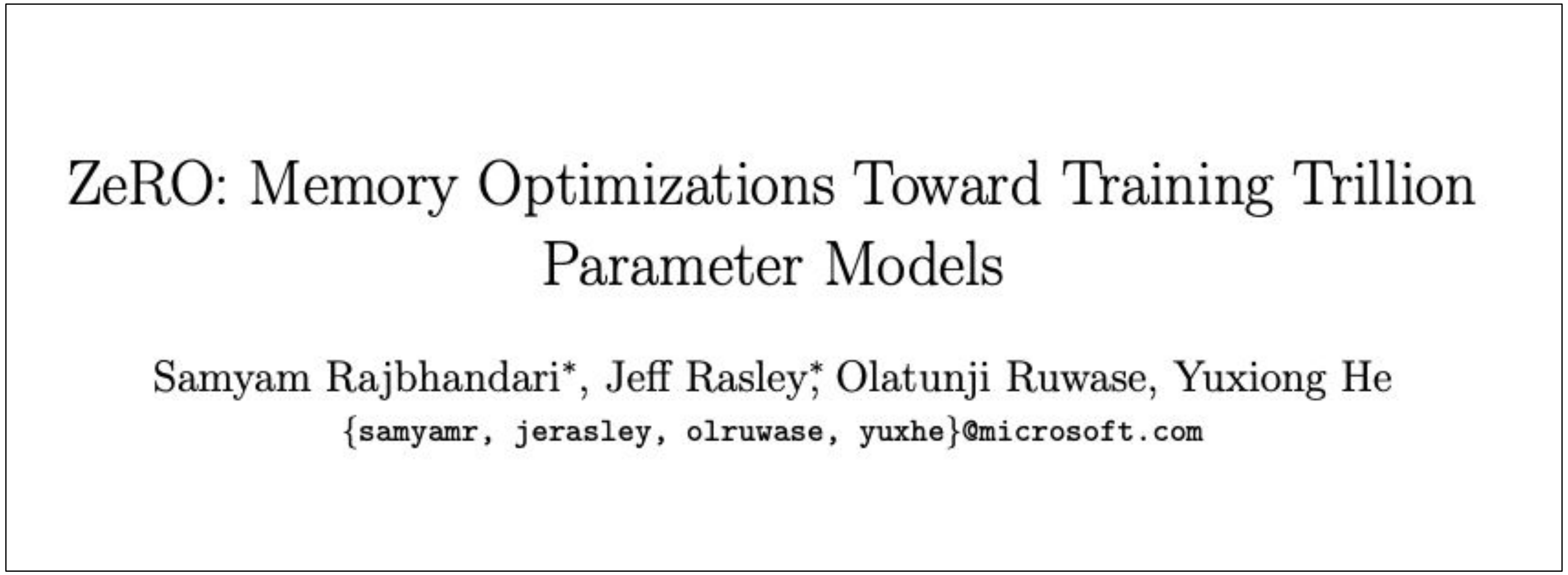
Model fits on GPU, train data in parallel

Distributed Data Parallel (DDP)



Fully Sharded Data Parallel (FSDP)

- Motivated by the “ZeRO” paper - zero data overlap between GPUs



ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Samyam Rajbhandari*, Jeff Rasley*, Olatunji Ruwase, Yuxiong He

{samyamr, jerasley, olruwase, yuxhe}@microsoft.com

Sources:
Rajbhandari et al. 2019: “ZeRO: Memory Optimizations Toward Training Trillion Parameter Models”
Zhao et al. 2023: “PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel”

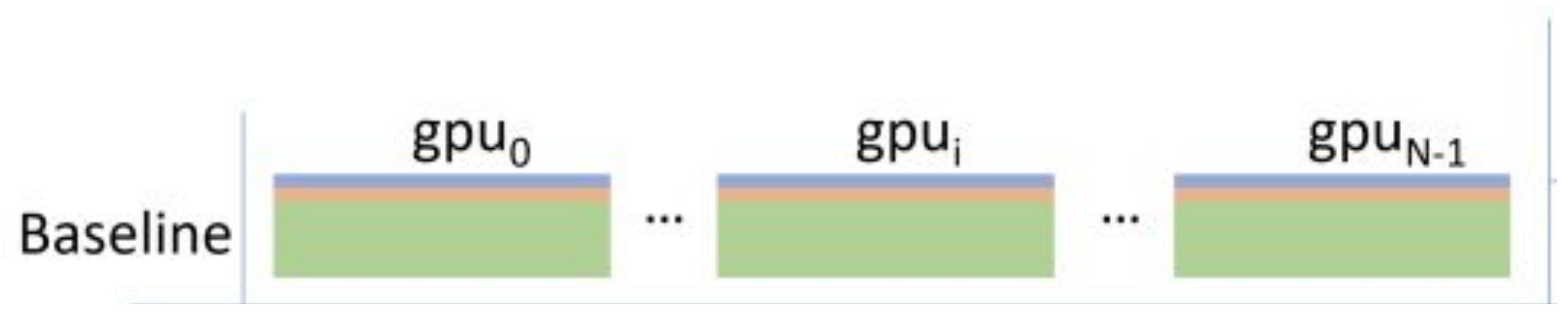
Recap: Additional GPU RAM needed for training

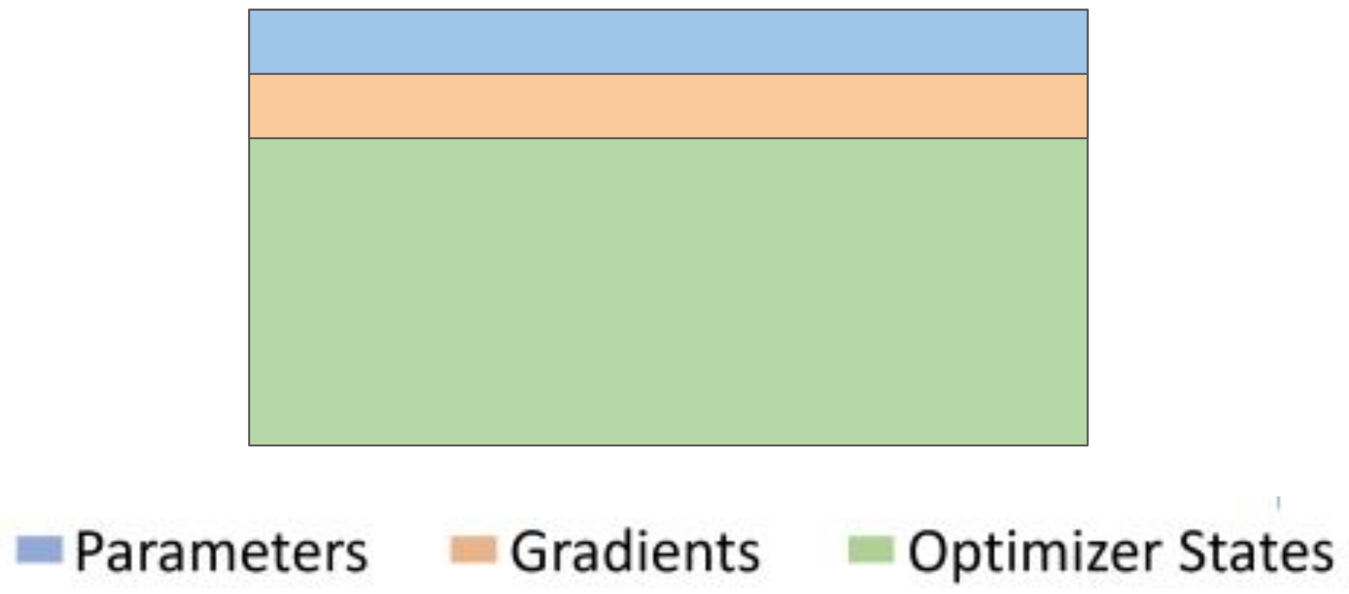
	Bytes per parameter
Model Parameters (Weights)	4 bytes per parameter
Adam optimizer (2 states)	+8 bytes per parameter
Gradients	+4 bytes per parameter
Activations and temp memory (variable size)	+8 bytes per parameter (high-end estimate)
TOTAL	**=4 bytes per parameter +20 extra bytes per parameter**

Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

Memory usage in DDP

- One full copy of model and training parameters on each GPU

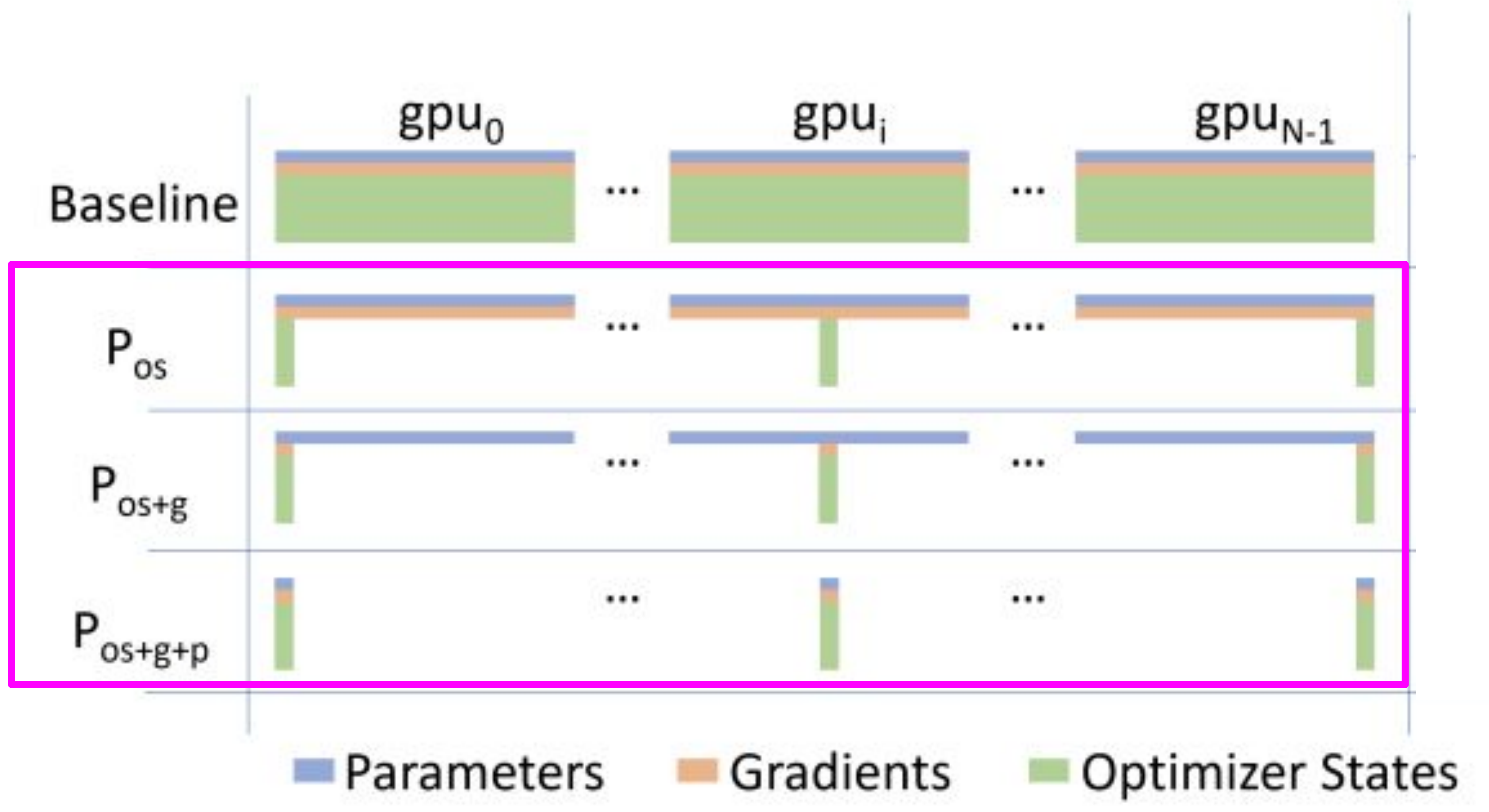


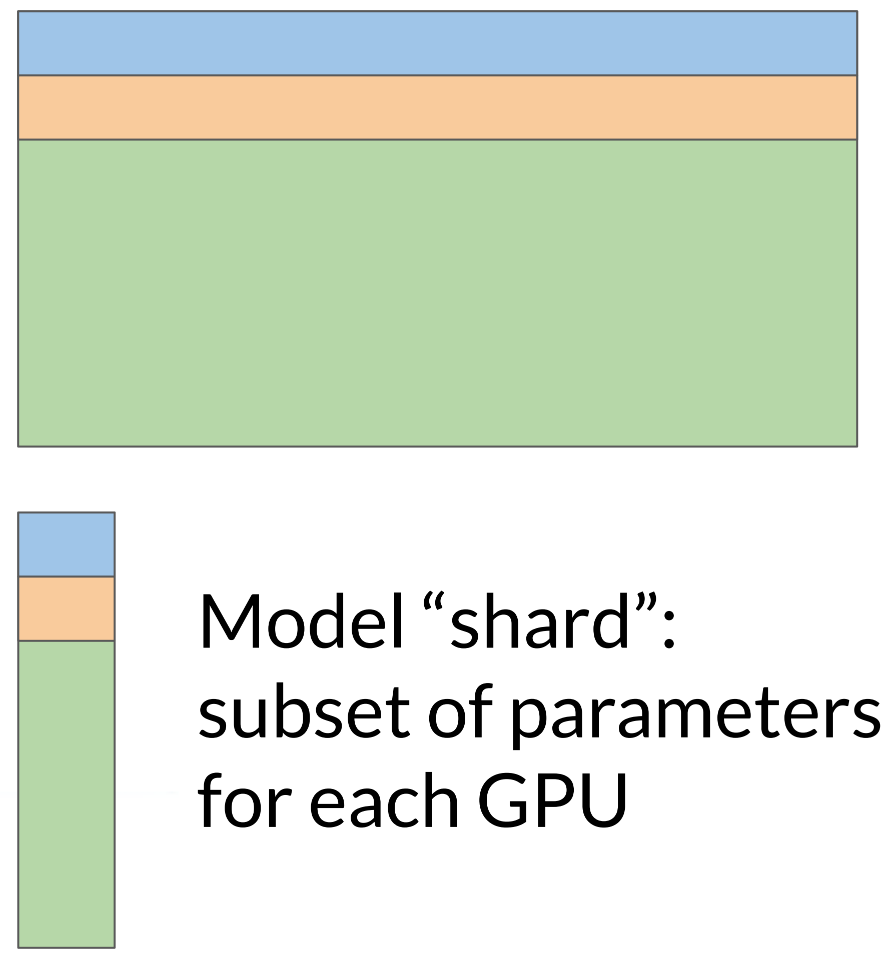


Sources:
Rajbhandari et al. 2019: “ZeRO: Memory Optimizations Toward Training Trillion Parameter Models”
Zhao et al. 2023: “PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel”

Zero Redundancy Optimizer (ZeRO)

- Reduces memory by distributing (sharding) the model parameters, gradients, and optimizer states across GPUs

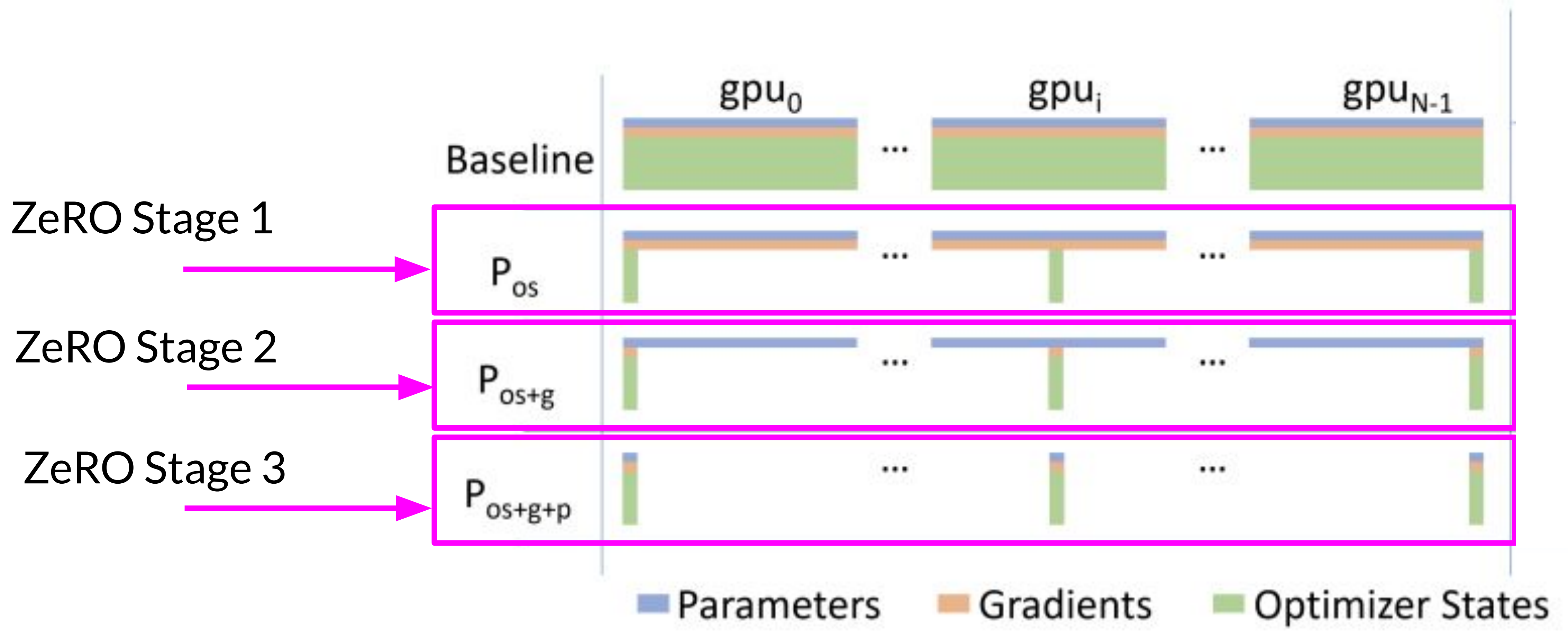




Sources:
Rajbhandari et al. 2019: "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models"
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

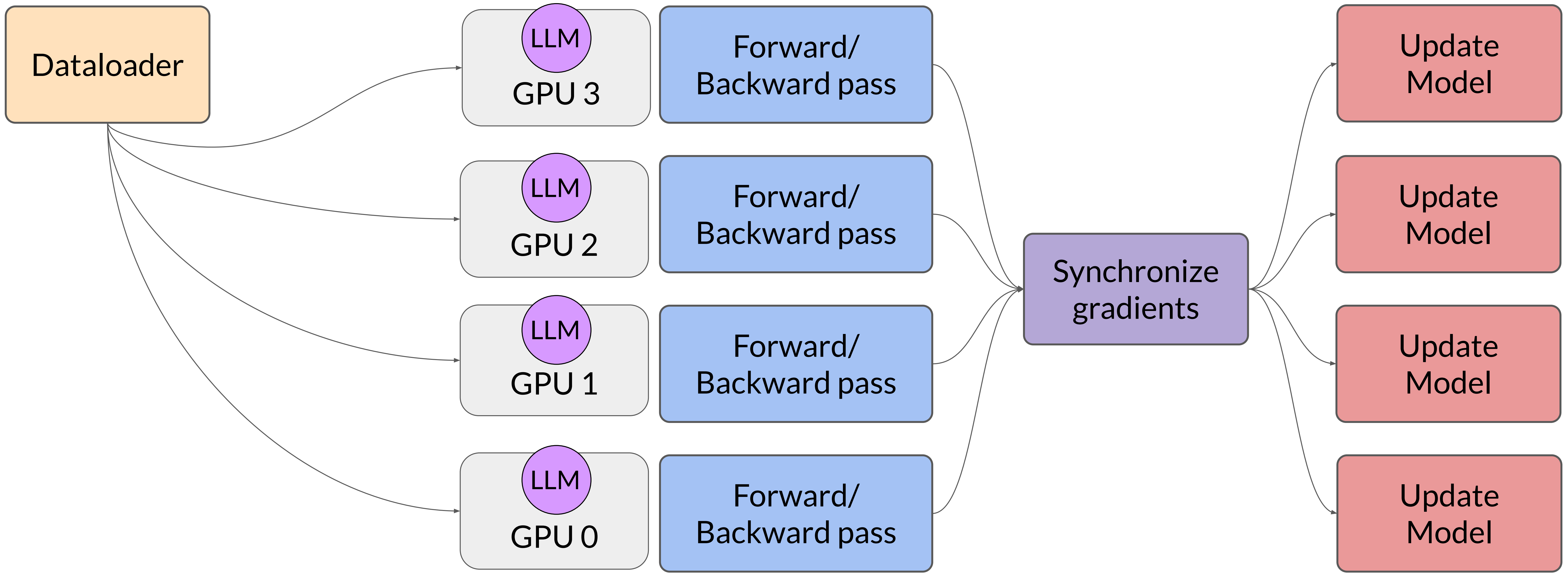
Zero Redundancy Optimizer (ZeRO)

- Reduces memory by distributing (sharding) the model parameters, gradients, and optimizer states across GPUs

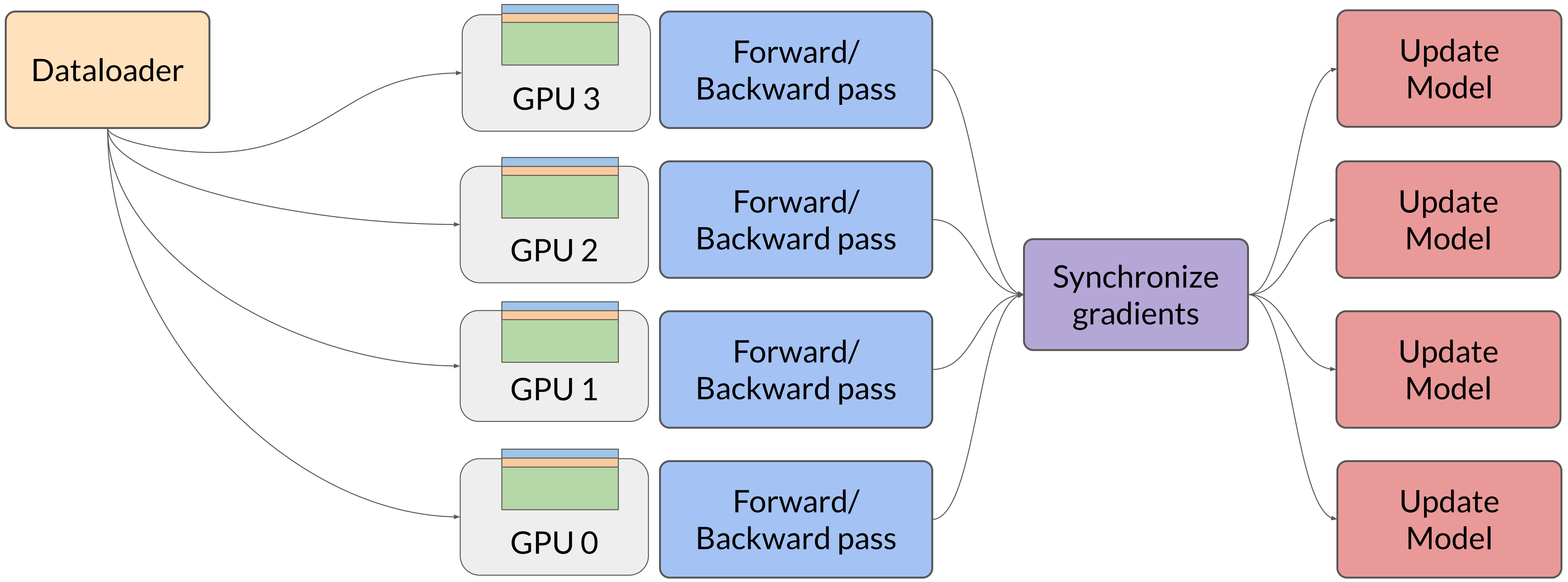


Sources:
Rajbhandari et al. 2019: “ZeRO: Memory Optimizations Toward Training Trillion Parameter Models”
Zhao et al. 2023: “PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel”

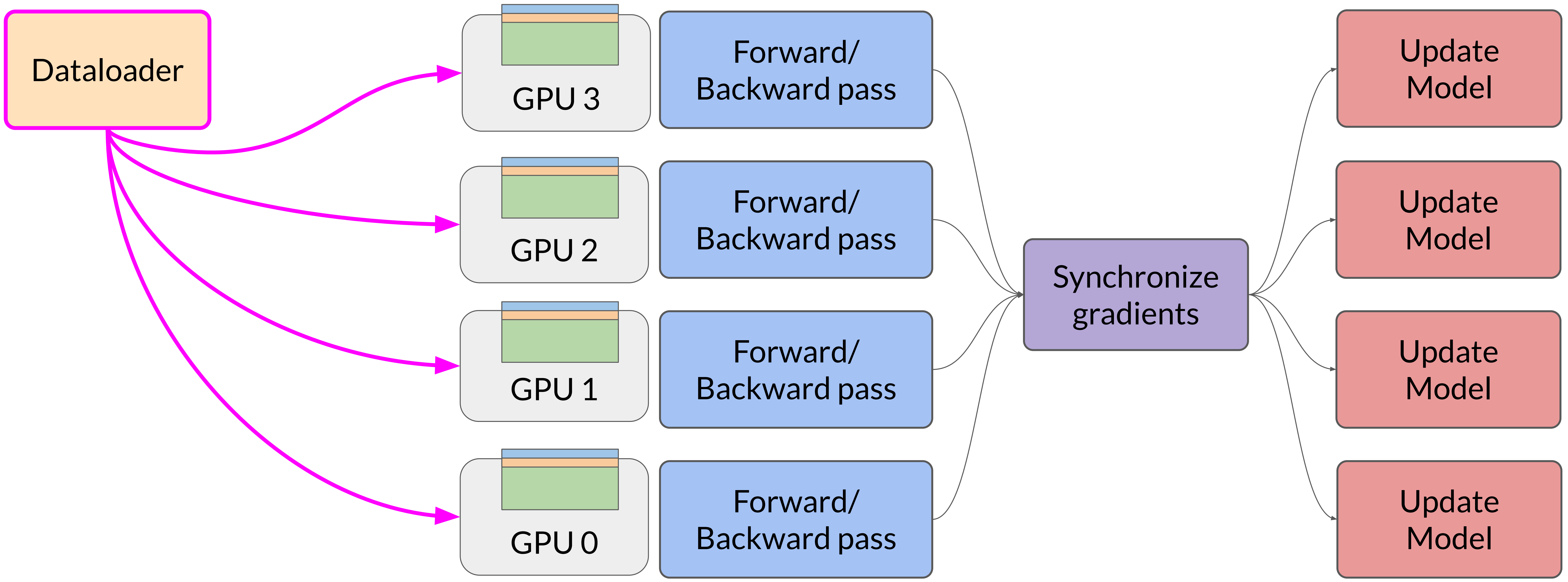
Distributed Data Parallel (DDP)



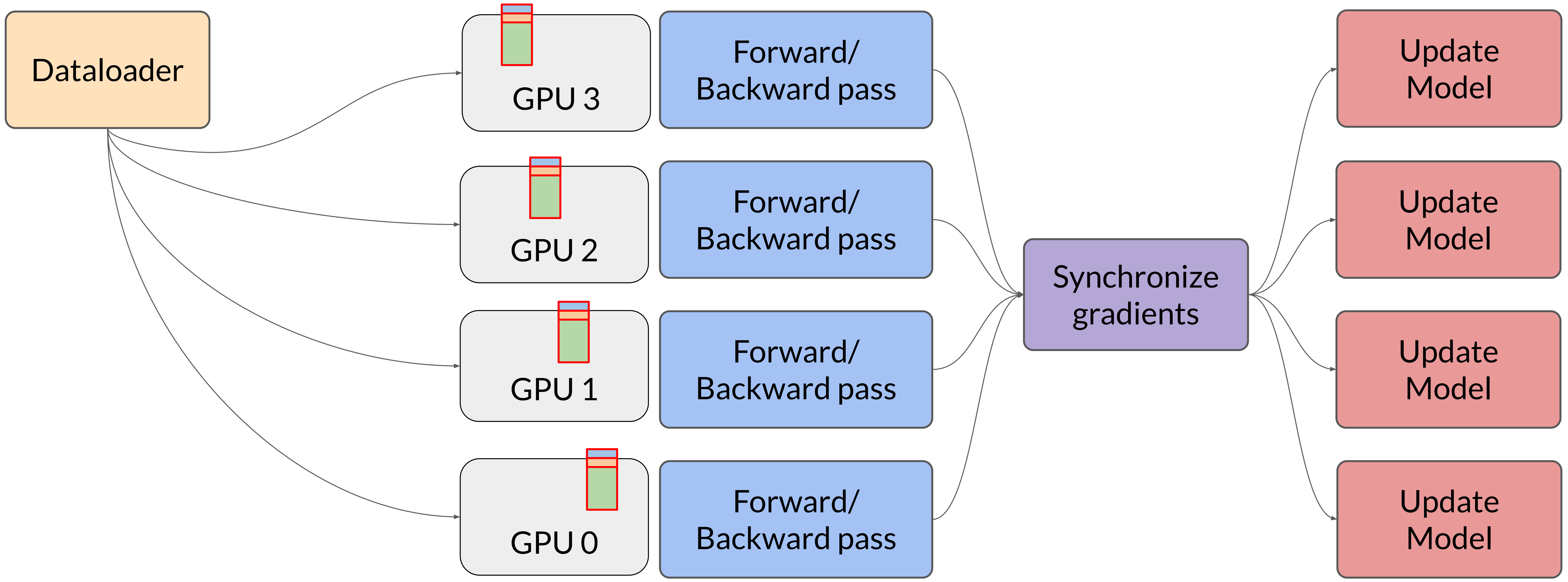
Distributed Data Parallel (DDP)



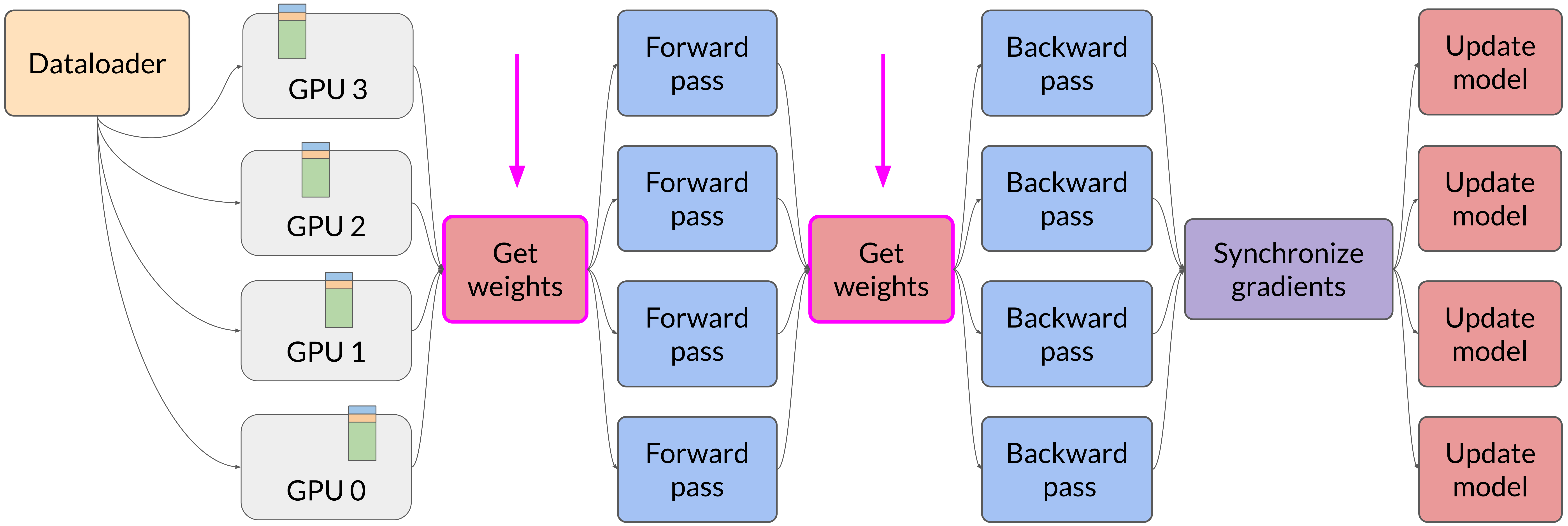
Fully Sharded Data Parallel (FSDP)



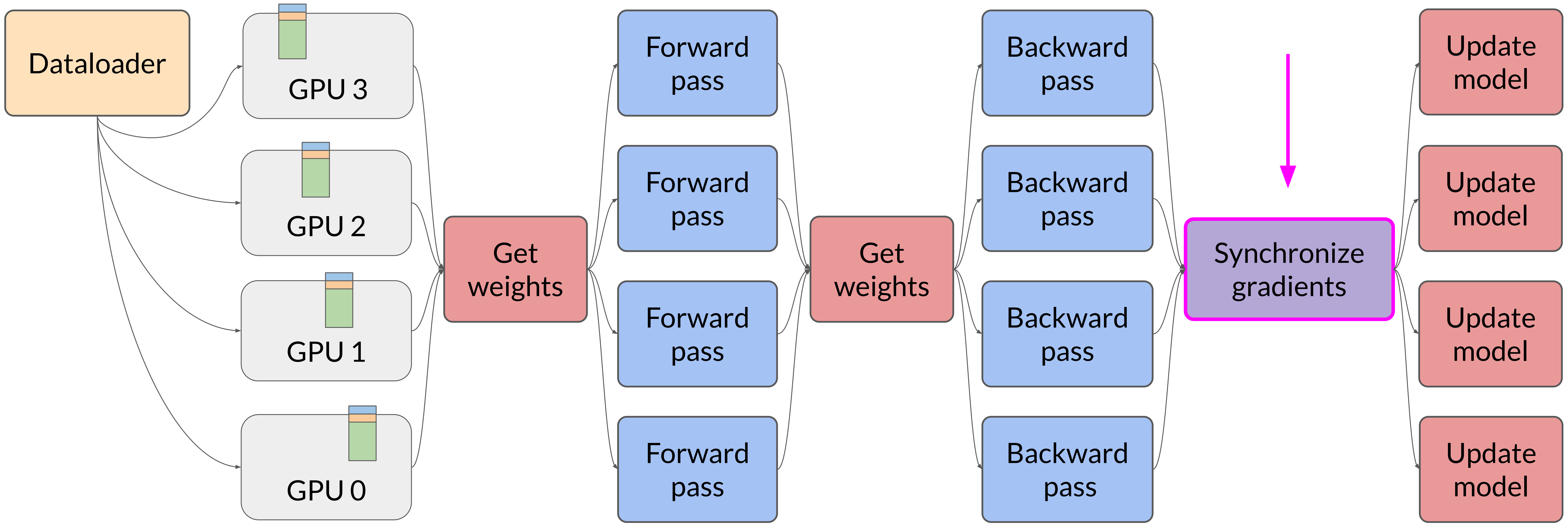
Fully Sharded Data Parallel (FSDP)



Fully Sharded Data Parallel (FSDP)



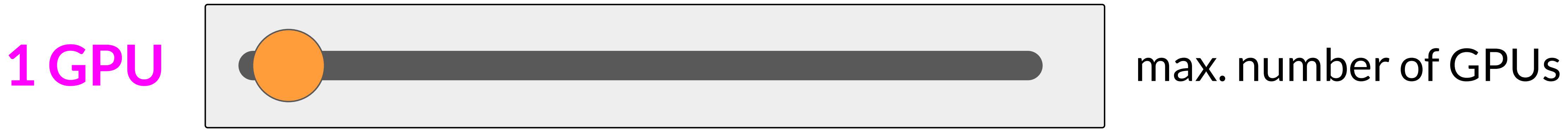
Fully Sharded Data Parallel (FSDP)



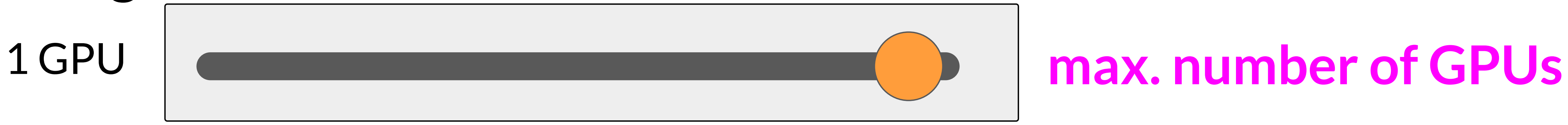
Fully Sharded Data Parallel (FSDP)

- Helps to reduce overall GPU memory utilization
- Supports offloading to CPU if needed
- Configure level of sharding via `sharding factor`

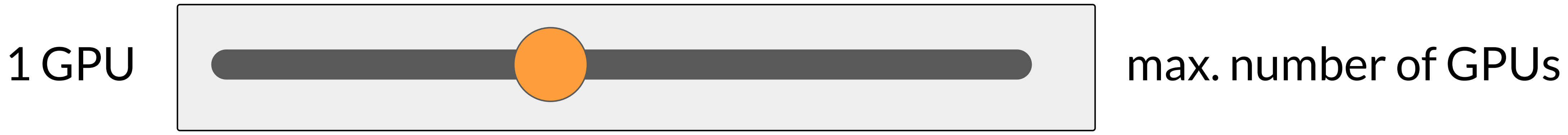
Full replication (no sharding)



Full sharding

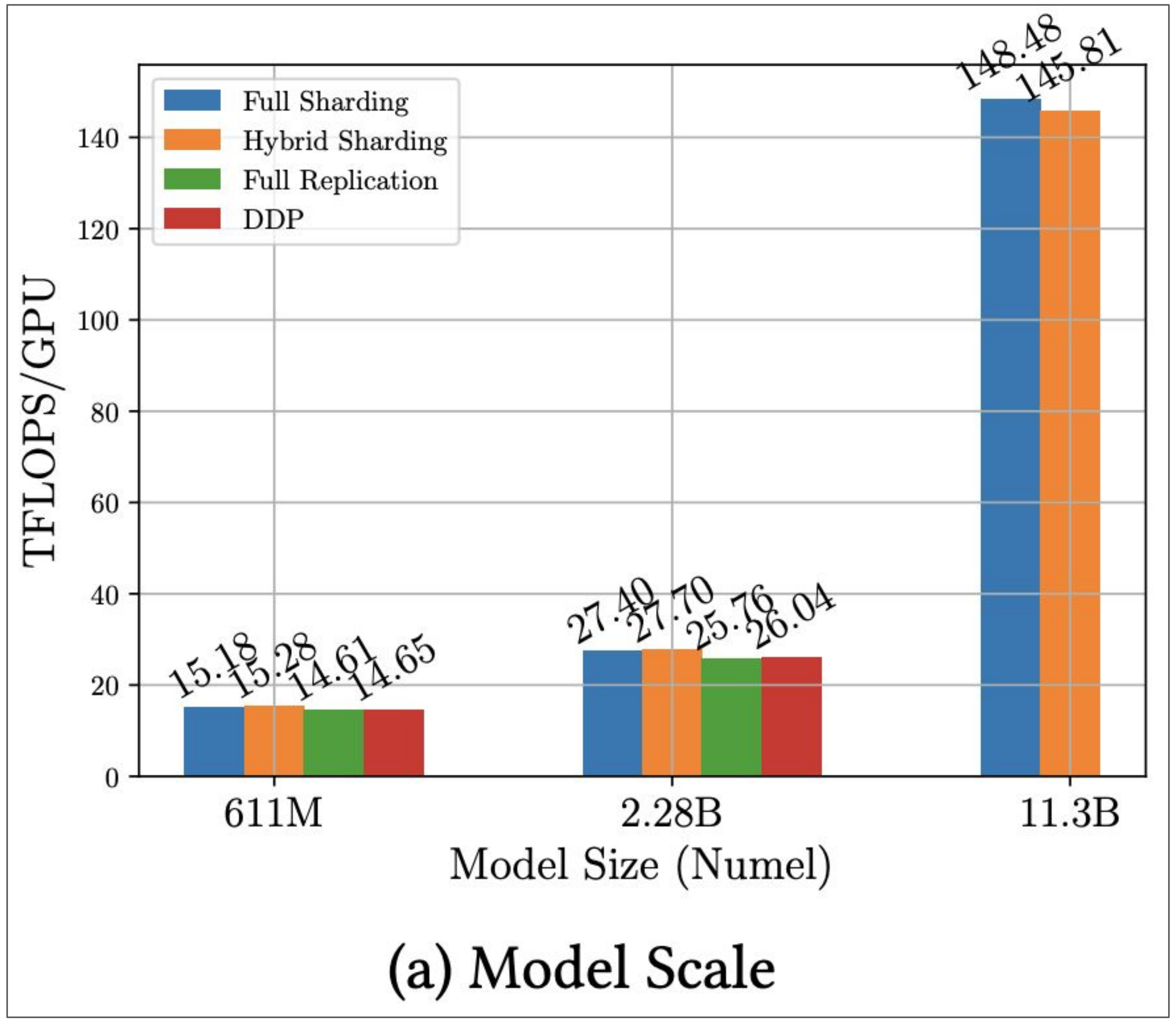


Hybrid sharding

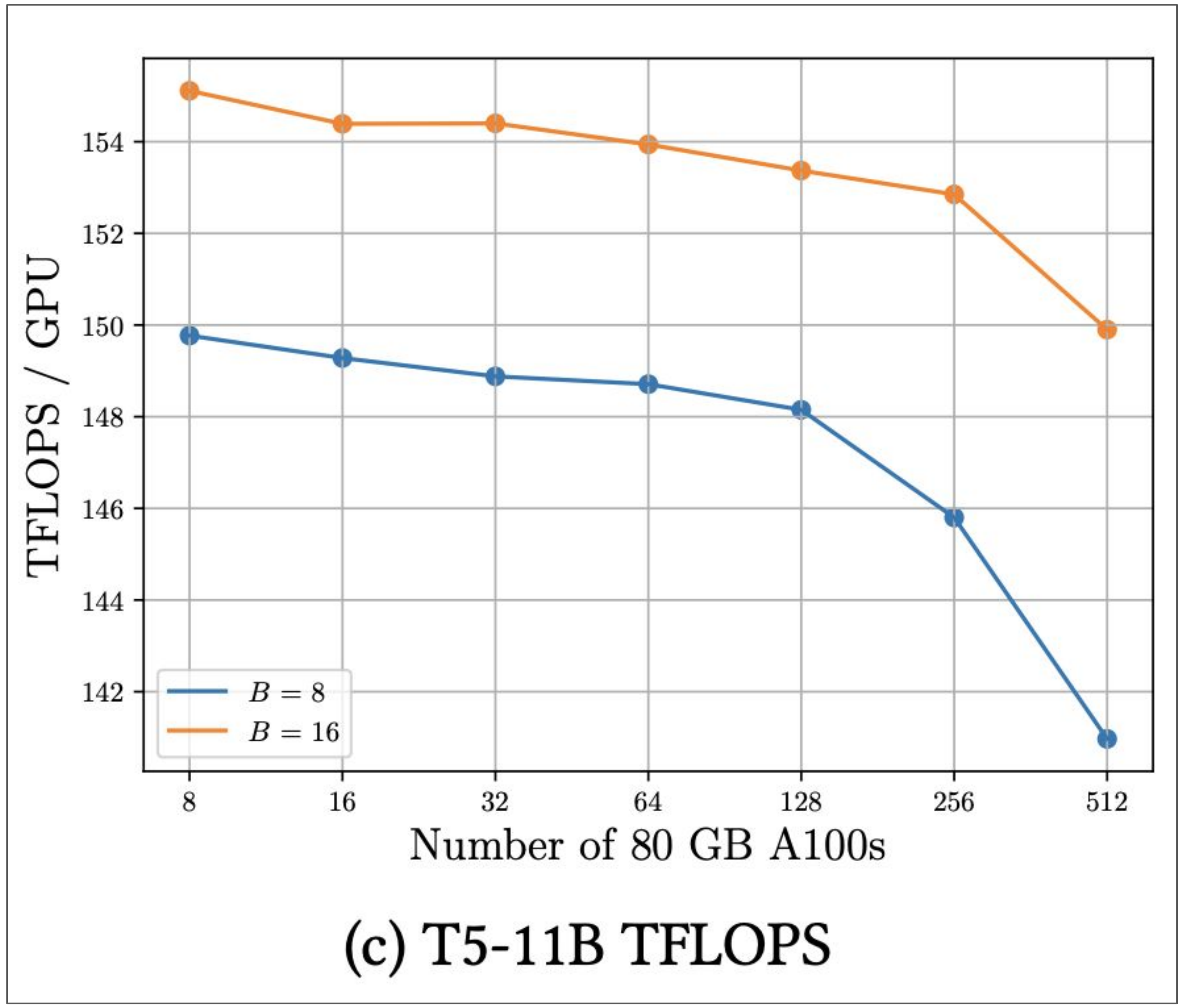


Impact of using FSDP

Note: 1 teraFLOP/s = 1,000,000,000,000 (one trillion) floating point operations per second



(a) Model Scale



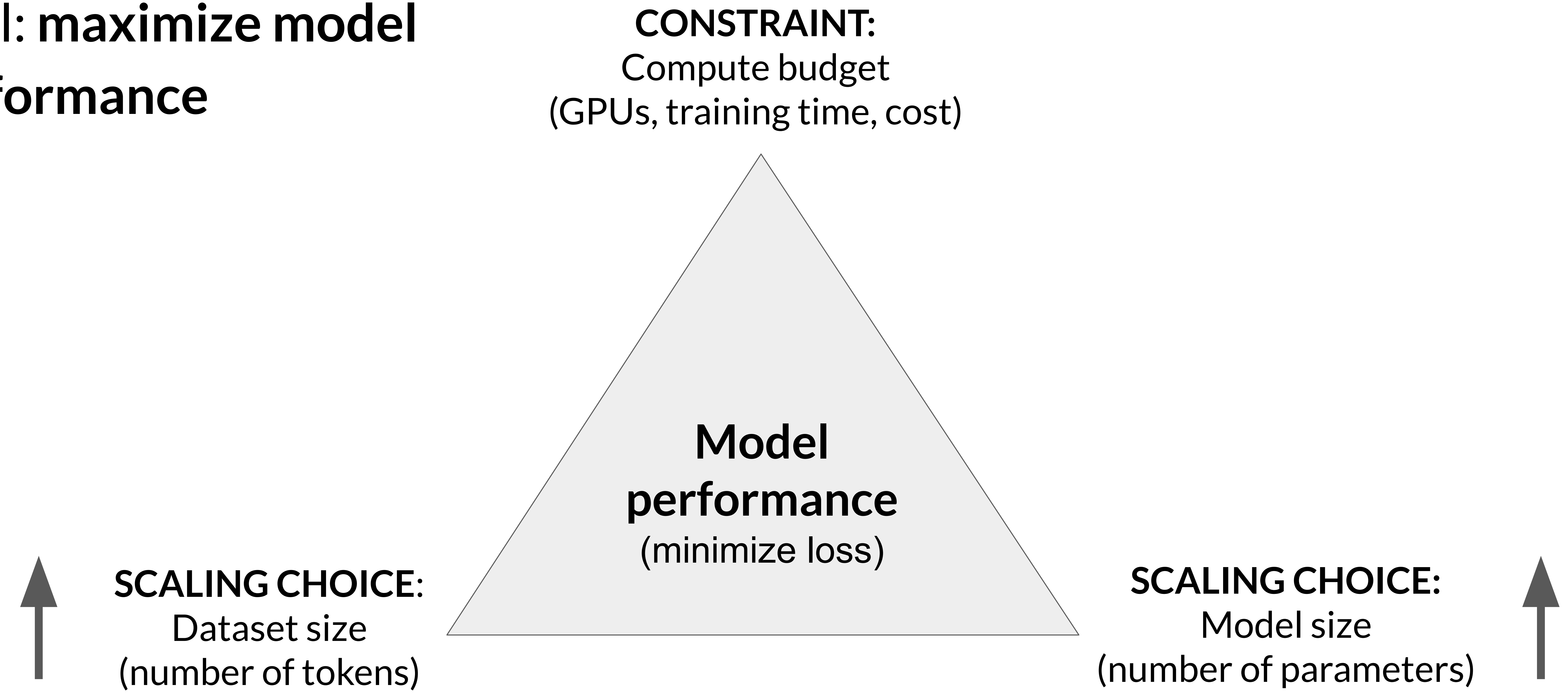
(c) T5-11B TFLOPS

Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

Scaling laws and compute-optimal models

Scaling choices for pre-training

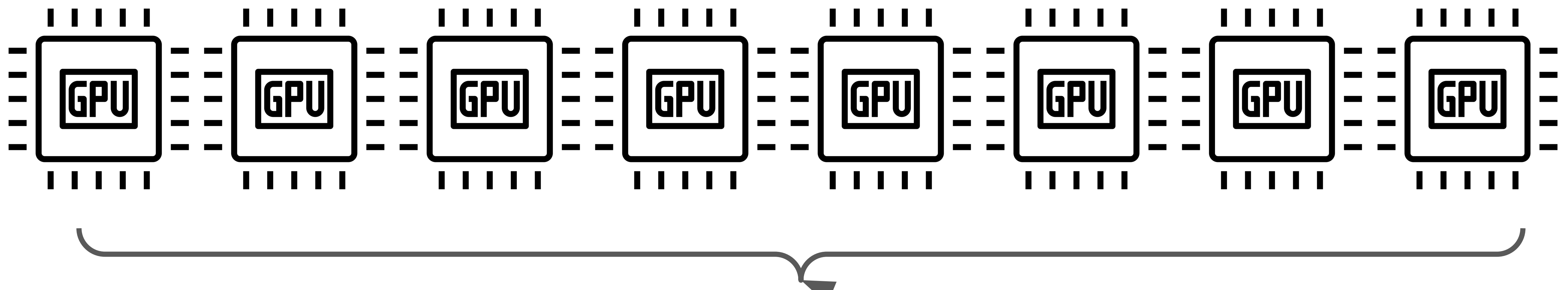
Goal: **maximize model performance**



Compute budget for training LLMs

1 “petaflop/s-day” =
# floating point operations performed at rate of 1 petaFLOP per second for one day

NVIDIA V100s

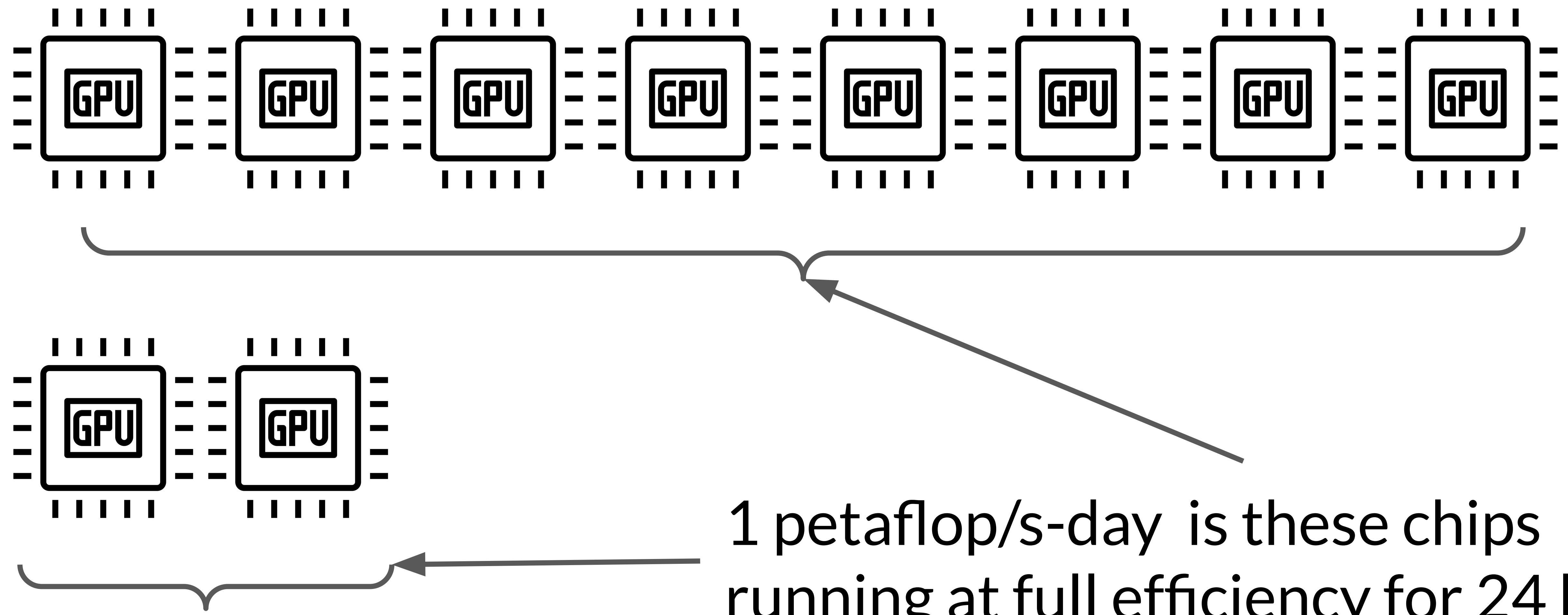


**Note: 1 petaFLOP/s = 1,000,000,000,000,000
(one quadrillion) floating point operations per second**

1 petaflop/s-day is these chips
running at full efficiency for 24 hours

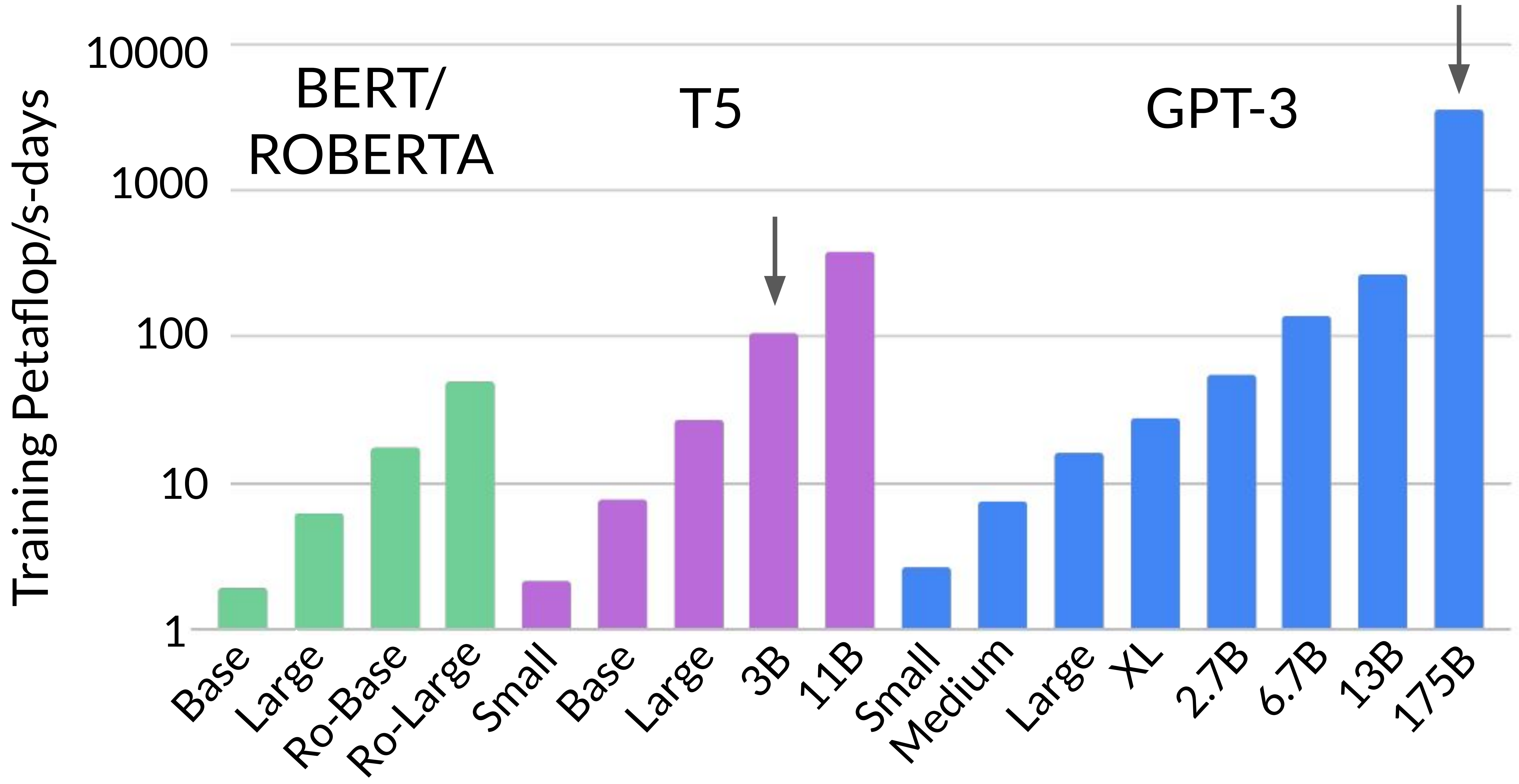
Compute budget for training LLMs

1 “petaflop/s-day” =
# floating point operations performed at rate of 1 petaFLOP per second for one day



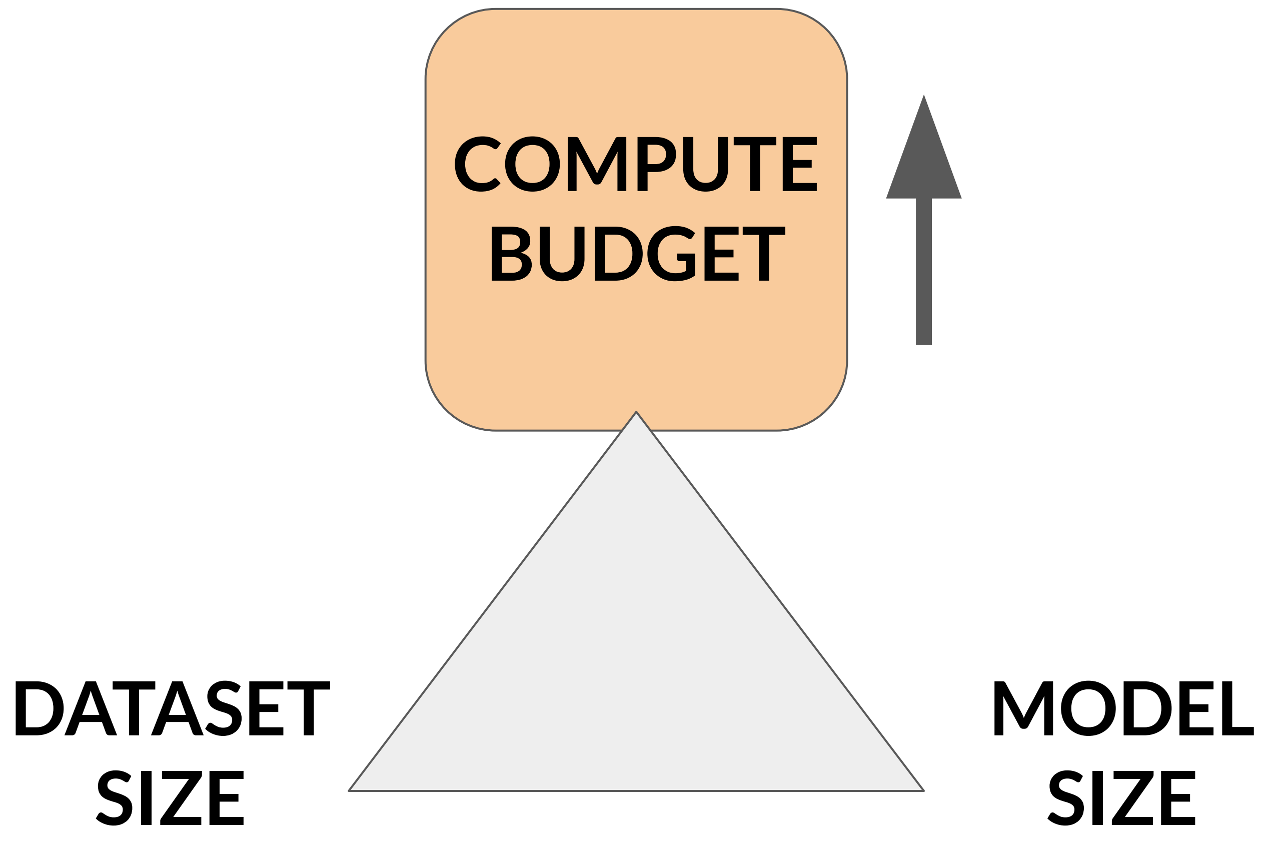
1 petaflop/s-day is these chips running at full efficiency for 24 hours

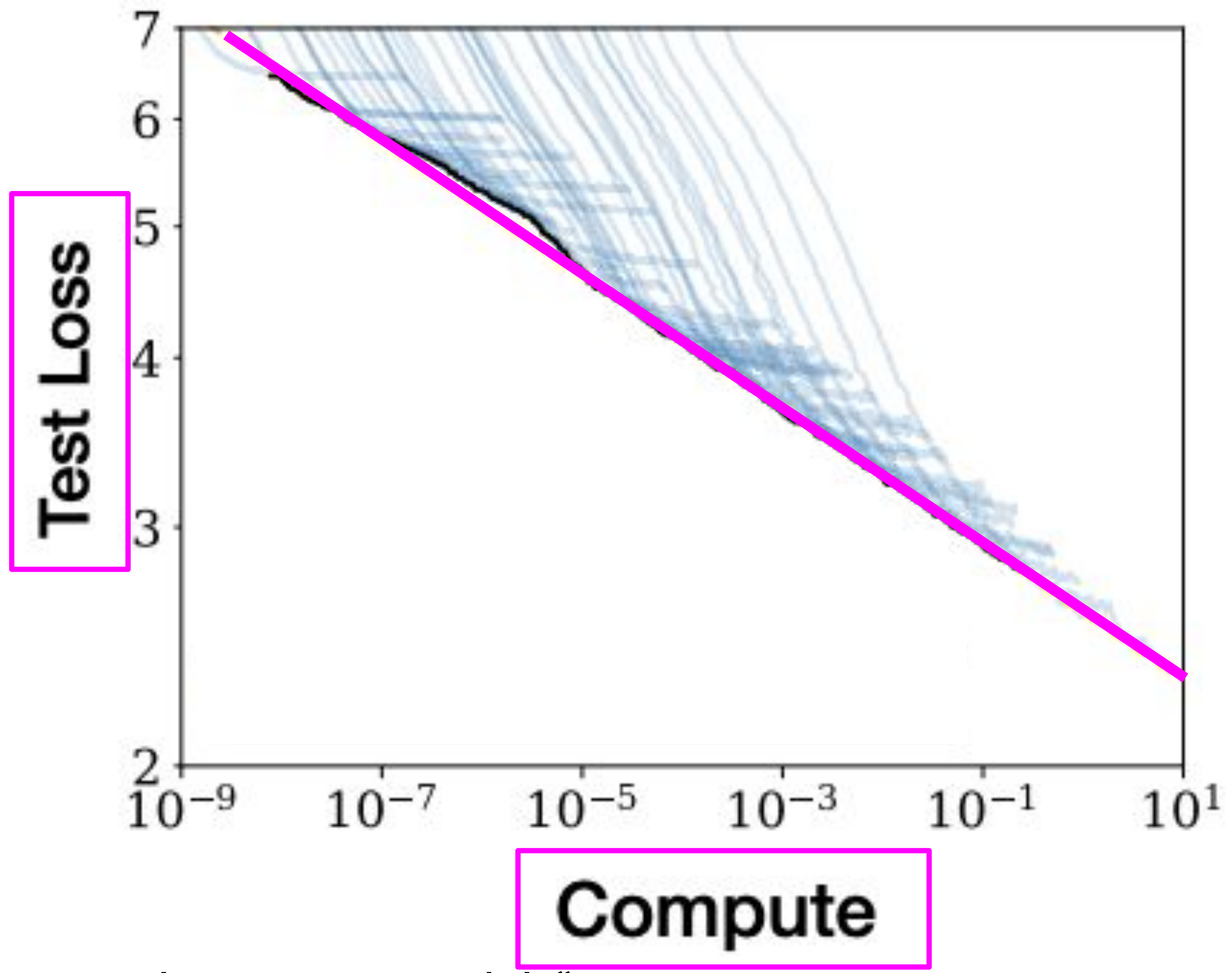
Number of petaflop/s-days to pre-train various LLMs



Source: Brown et al. 2020, "Language Models are Few-Shot Learners"

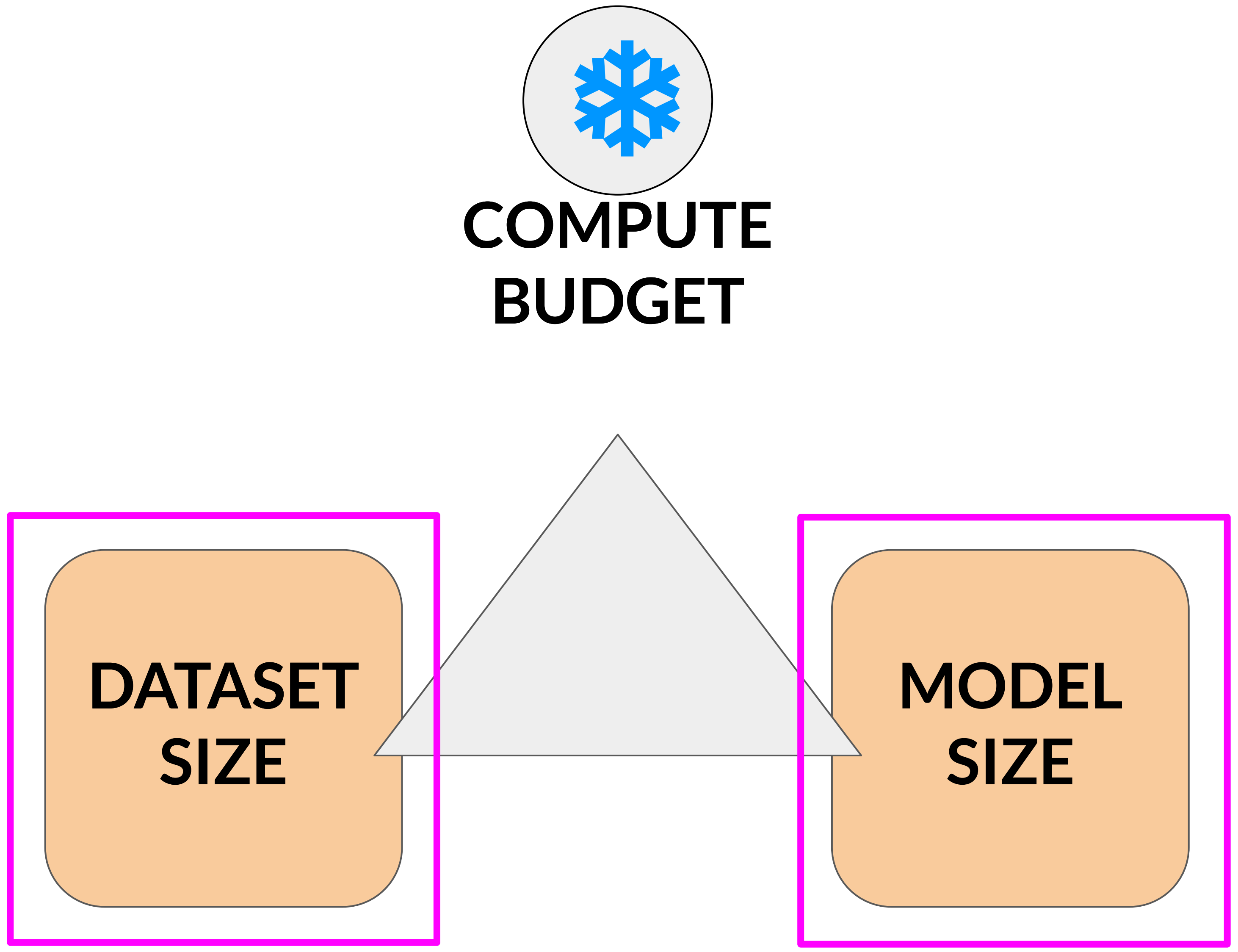
Compute budget vs. model performance





Source: Kaplan et al. 2020, “Scaling Laws for Neural Language Models”

Dataset size and model size vs. performance

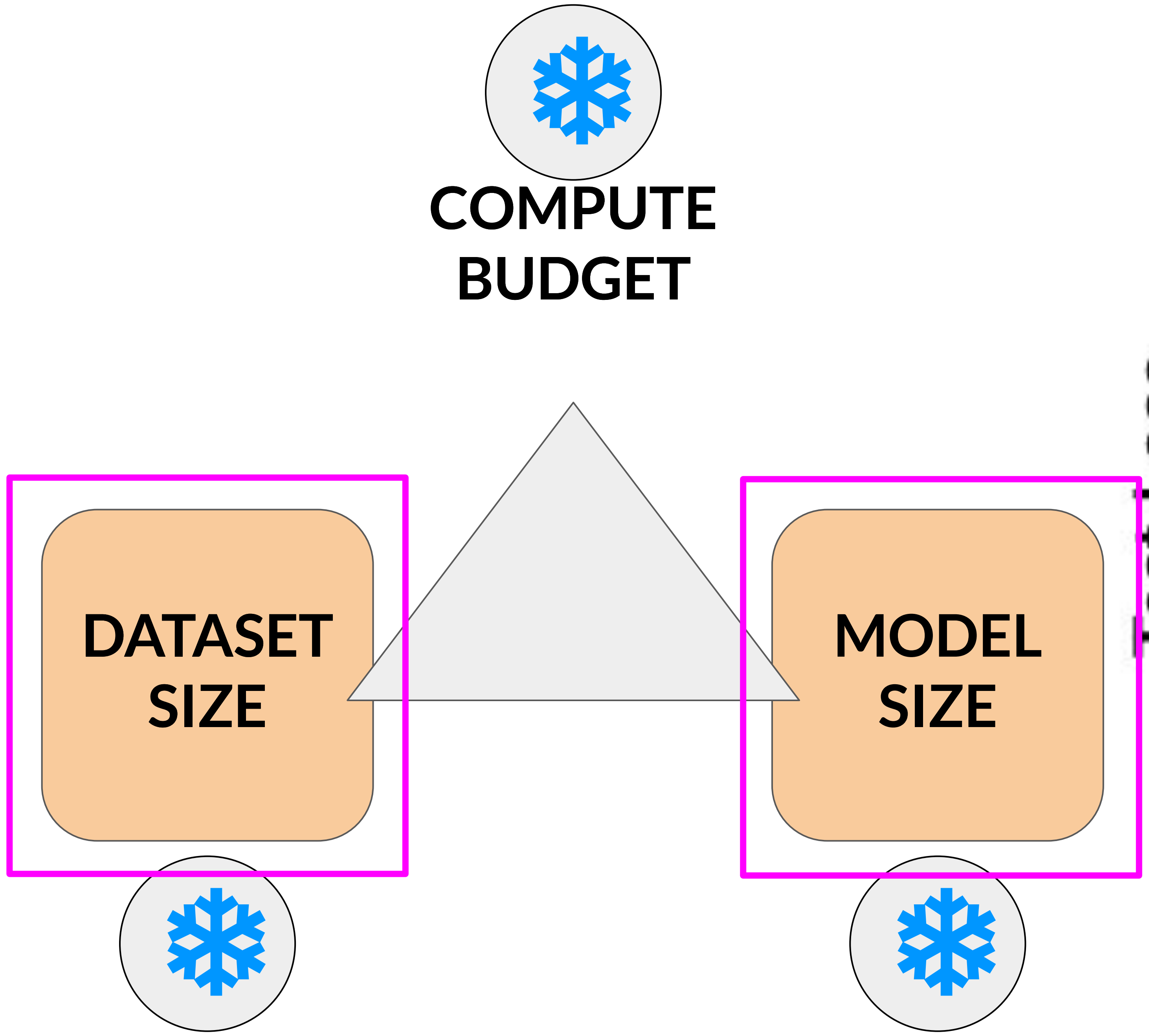


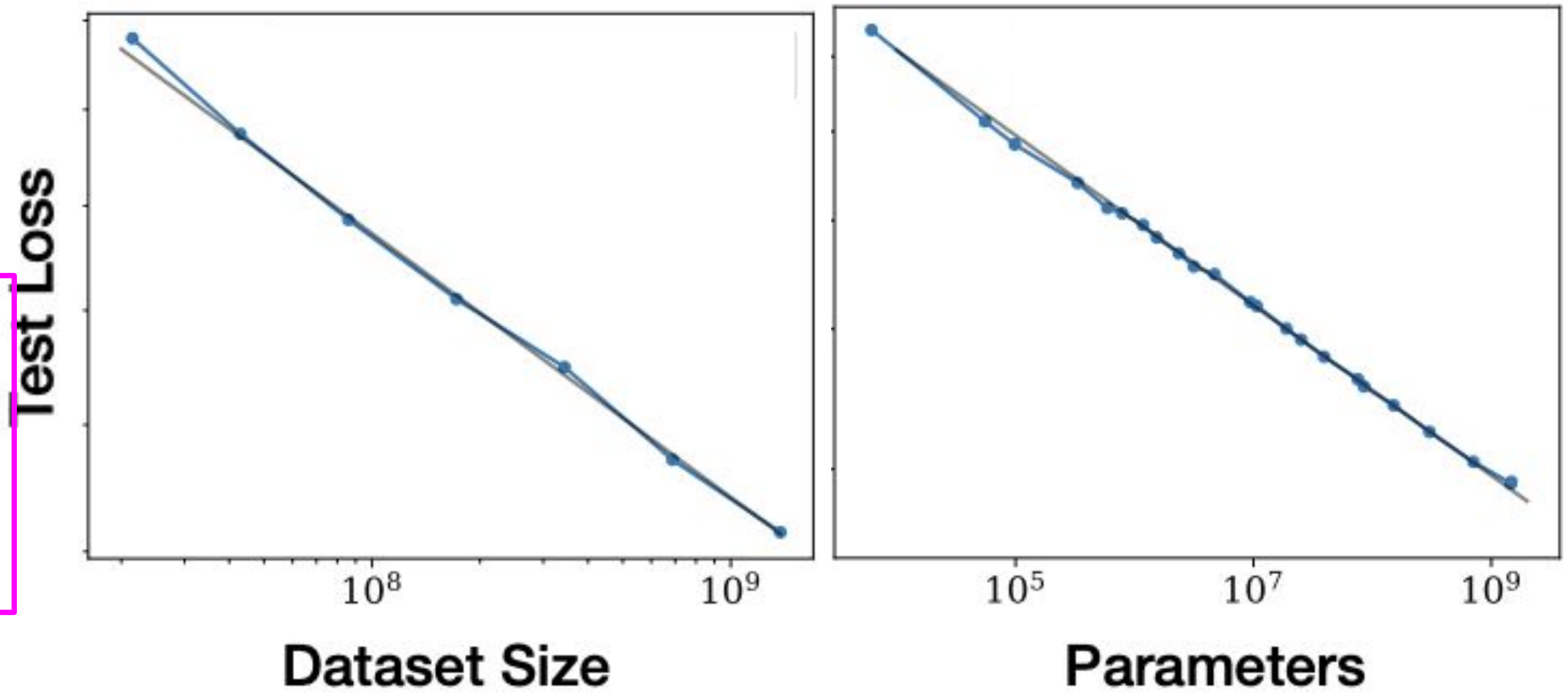
Compute resource constraints

- Hardware
- Project timeline
- Financial budget

Source: Kaplan et al. 2020, “Scaling Laws for Neural Language Models”

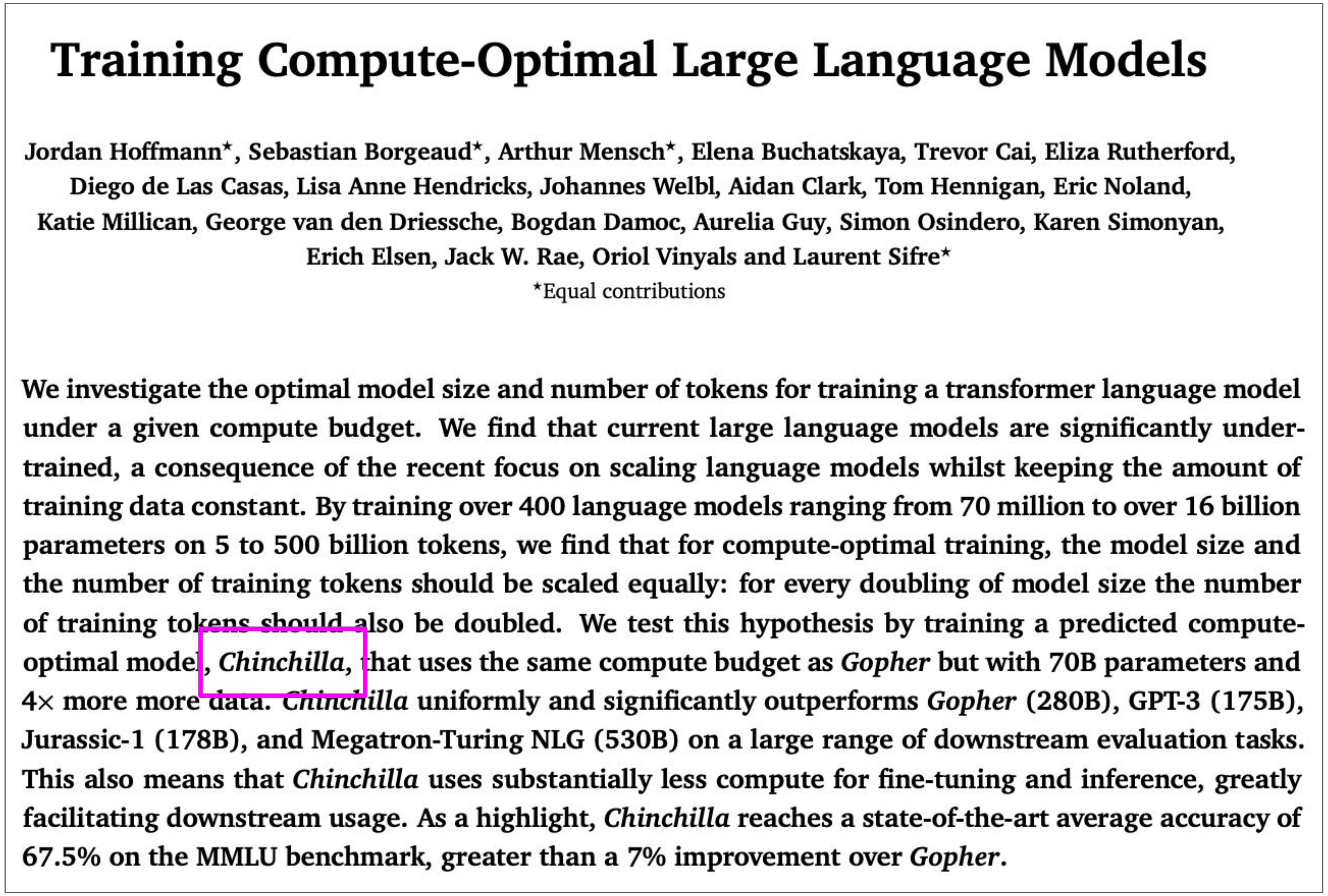
Dataset size and model size vs. performance





Source: Kaplan et al. 2020, "Scaling Laws for Neural Language Models"

Chinchilla paper



Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

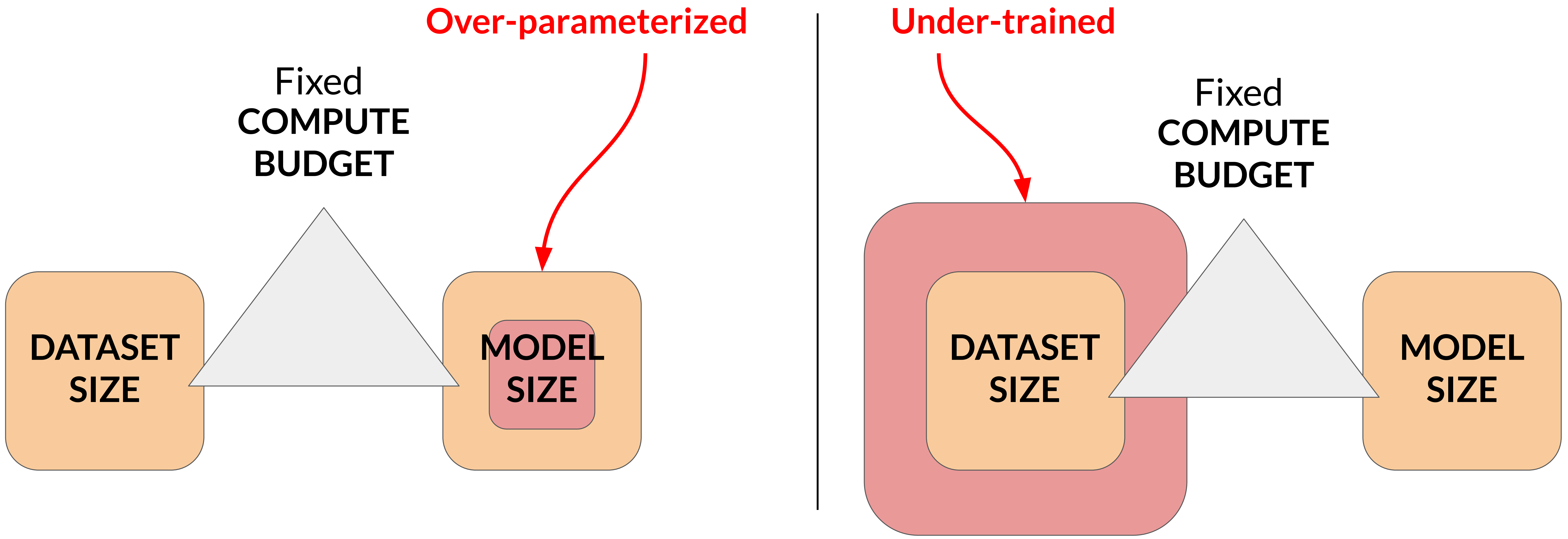
*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Jordan et al. 2022

Compute optimal models

- Very large models may be **over-parameterized** and **under-trained**
- Smaller models trained on more data could perform as well as large models



Chinchilla scaling laws for model and dataset size

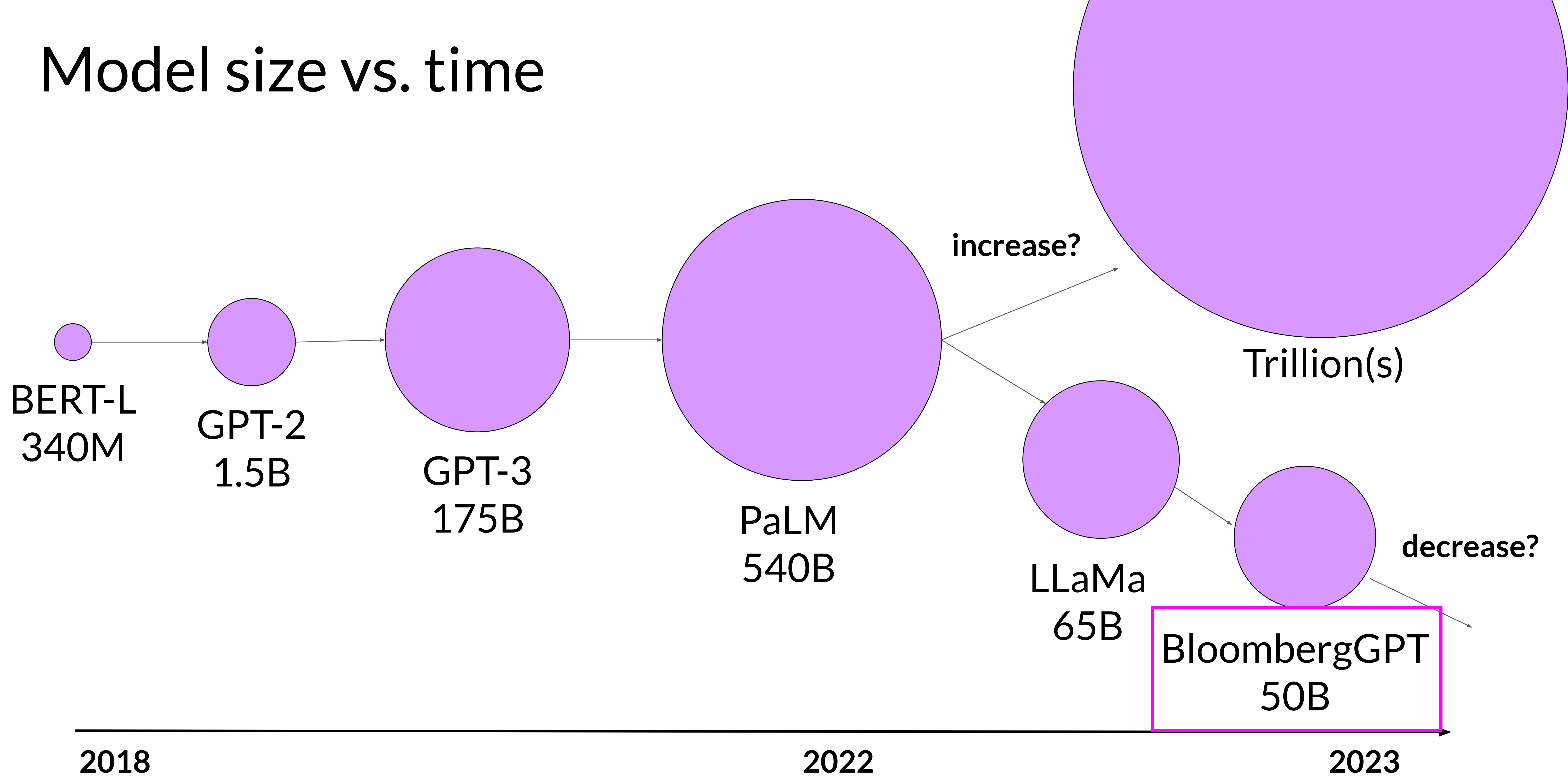
Model	# of parameters	Compute-optimal* # of tokens (~20x)	Actual # tokens
Chinchilla	70B	~1.4T	1.4T
LLaMA-65B	65B	~1.3T	1.4T
GPT-3	175B	~3.5T	300B
OPT-175B	175B	~3.5T	180B
BLOOM	176B	~3.5T	350B

Compute optimal training datasize
is ~**20x** number of parameters

* assuming models are trained to be compute-optimal per Chinchilla paper

Sources: Hoffmann et al. 2022, "Training Compute-Optimal Large Language Models"
Touvron et al. 2023, "LLaMA: Open and Efficient Foundation Language Models"

Model size vs. time



Pre-training for domain adaptation

Pre-training for domain adaptation

Legal language

Pre-training for domain adaptation

Legal language

```
The prosecutor had difficulty
proving mens rea, as the defendant
seemed unaware that his actions
were illegal.

The judge dismissed the case,
citing the principle of res
judicata as the issue had already
been decided in a previous trial.

Despite the signed agreement, the
contract was invalid as there was
no consideration exchanged between
the parties.
```

Pre-training for domain adaptation

Legal language

```
The prosecutor had difficulty
proving mens rea, as the defendant
seemed unaware that his actions
were illegal.

The judge dismissed the case,
citing the principle of res
judicata as the issue had already
been decided in a previous trial.

Despite the signed agreement, the
contract was invalid as there was
no consideration exchanged between
the parties.
```

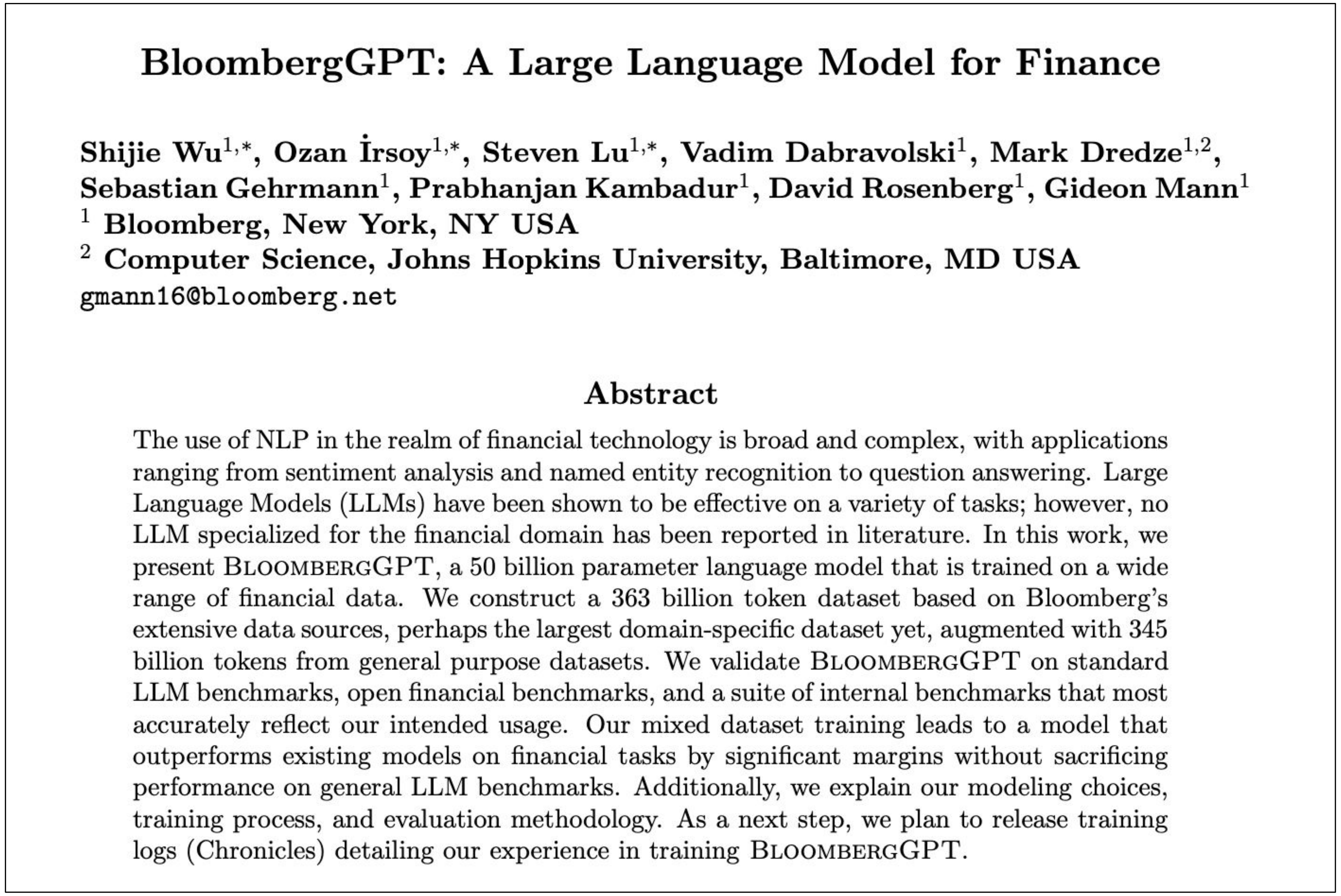
Medical language

```
After a strenuous workout, the
patient experienced severe myalgia
that lasted for several days.

After the biopsy, the doctor
confirmed that the tumor was
malignant and recommended immediate
treatment.

Sig: 1 tab po qid pc & hs
```

$\downarrow$

Take one tablet by mouth four times a day, after meals, and at bedtime.

BloombergGPT: domain adaptation for finance



BloombergGPT: A Large Language Model for Finance

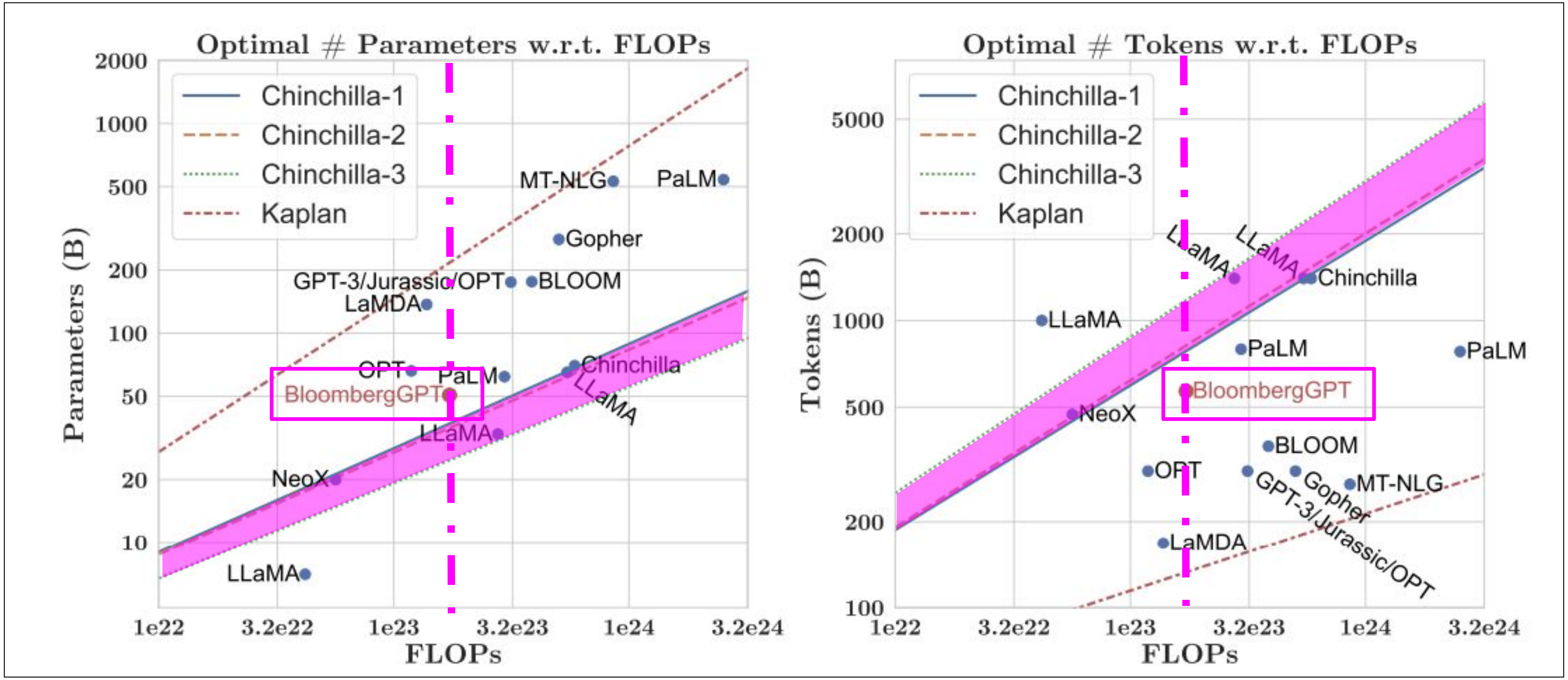
Shijie Wu[1,*], Ozan İrsoy[1,*], Steven Lu[1,*], Vadim Dabravolski[1], Mark Dredze[1,2], Sebastian Gehrmann[1], Prabhanjan Kambadur[1], David Rosenberg[1], Gideon Mann[1]
[1] **Bloomberg, New York, NY USA**
[2] **Computer Science, Johns Hopkins University, Baltimore, MD USA**
gmann16@bloomberg.net

Abstract

The use of NLP in the realm of financial technology is broad and complex, with applications ranging from sentiment analysis and named entity recognition to question answering. Large Language Models (LLMs) have been shown to be effective on a variety of tasks; however, no LLM specialized for the financial domain has been reported in literature. In this work, we present BloombergGPT, a 50 billion parameter language model that is trained on a wide range of financial data. We construct a 363 billion token dataset based on Bloomberg's extensive data sources, perhaps the largest domain-specific dataset yet, augmented with 345 billion tokens from general purpose datasets. We validate BloombergGPT on standard LLM benchmarks, open financial benchmarks, and a suite of internal benchmarks that most accurately reflect our intended usage. Our mixed dataset training leads to a model that outperforms existing models on financial tasks by significant margins without sacrificing performance on general LLM benchmarks. Additionally, we explain our modeling choices, training process, and evaluation methodology. As a next step, we plan to release training logs (Chronicles) detailing our experience in training BloombergGPT.

~51%	**Financial (Public & Private)**
~49%	**Other (Public)**

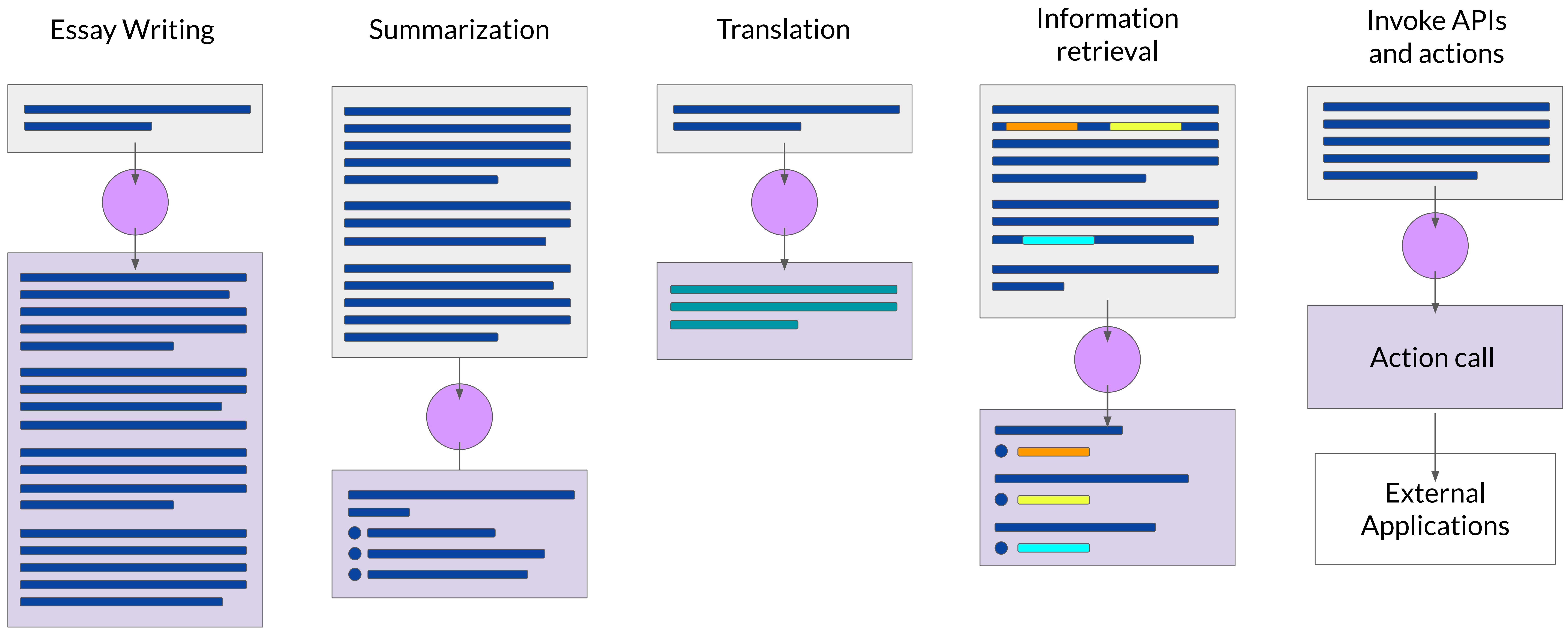
BloombergGPT relative to other LLMs



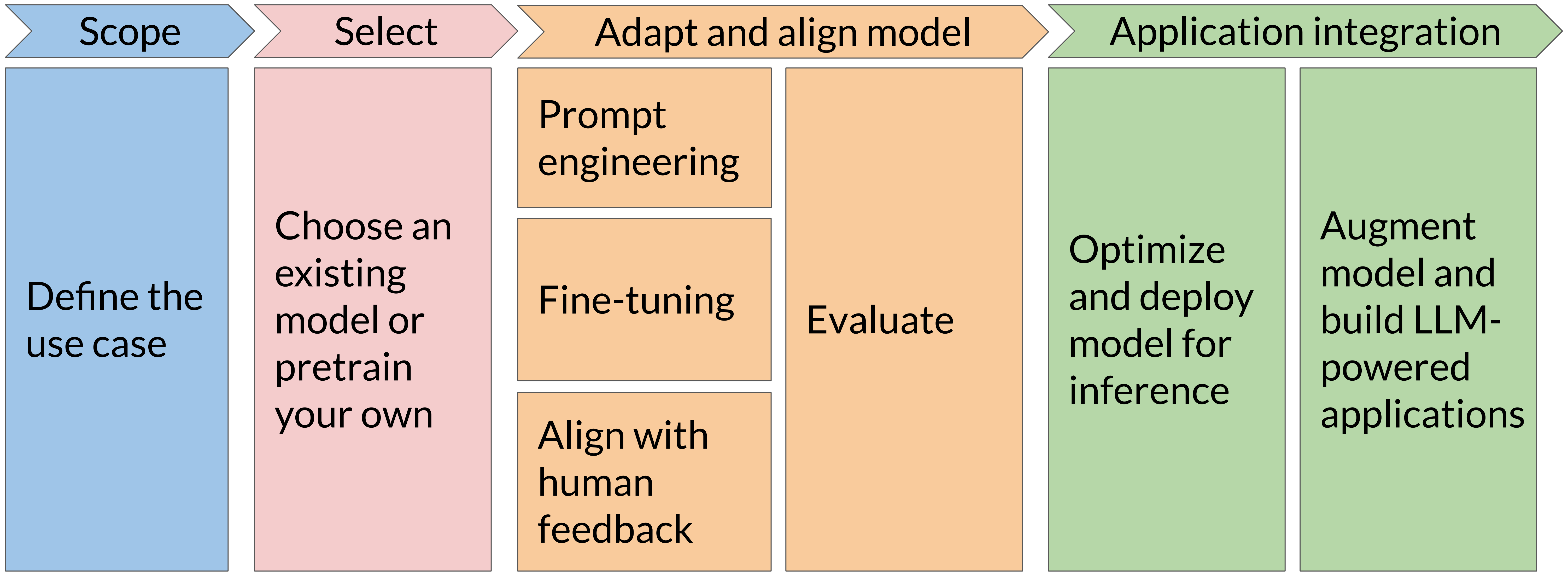
Source: Wu et al. 2023, "BloombergGPT: A Large Language Model for Finance"

Key takeaways

LLM use cases & tasks



Generative AI project lifecycle


Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications