



# Computational Language Technologies

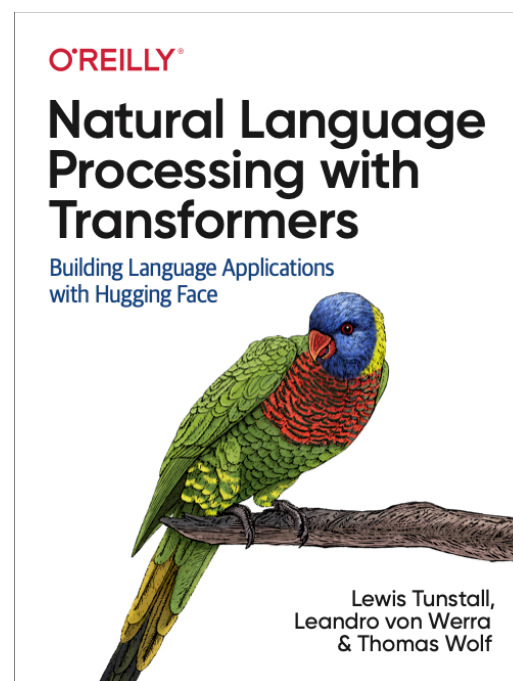
Lewis Tunstall | Open-source @ Hugging Face | [lewis@huggingface.co](mailto:lewis@huggingface.co)

Leandro von Werra | Open-source @ Hugging Face | [leandro@huggingface.co](mailto:leandro@huggingface.co)

# About us



[huggingface.co/  
course/](https://huggingface.co/course/)



[NLP with  
Transformers](#)

Education

The screenshot shows the GitHub profile for 'Hugging Face'. The profile header includes the Hugging Face emoji logo, the name 'Hugging Face', the bio 'Solving NLP, one commit at a time!', location 'NYC + Paris', website 'https://huggingface.co/', and a 'Verified' badge. Below the header is a navigation bar with links to Repositories (203), Packages, People (80), Teams (5), Projects (4), Sponsoring (4), and Settings. The 'Pinned repositories' section displays six repositories in a grid:

Repository Name	Description	Language	Stars	Forks
transformers	Transformers: State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0.	Python	44.9k	10.7k
datasets	The largest hub of ready-to-use NLP datasets for ML models with fast, easy-to-use and efficient data manipulation tools	Python	7.2k	860
tokenizers	Fast State-of-the-Art Tokenizers optimized for Research and Production	Rust	4.5k	336
awesome-papers	Papers & presentation materials from Hugging Face's internal science day	-	1.8k	104
accelerate	A simple way to train and use PyTorch models with multi-GPU, TPU, mixed-precision	Python	506	14
huggingface_hub	Client library to download and publish models and other files on the huggingface.co hub	Python	65	7

Open Source



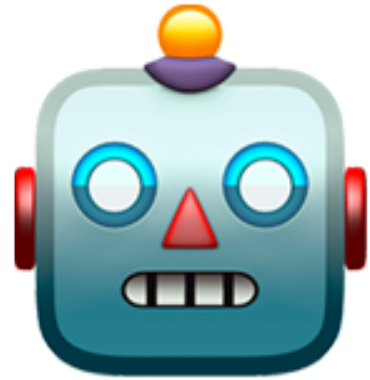
# Workshop materials

**Notebooks:** <https://github.com/huggingface/workshops/>

 **Account:** <https://huggingface.co/join>



# Plan of attack



Transformers



Hugging Face



Applications



# Plan of attack

- Part 1: Transformers & 🤗 ecosystem
- Part 2: NLP applications and beyond
- Part 3: Hands-on example: Training and deployment



# Transformers in the wild 🤠

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses
```

Code generation

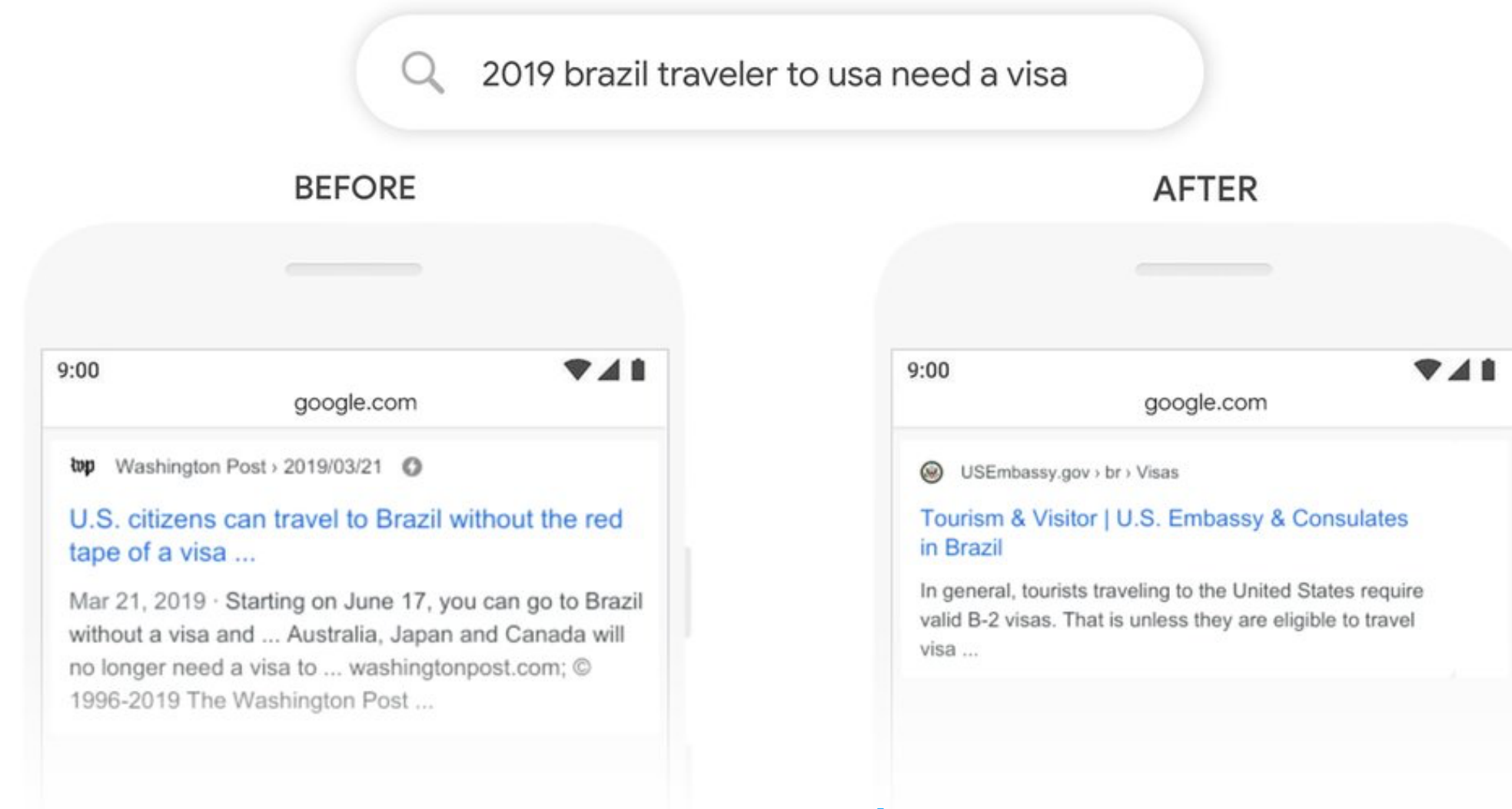




# Transformers in the wild 🤠

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses
```

Code generation



Search



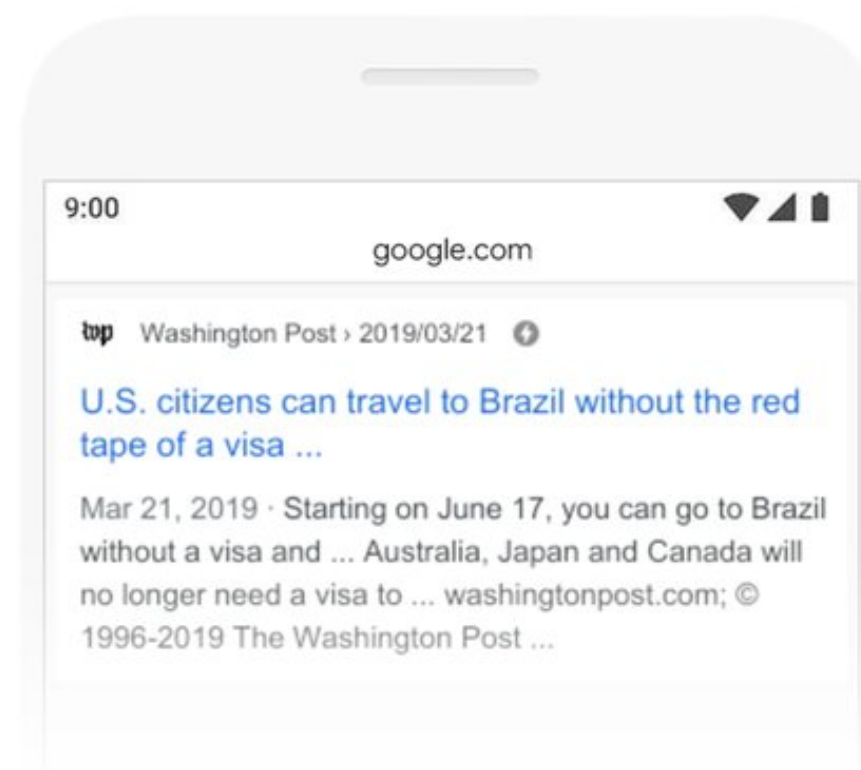
# Transformers in the wild 🤠

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses
```

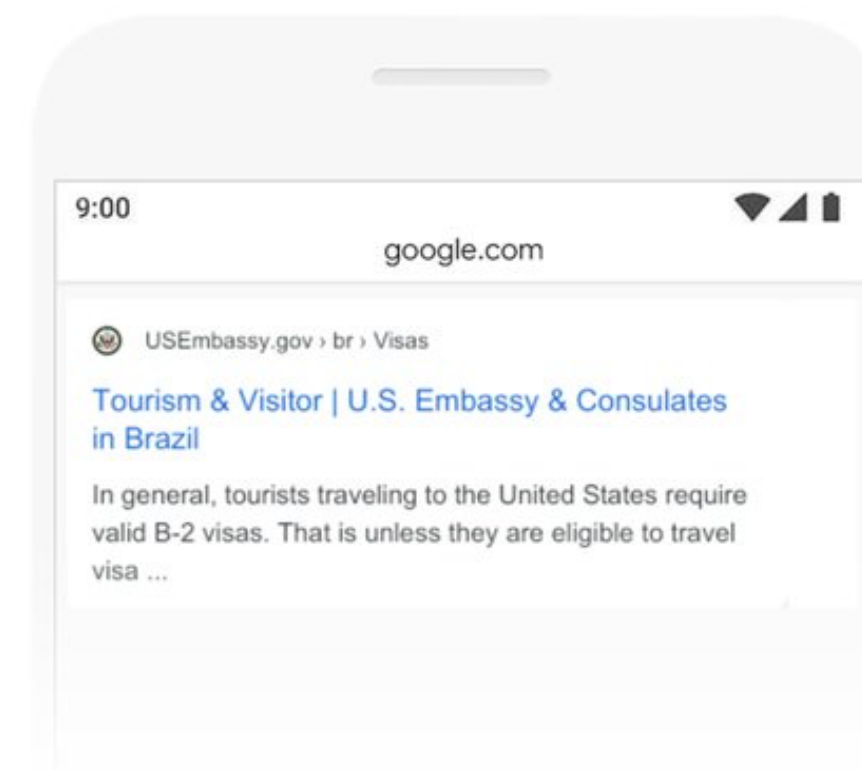
## Code generation

🔍 2019 brazil traveler to usa need a visa

BEFORE



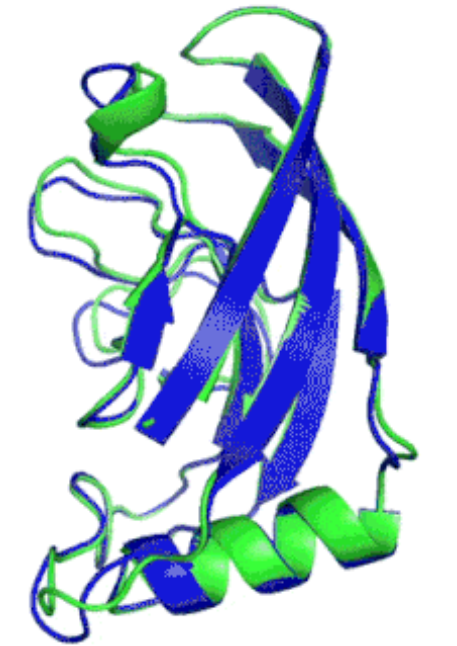
AFTER



## Search



T1037 / 6vr4  
90.7 GDT  
(RNA polymerase domain)



T1049 / 6y4f  
93.3 GDT  
(adhesin tip)

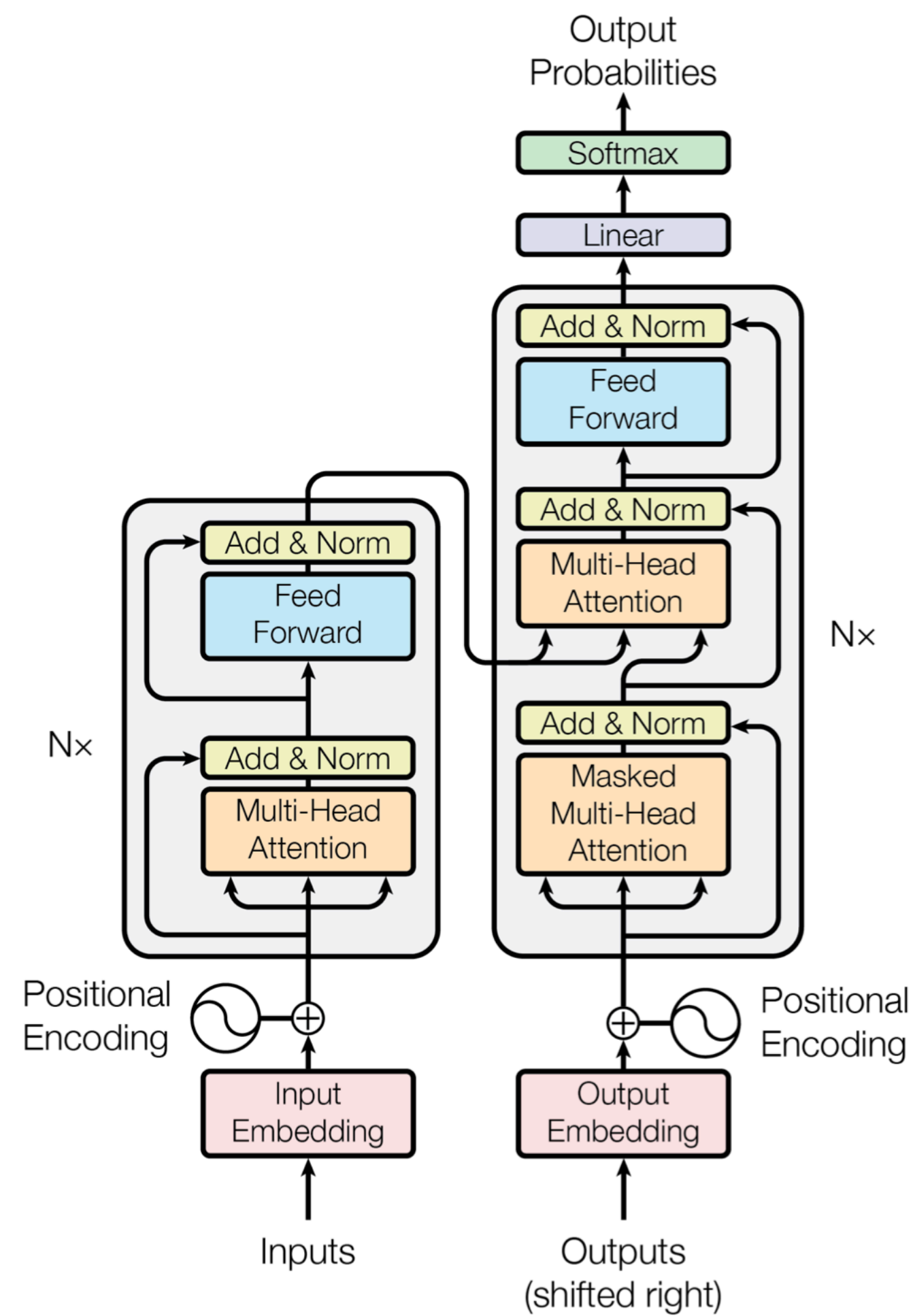
● Experimental result  
● Computational prediction

## Protein folding





# What is a Transformer?



---

## 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\*<sup>†</sup>**  
University of Toronto  
aidan@cs.toronto.edu

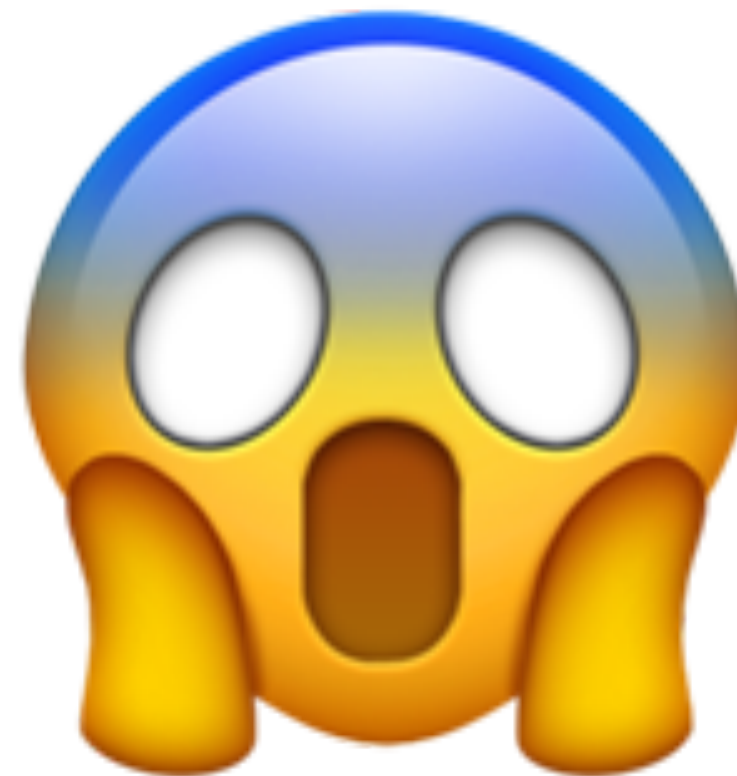
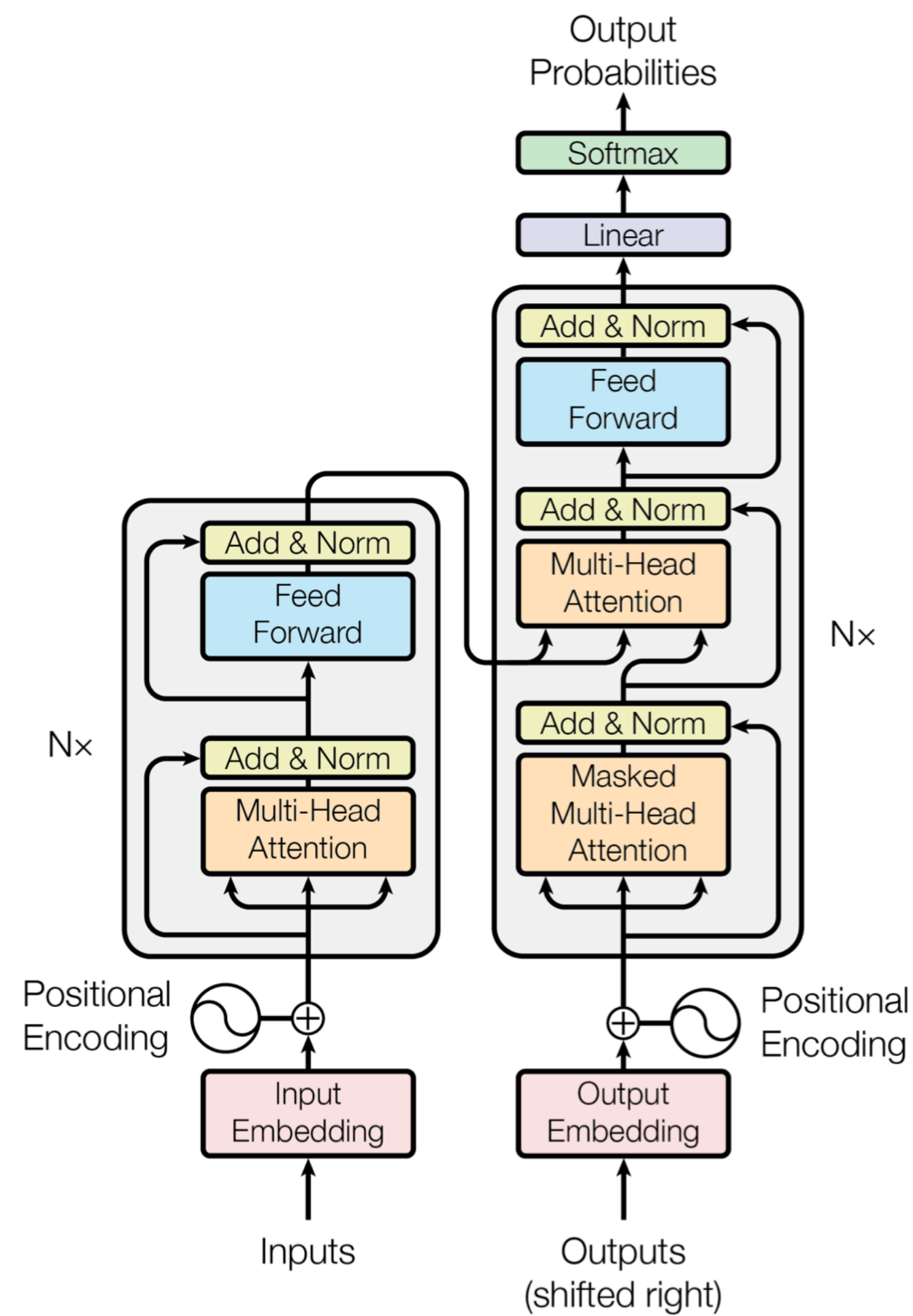
**Łukasz Kaiser\***  
Google Brain  
lukaszkaiser@google.com

**Illia Polosukhin\*<sup>‡</sup>**  
illia.polosukhin@gmail.com

[Link to paper](#)



# What is a Transformer?



---

## 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\*<sup>†</sup>**  
University of Toronto  
aidan@cs.toronto.edu

**Łukasz Kaiser\***  
Google Brain  
lukaszkaiser@google.com

**Illia Polosukhin\*<sup>‡</sup>**  
illia.polosukhin@gmail.com

[Link to paper](#)



# Main ingredients



Attention  
mechanisms



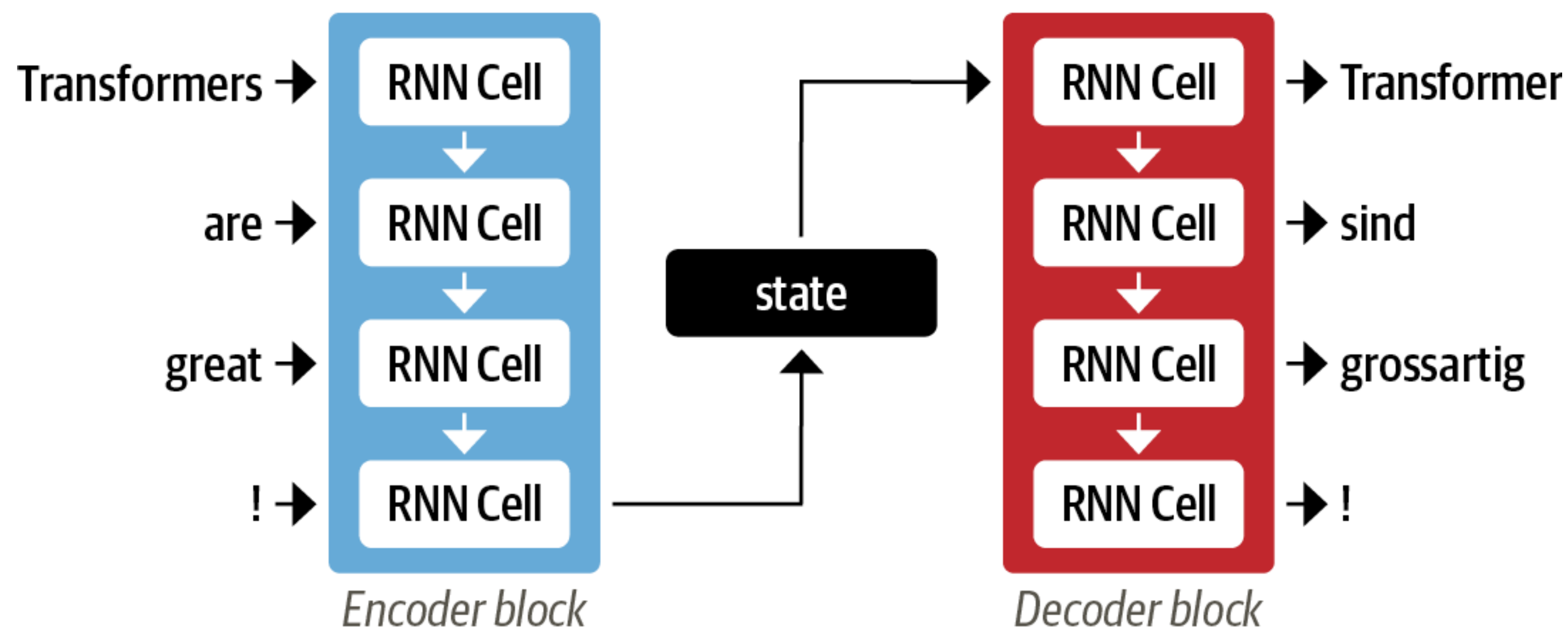
Self-supervised learning  
(Pretraining)



Transfer learning  
(Fine-tuning)



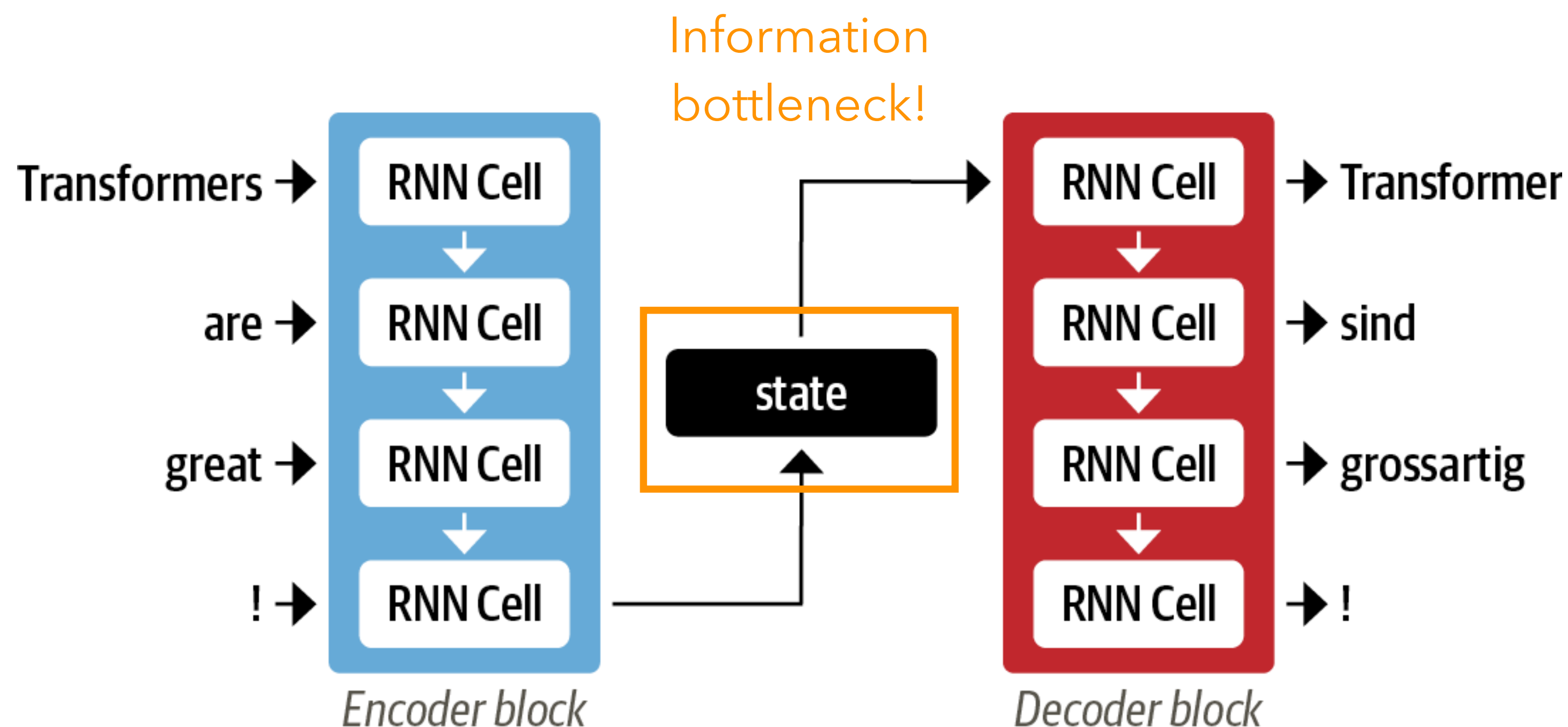
# Attention mechanisms



Originally developed for recurrent neural networks



# Attention mechanisms

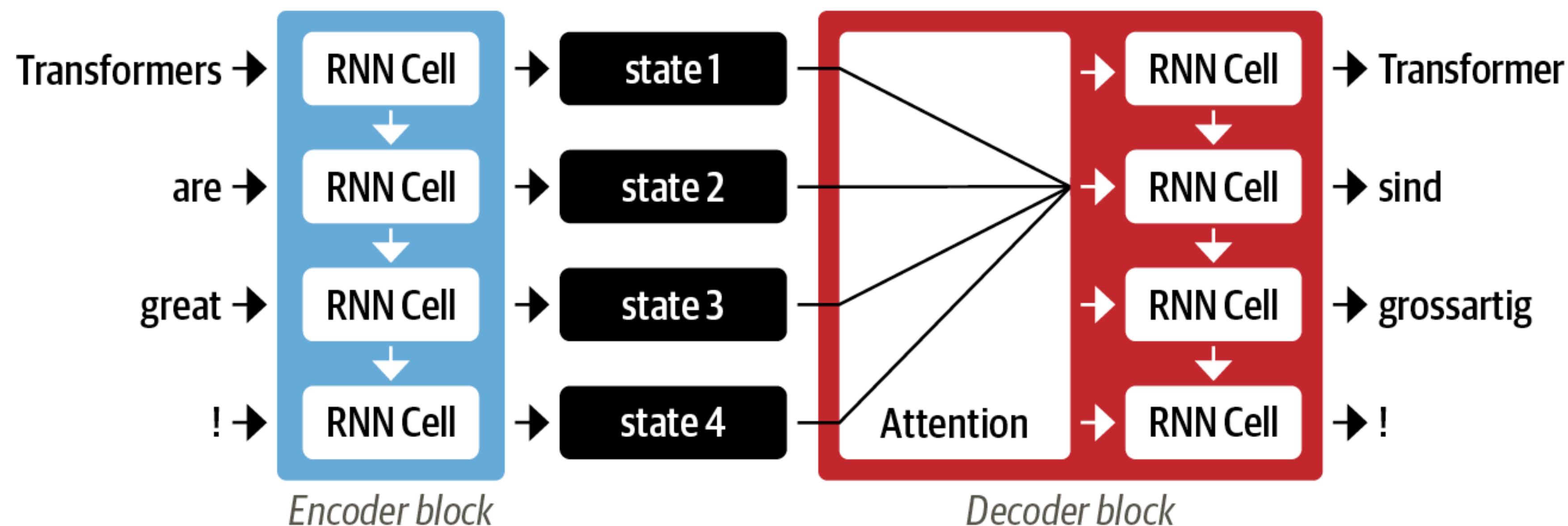


Originally developed for recurrent neural networks





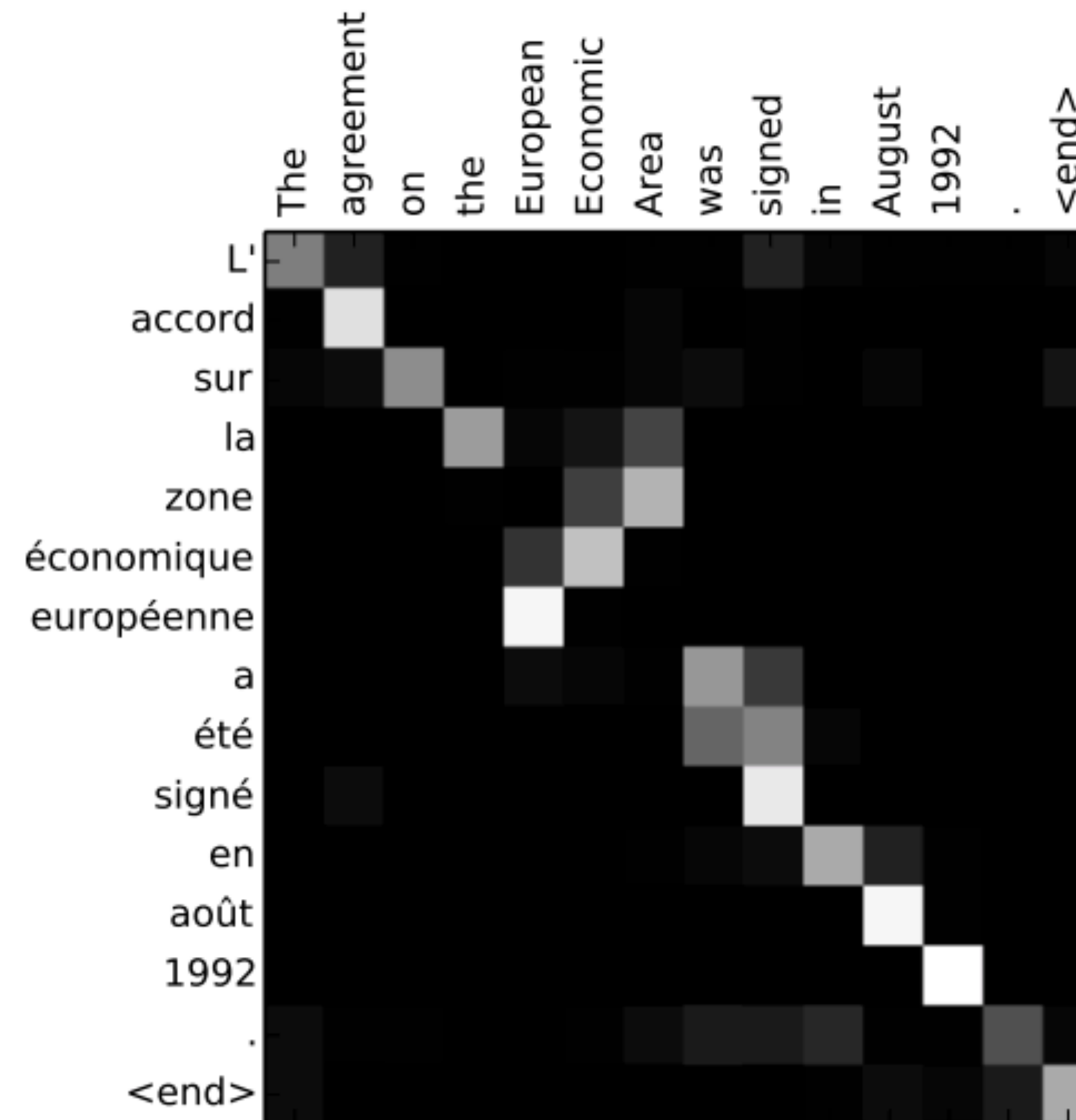
# Attention mechanisms



Assign a weight or "pay attention" to specific states



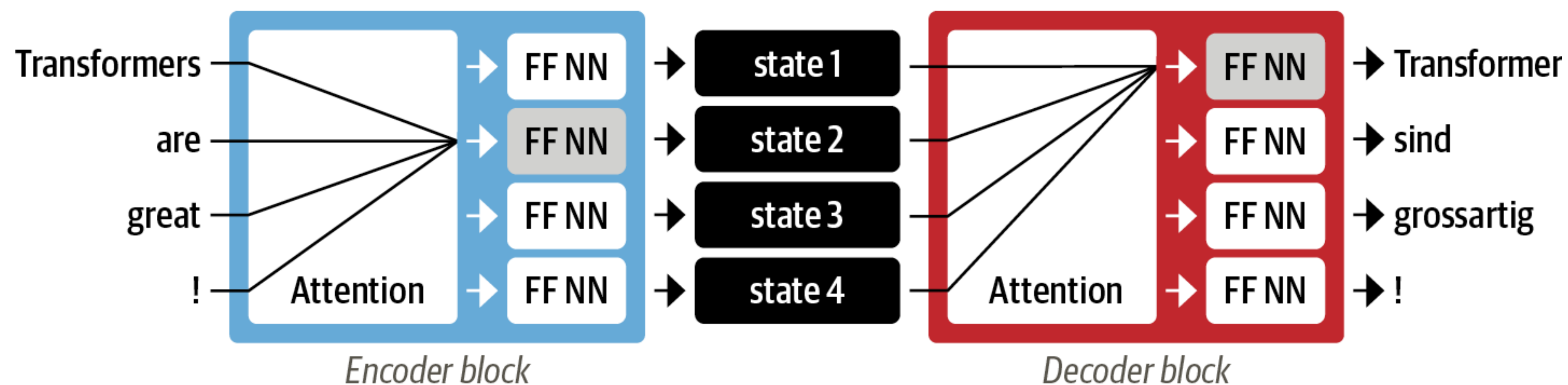
# Attention mechanisms



Attention gives better modelling of word order



# Attention mechanisms



Transformers much easier to scale with compute & data



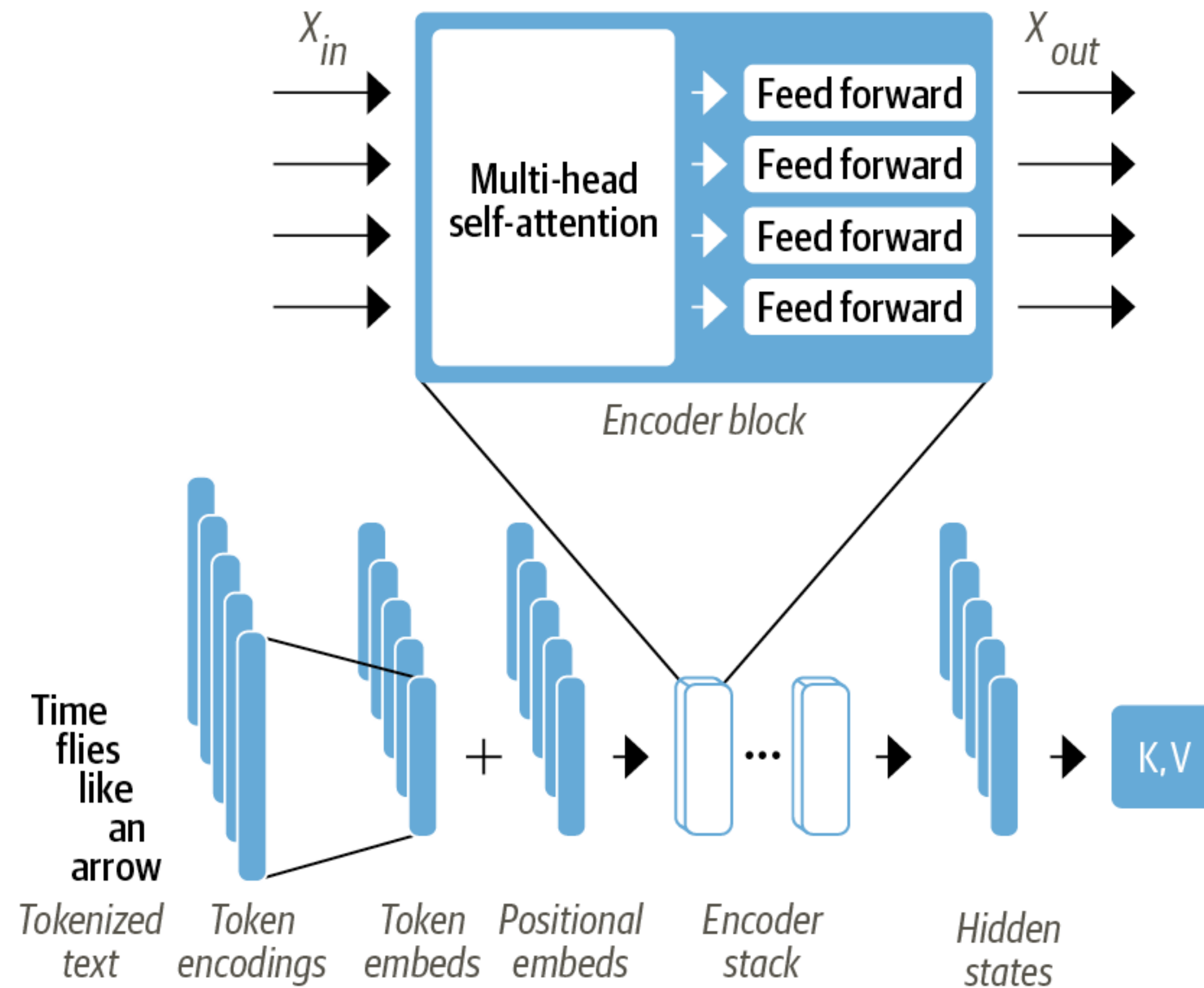
# Scaling Transformers



Scaling with blocks, attention, or dimension



# Scaling Transformers

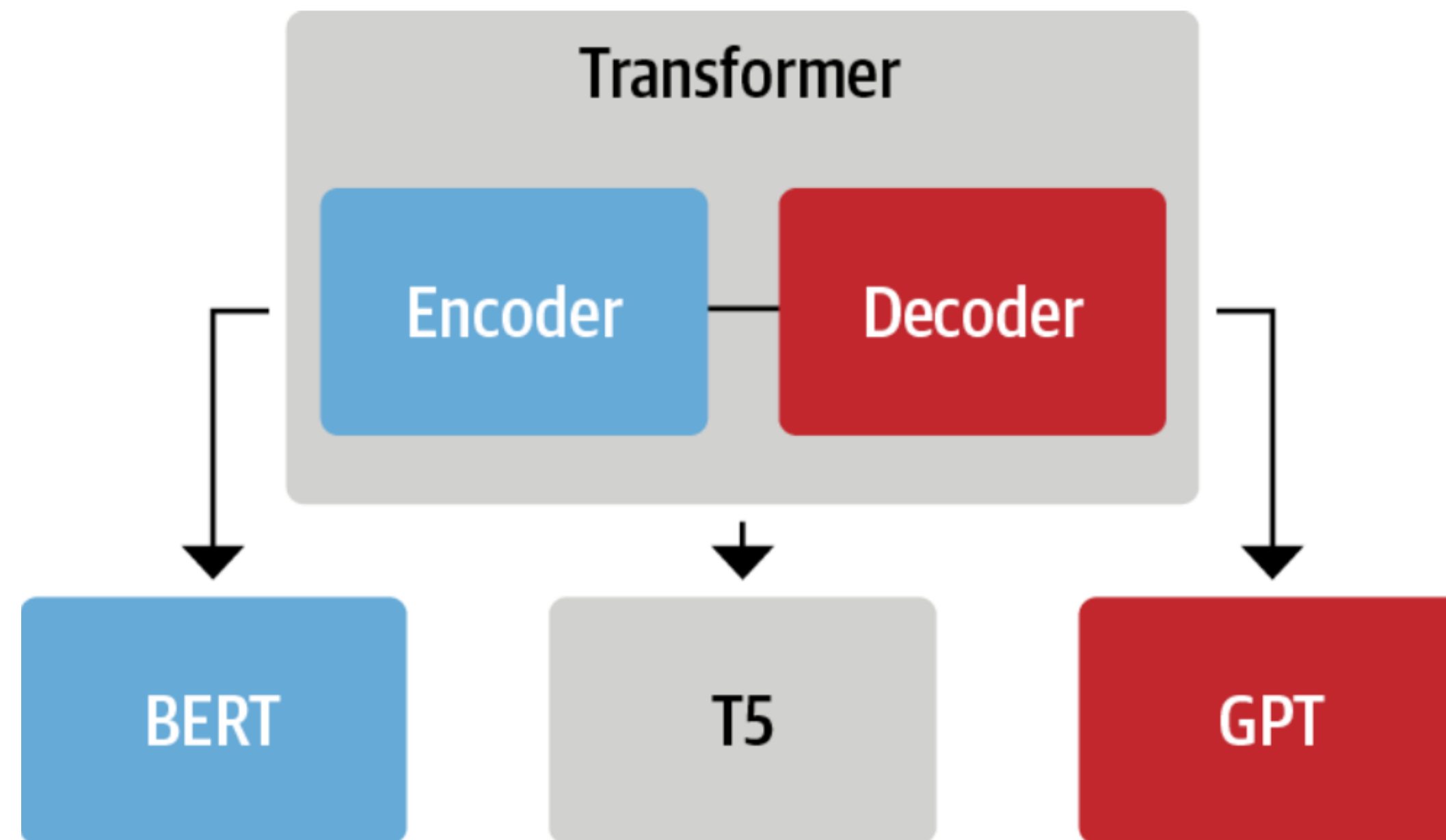


Scaling with blocks, attention, or dimension





# Three types of architectures

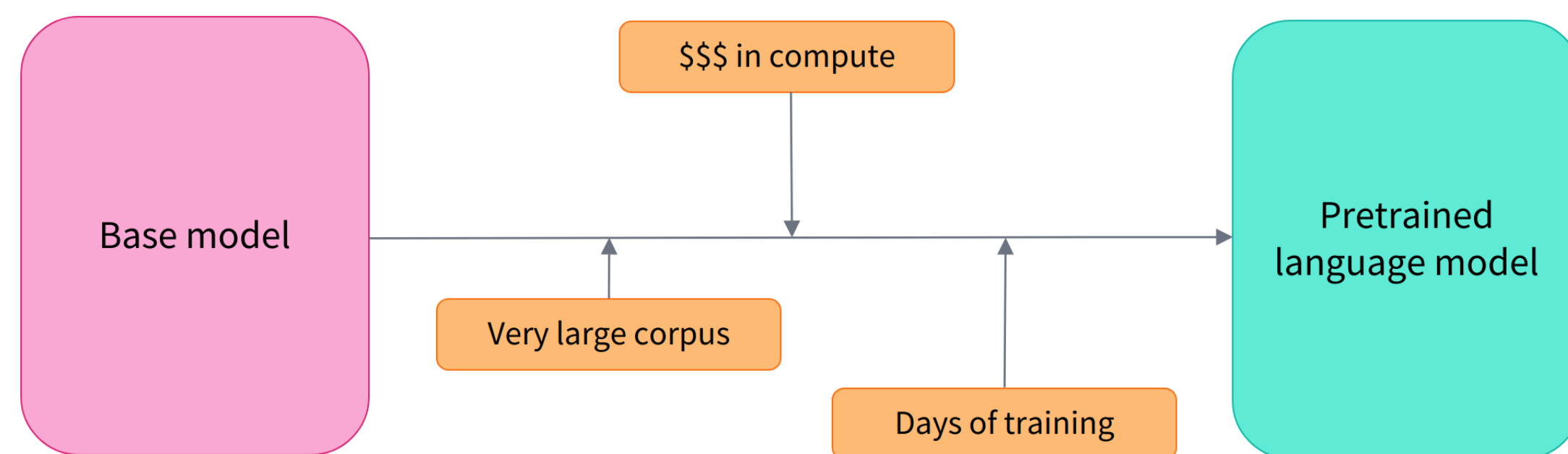


Each architecture excels at specific tasks

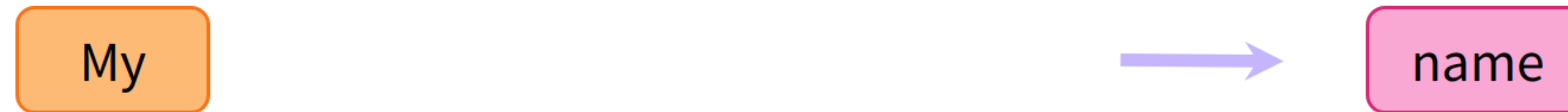


# The modern paradigm

## Pretraining



# Transformer pretraining?



... trained to predict next token



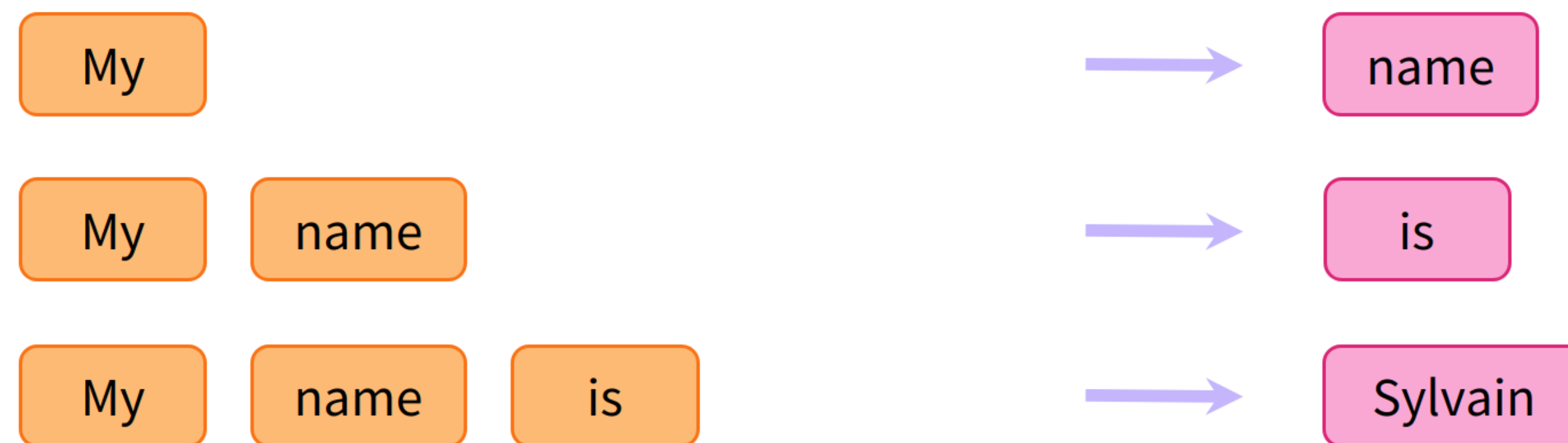
# Transformer pretraining?



... trained to predict next token



# Transformer pretraining?

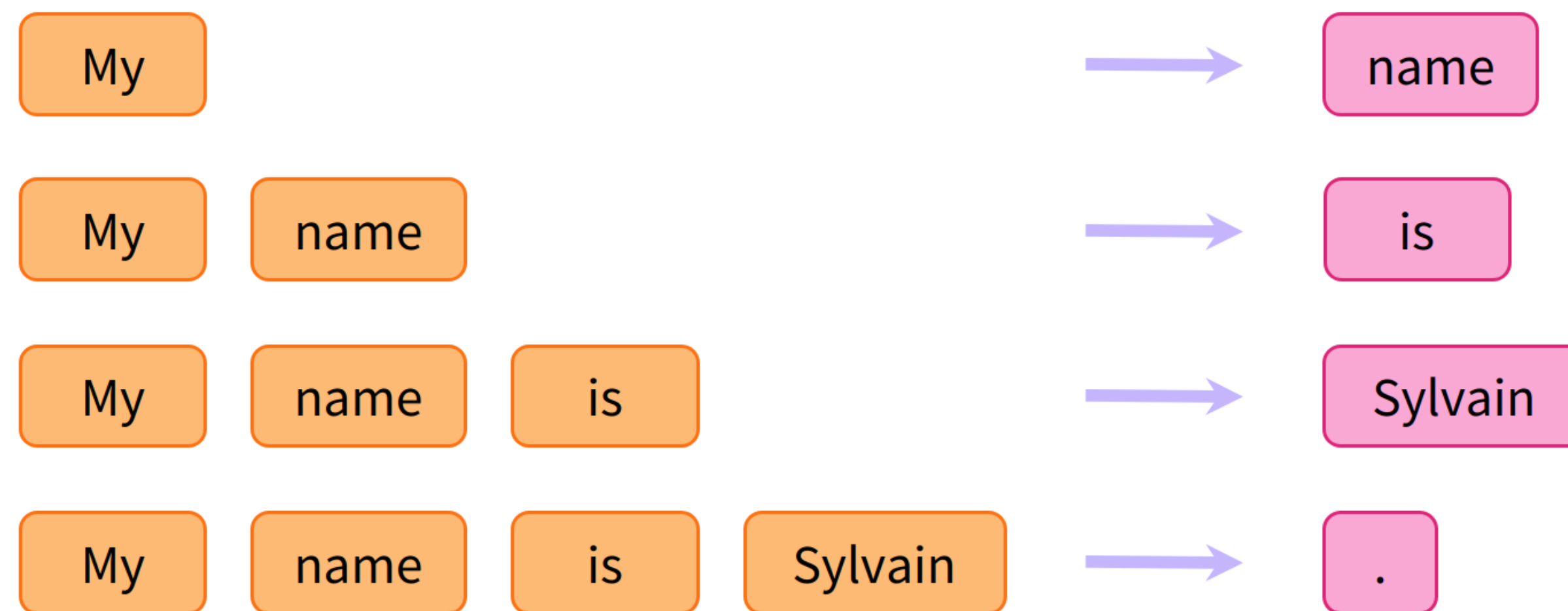


... trained to predict next token





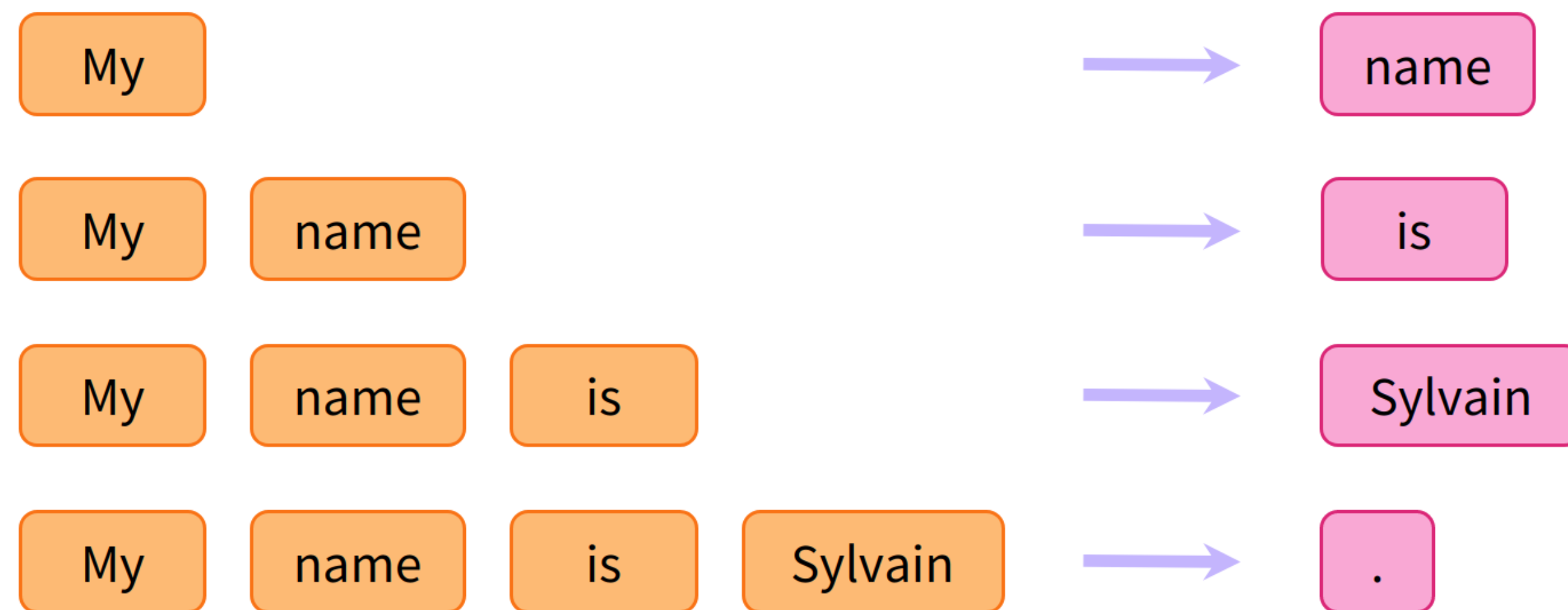
# Transformer pretraining?



... trained to predict next token



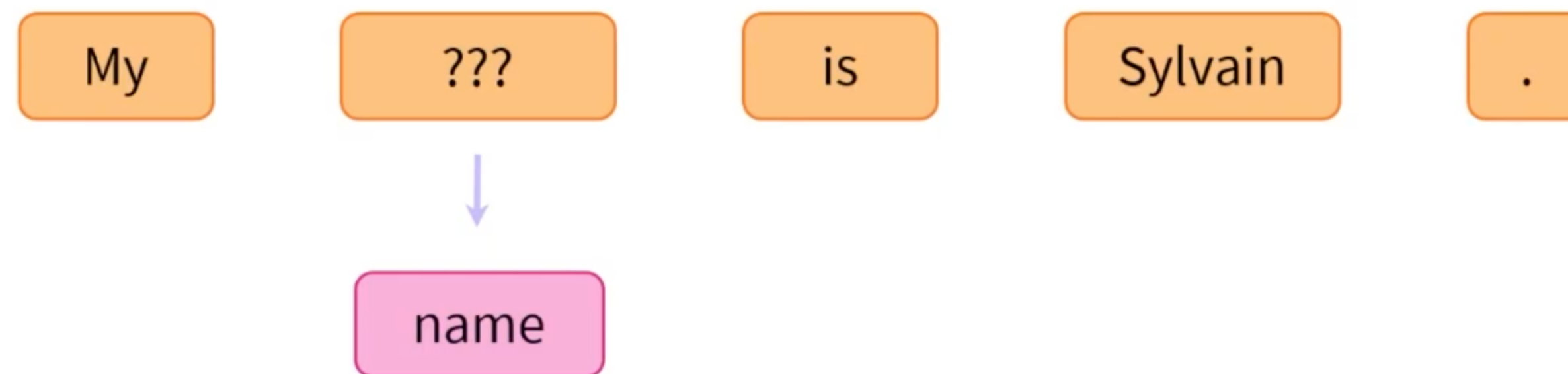
# Transformer pretraining?



... trained to predict next token



# Transformer pretraining?

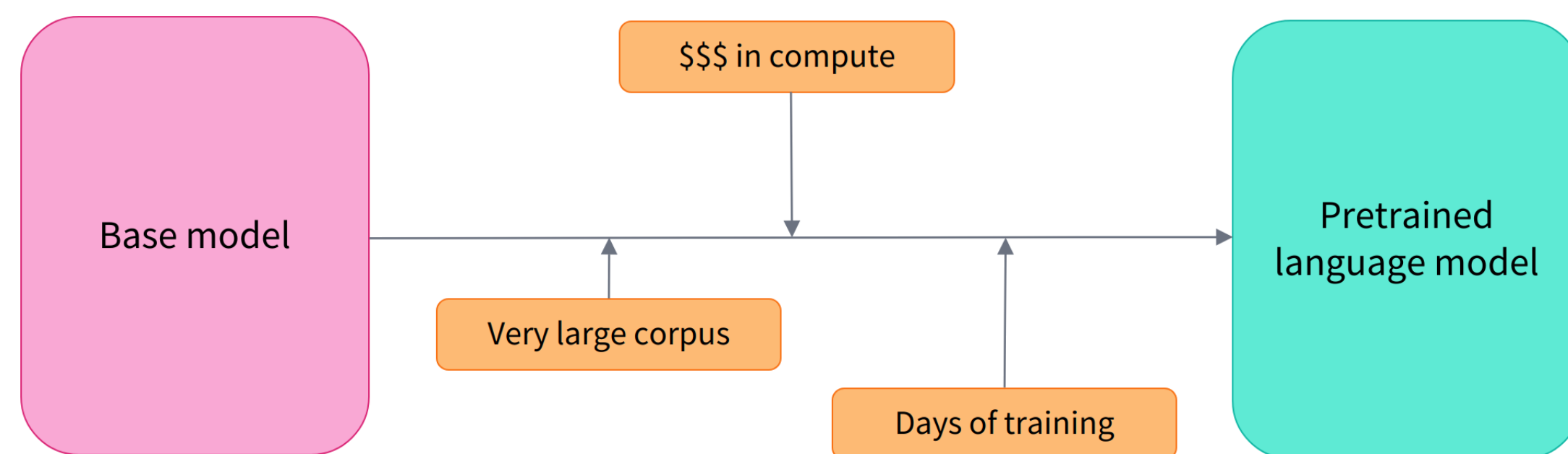


... or to predict the masked token



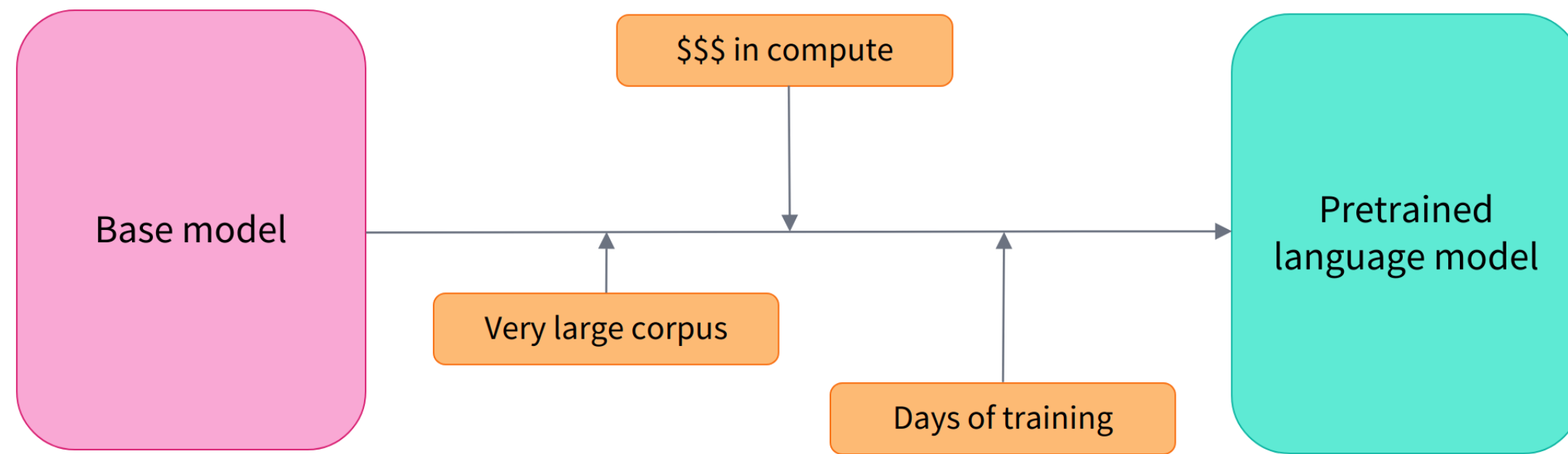
# The modern paradigm

## Pretraining

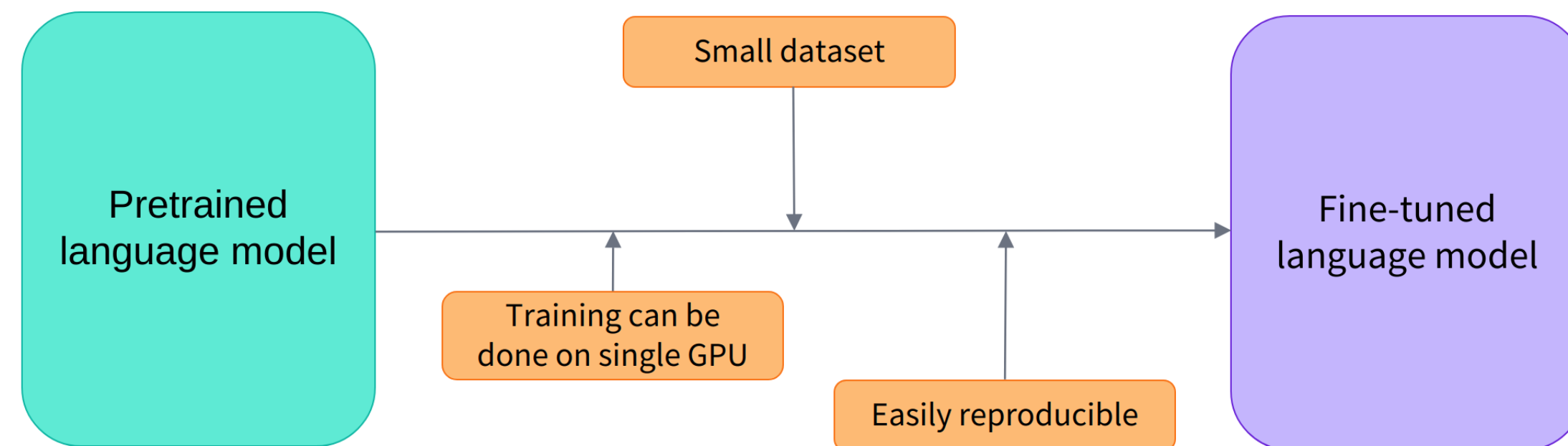


# The modern paradigm

## Pretraining

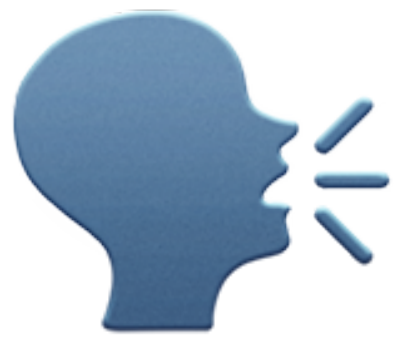


## Fine-tuning





# Main challenges



Language barrier



Black boxes



Data hungry



Biases



# Bridging the science / industry divide



Humble Data Scientist



# Get the code & weights?

arXiv.org > cs > arXiv:1706.03762

Search...  
Help | Adv

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder–decoder configuration. 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 be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English–to–German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English–to–French translation task, our model establishes a new single–model state–of–the–art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Comments: 15 pages, 5 figures  
Subjects: **Computation and Language (cs.CL)**; Machine Learning (cs.LG)  
Cite as: [arXiv:1706.03762 \[cs.CL\]](#)  
(or [arXiv:1706.03762v5 \[cs.CL\]](#) for this version)

Submission history


From: Ashish Vaswani [[view email](#)]  
[v1] Mon, 12 Jun 2017 17:57:34 UTC (1,102 KB)  
[v2] Mon, 19 Jun 2017 16:49:45 UTC (1,125 KB)  
[v3] Tue, 20 Jun 2017 05:20:02 UTC (1,125 KB)  
[v4] Fri, 30 Jun 2017 17:29:30 UTC (1,124 KB)  
[v5] Wed, 6 Dec 2017 03:30:32 UTC (1,124 KB)

Bibliographic ToolsCode & DataRelated PapersAbout arXivLabs


Code and Data Associated with this Article

☒ arXiv Links to Code & Data ([What is Links to Code & Data?](#))

Official Code

 <https://github.com/tensorflow/tensor2tensor>

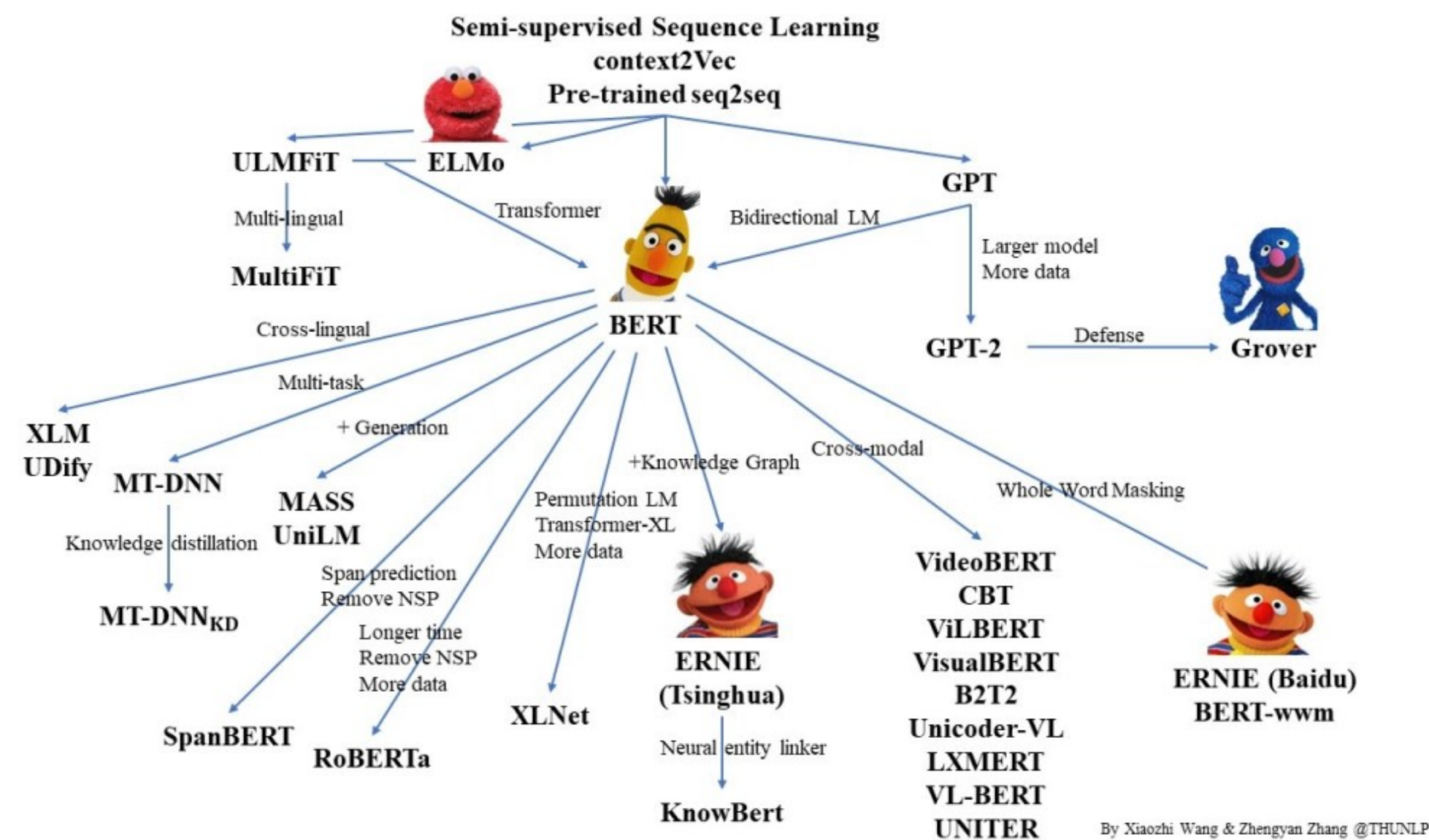
Community Code

 455 code implementations (in PyTorch, TensorFlow, MXNet and JAX)





# The wild west of open-source ML



## Python 2? Really? #8



impredicative opened this issue on 11 Jul 2019 · 3 comments

## Oh c'mon you guys... #2



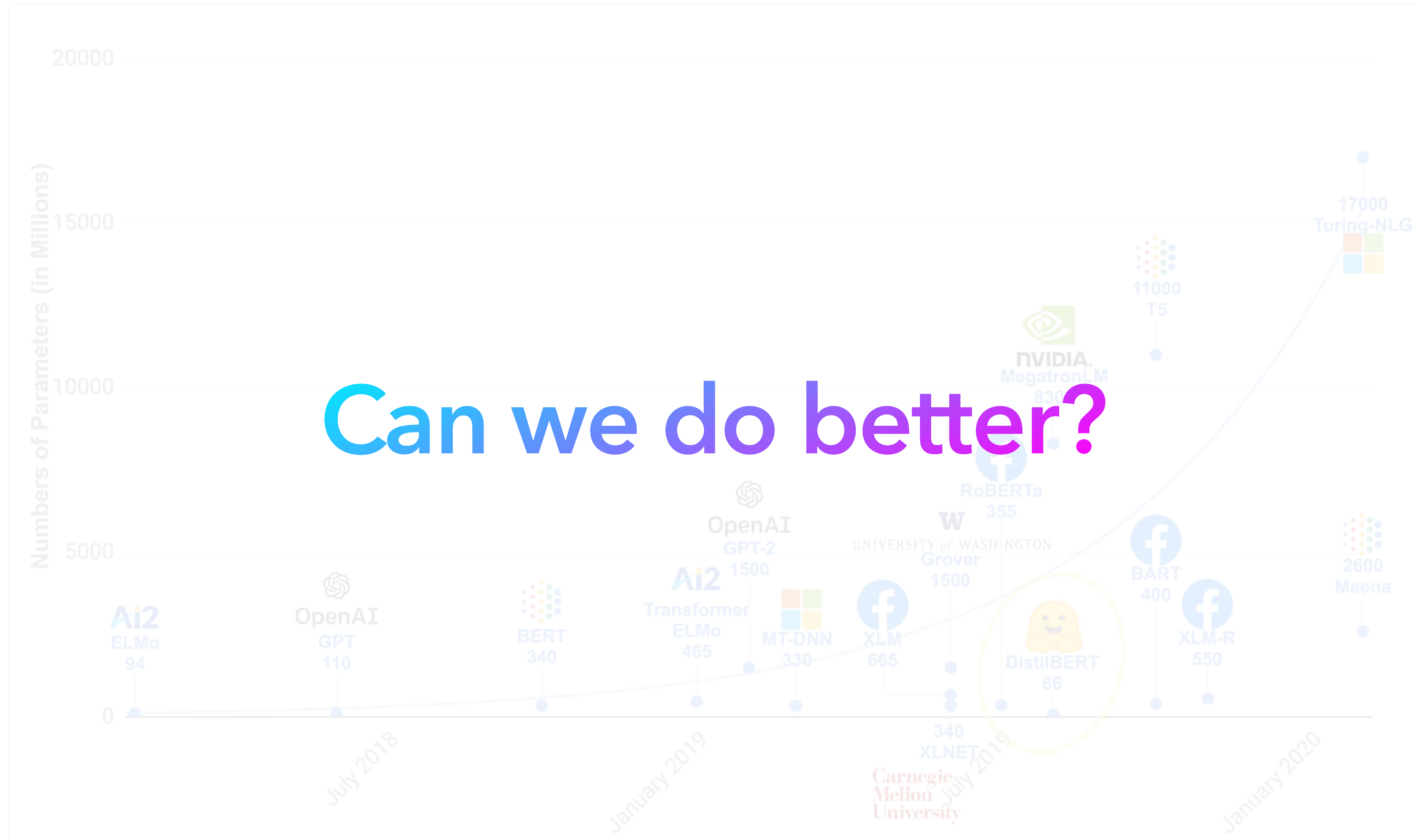
kcrosley-leisurelabs opened this issue on 18 Jun 2020 · 15 comments

Explosion of pretrained models:  
which one do I choose?

Different APIs, missing docs,  
reproducibility issues, ...



# Can we do better?



Check out the first part of the [Hugging Face Course](#) and learn how the HF Ecosystem works!



# The AI community building the future.

Build, train and deploy state of the art models powered by  
the reference open source in natural language processing.



51,261

More than 5,000 organizations are using Hugging Face



**Allen Institute for AI**  
Non-Profit • 57 models



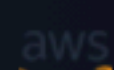
**Facebook AI**  
Company • 132 models



**asteroid-team**  
Non-Profit



**Google AI**  
Company • 149 models



**Amazon Web Services**  
Company • 1 model



**SpeechBrain**  
Non-Profit • 27 models



**Microsoft**  
Company • 65 models



**Grammarly**  
Company







# Hugging Face

The AI community building the future.

📍 NYC + Paris

🔗 <https://huggingface.co/>

Verified

🏠 Overview

📁 Repositories 190

📦 Packages

👤 People 105

👥 Teams 8

📅 Projects 4

❤️ Sponsoring 4

## Pinned



**transformers**

Public

🤖 Transformers: State-of-the-art Natural Language Processing for Pytorch, TensorFlow, and JAX.

Python 51.3k 12.1k



**datasets**

Public

🤖 The largest hub of ready-to-use datasets for ML models with fast, easy-to-use and efficient data manipulation tools

Python 9.8k 1.2k



**tokenizers**

Public

🔥 Fast State-of-the-Art Tokenizers optimized for Research and Production

Rust 4.8k 385



**knockknock**

Public

👋 Knock Knock: Get notified when your training ends with only two additional lines of code

Python 2.2k 192



**accelerate**

Public

🚀 A simple way to train and use PyTorch models with multi-GPU, TPU, mixed-precision

Python 1.8k 97



**huggingface\_hub**

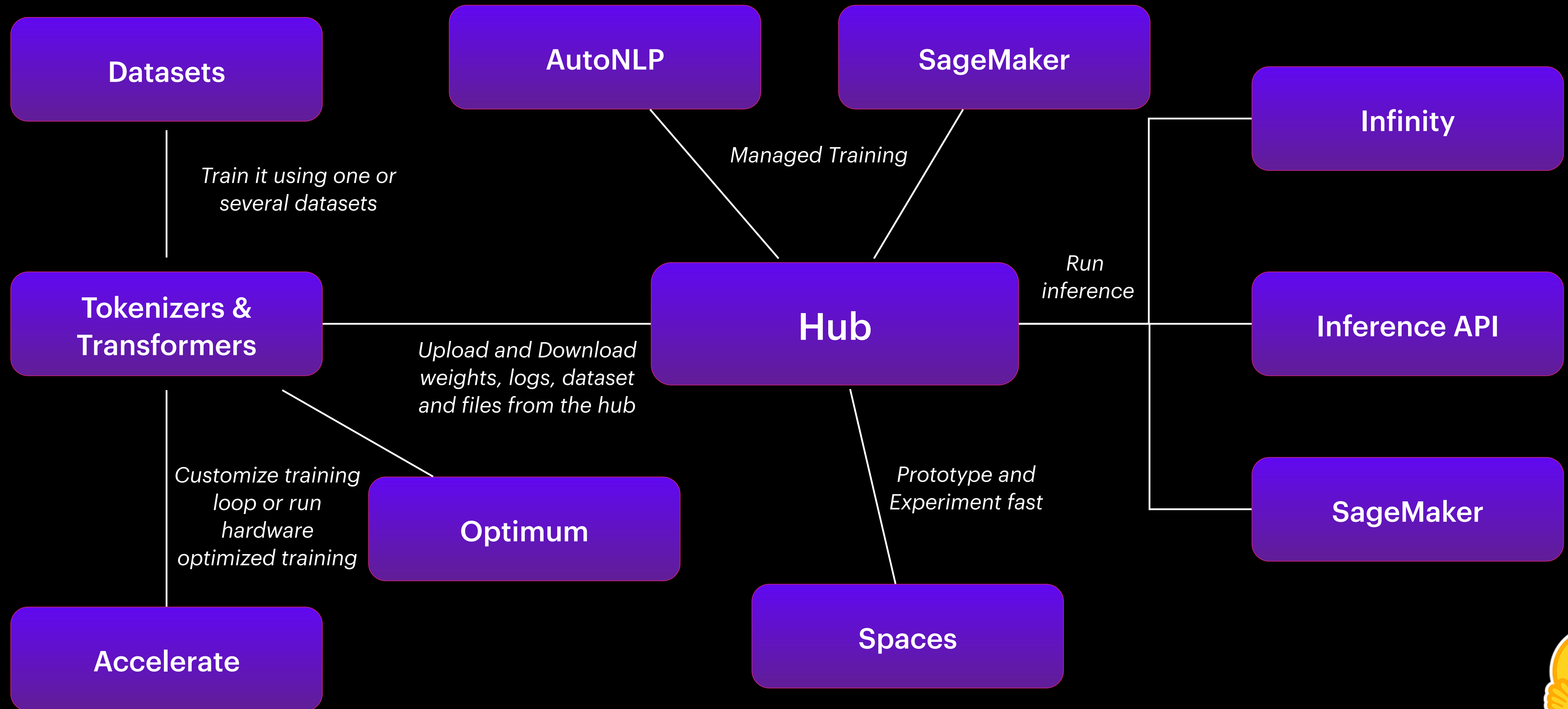
Public

All the open source things related to the Hugging Face Hub.

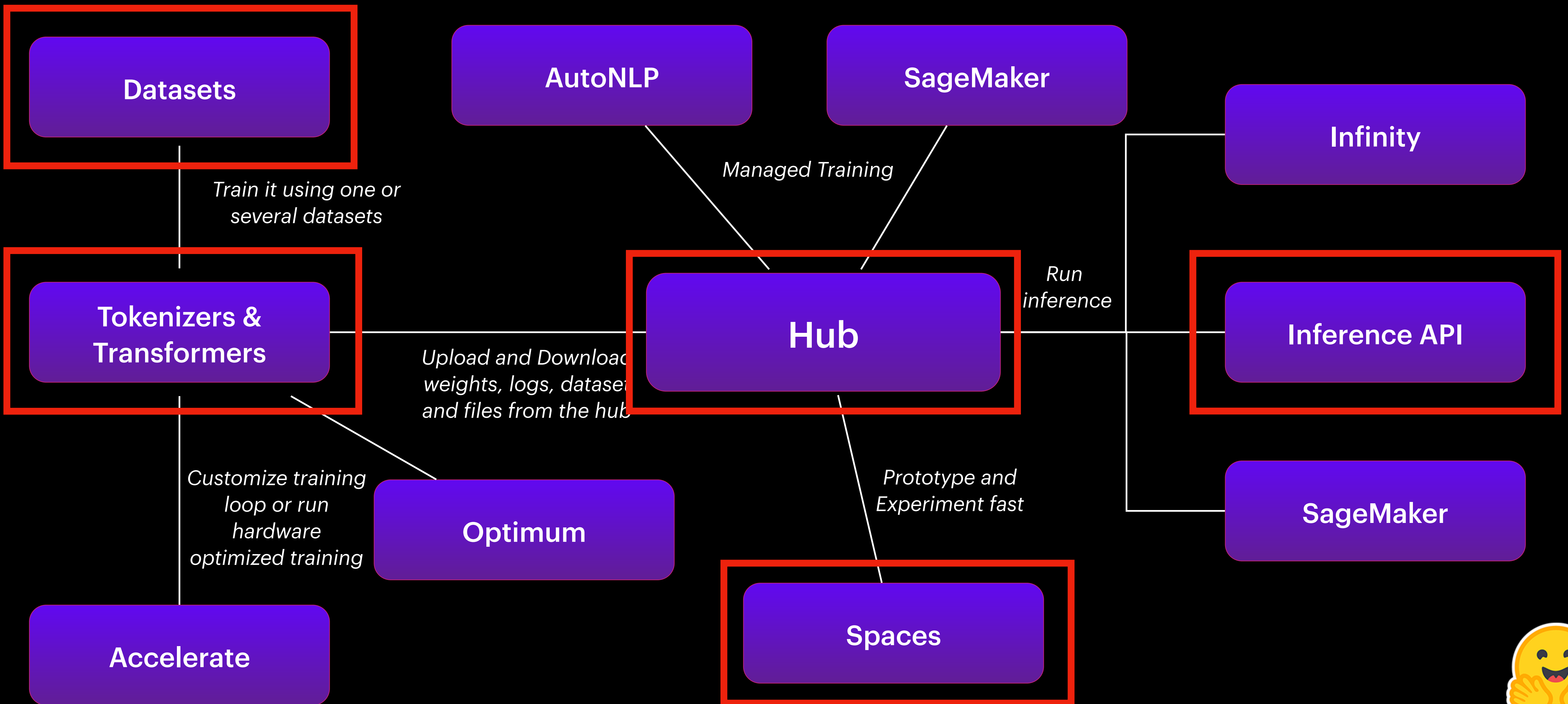
Python 209 40



# Ecosystem Overview



# Ecosystem Overview



# Workshop materials

**Notebooks:** <https://github.com/huggingface/workshops/>

 **Account:** <https://huggingface.co/join>

