CS 639: Foundation Models
Architectures I

Fred Sala

University of Wisconsin-Madison

Feb. 17, 2026

Announcements

- Midterm: **March 11, 5:40 pm - 7:20 pm**
  - Location: **Ingraham Hall, Room B10**
- Homework 1: out!
- **Resources**
  - https://jalammar.github.io/illustrated-transformer/
- Class roadmap:

Tuesday Feb. 17	Architectures: Encoder-Only
Thursday Feb. 19	Architectures: Others
Tuesday Feb. 24	Attention Variants
Thursday Feb. 26	Multimodal Architectures

LLMs, FMs and Arch

Outline

- **Finish up last time: full Transformer architecture**
  - Encoder layer, decoder layer, full original Transformer architecture
- **Encoder-only models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Preview of decoder-only models**
  - GPT architecture, examples

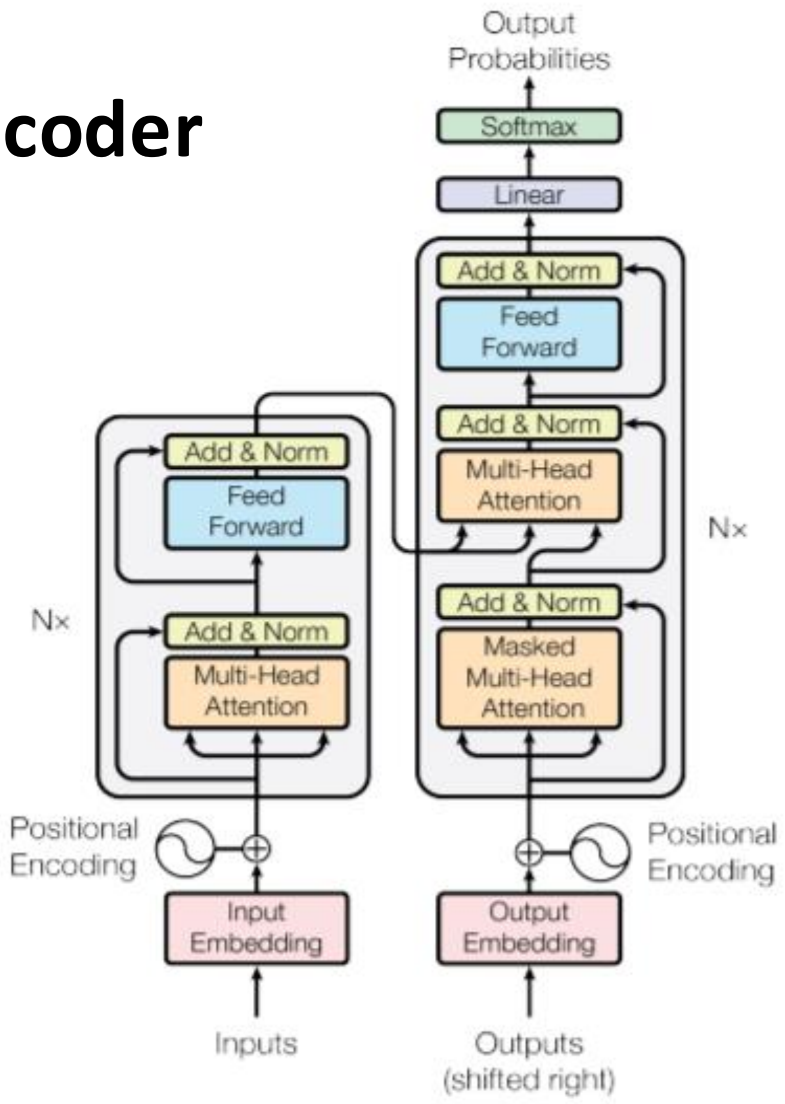
Outline

- **Finish up last time: full Transformer architecture**
  - Encoder layer, decoder layer, full original Transformer architecture
- **Encoder-only models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Preview of decoder-only models**
  - GPT architecture, examples

Transformers: Model Architecture

- Initial goal for an architecture: **encoder-decoder**
  - Get **rid of recurrence**
  - Replace with **self-attention**

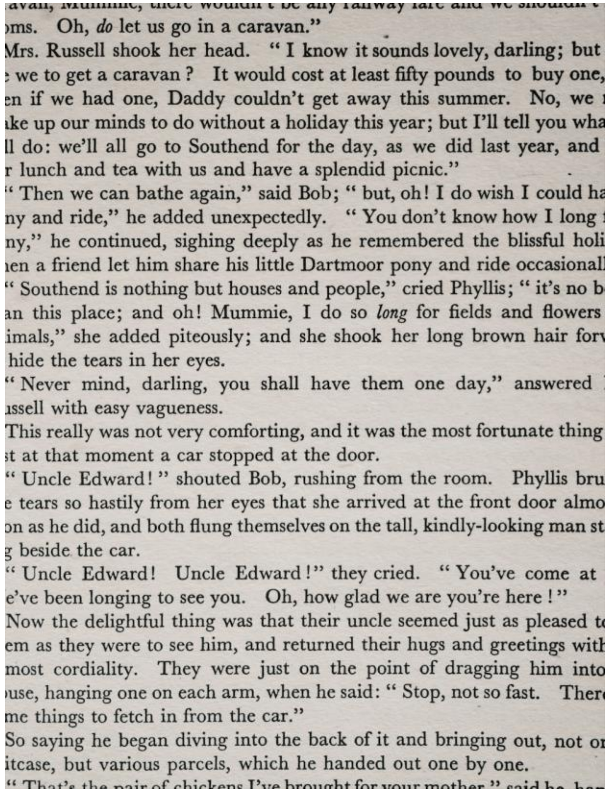
- Architecture
  - You may have seen this picture
  - Centered on self-attention blocks

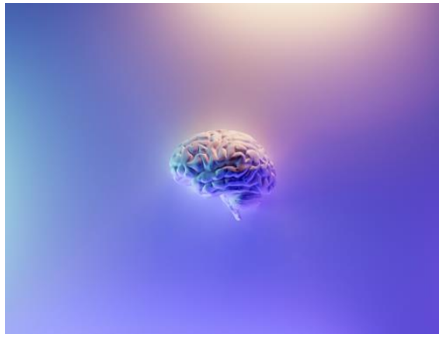


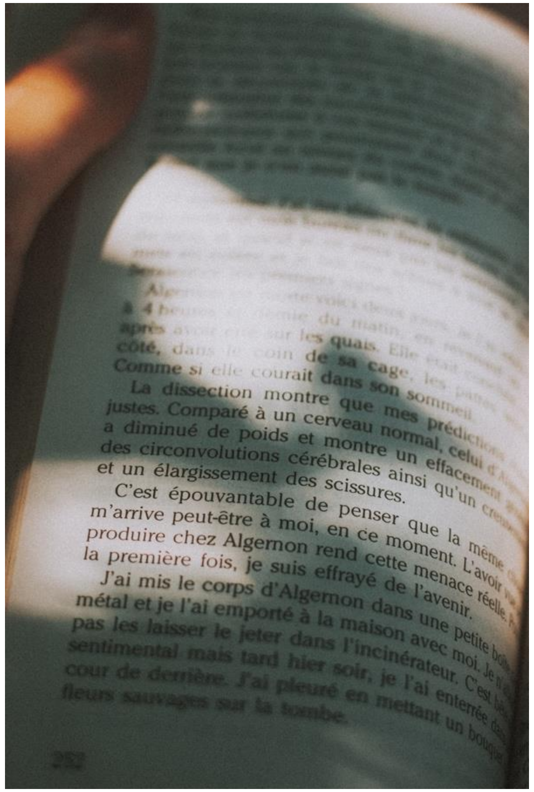
Vaswani et al. ‘17

Interlude: Encoder-Decoder Models

- Translation tasks: natural encoder-decoder architecture
- Intuition:

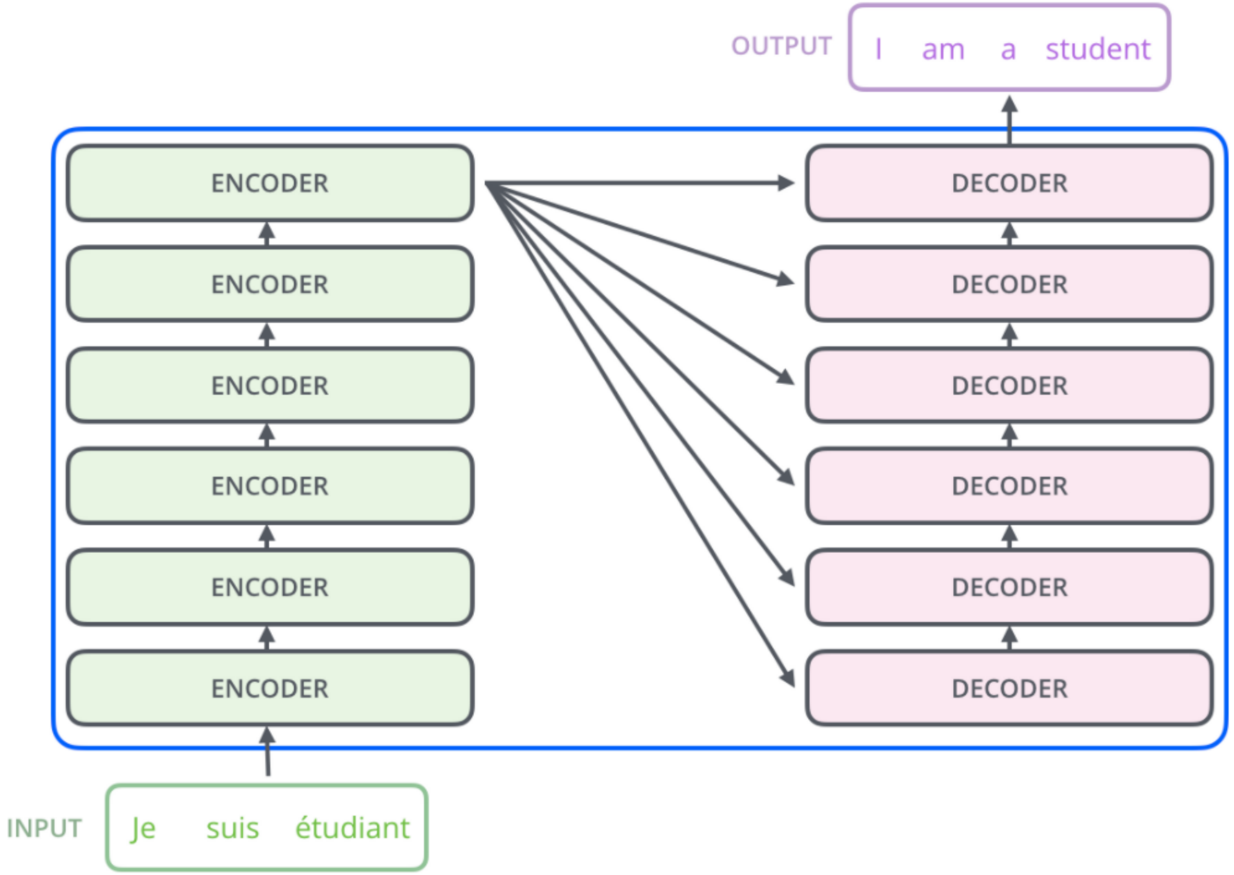






Transformers: Architecture

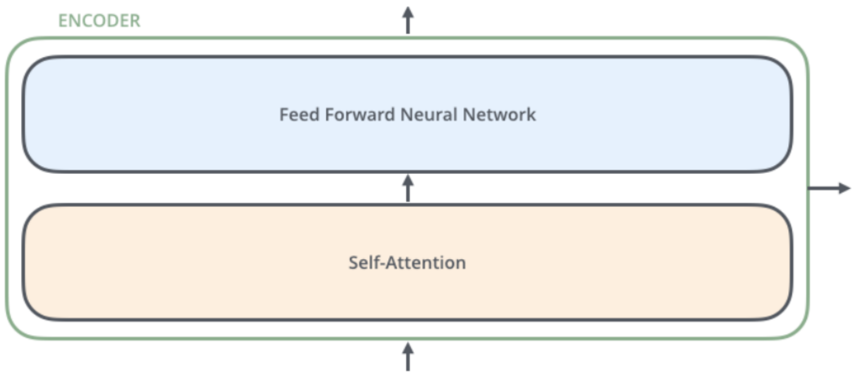
- **Sequence-sequence** model with **stacked** encoders/decoders:
  - For example, for French-English translation:

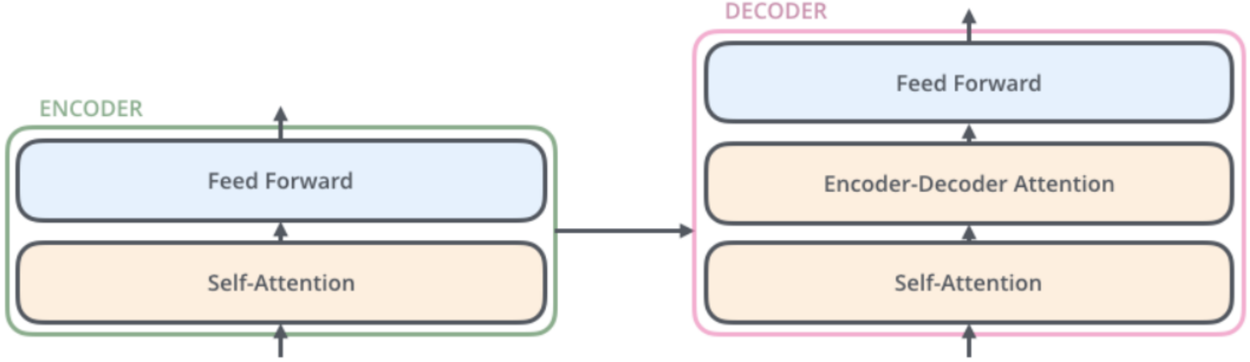


Transformers: Architecture

- Sequence-sequence model with **stacked** encoders/decoders:
  - What's inside each encoder/decoder unit?

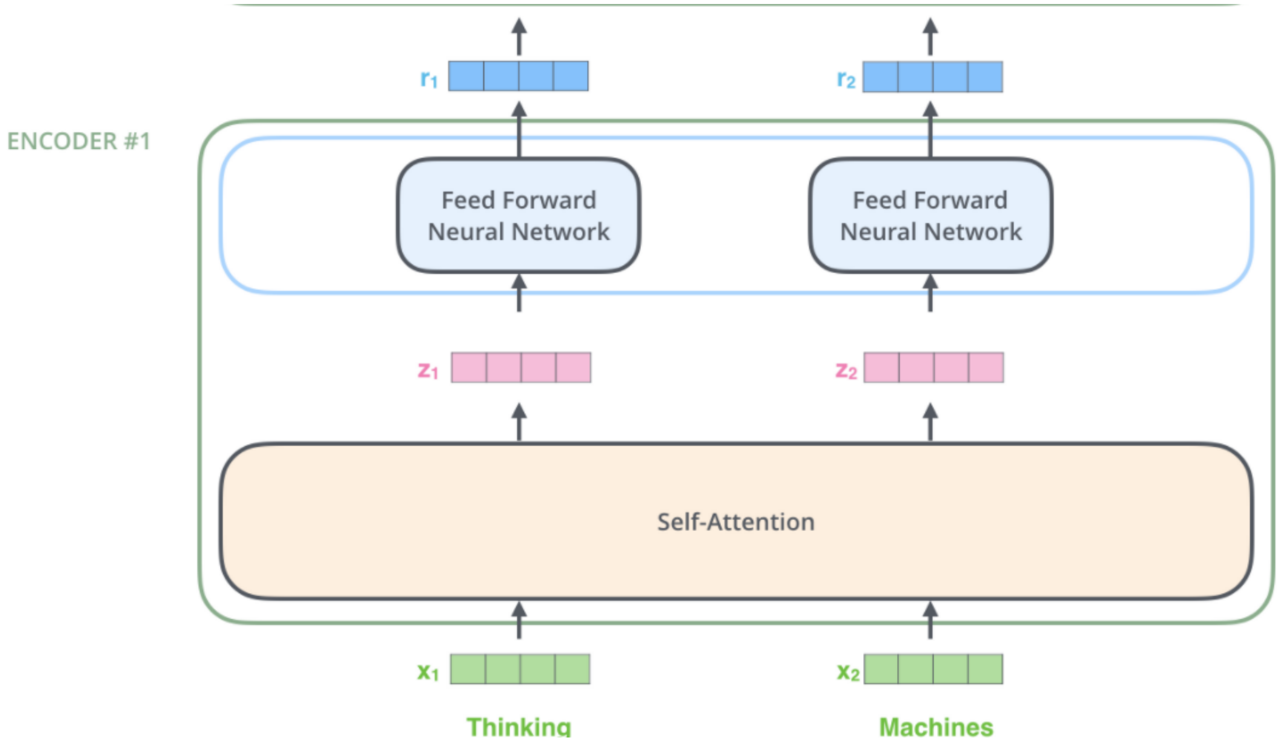
  - Focus encoder first: **pretty simple**! 2 components:
    - Self-attention block
    - Fully-connected layers (i.e., an MLP)





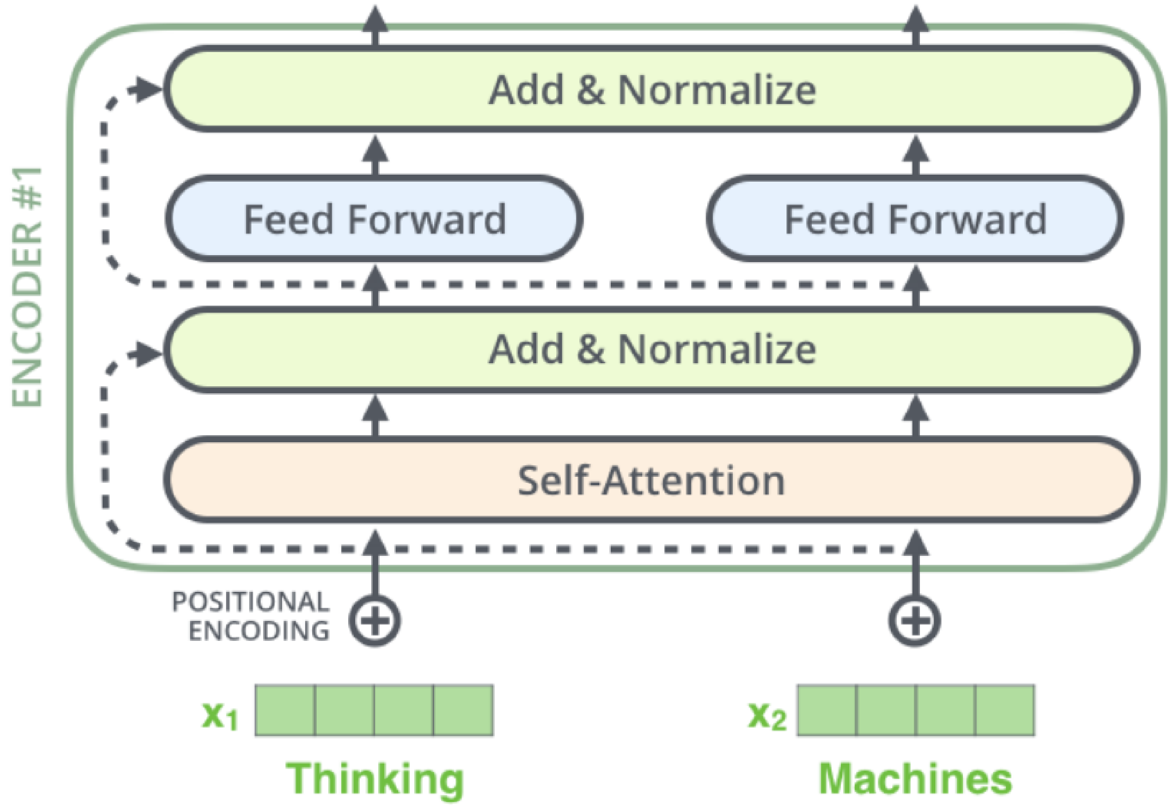
Transformers: Inside an Encoder

- Let's take a look at the encoder. Two components:
  - 1. **Self-attention** layer (covered this)
  - 2. "Independent" **feedforward nets**
    - **Note: same** MLP (often 2-layer) at every position



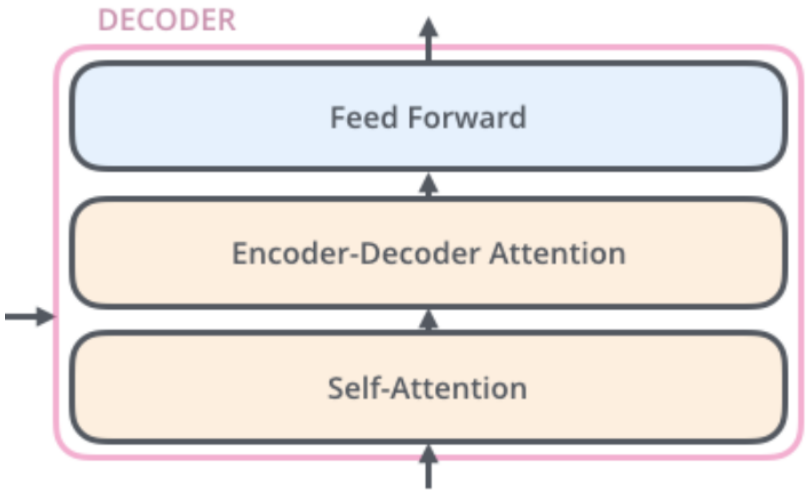
Transformers: More Tricks

- Recall a big innovation for ResNets: residual connections
  - And also layer normalizations
  - Apply to our encoder layers



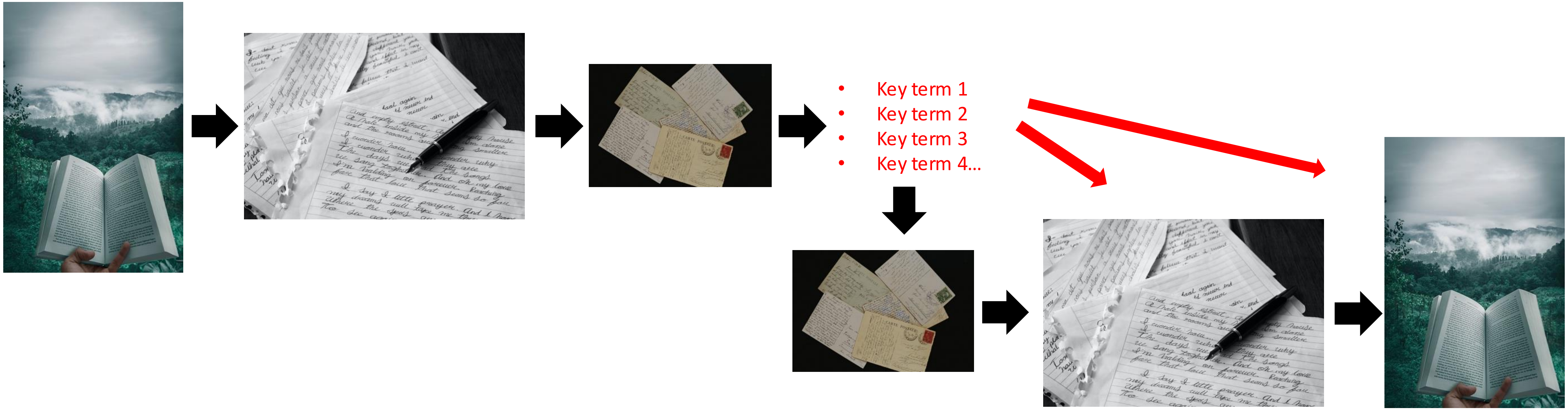
Transformers: Inside a Decoder

- Let's take a look at the decoder. Three components:
  - 1. **Self-attention** layer (covered this)
  - 2. Encoder-decoder attention (same, but K, V come from encoder)
  - 3. "Independent" **feedforward nets**



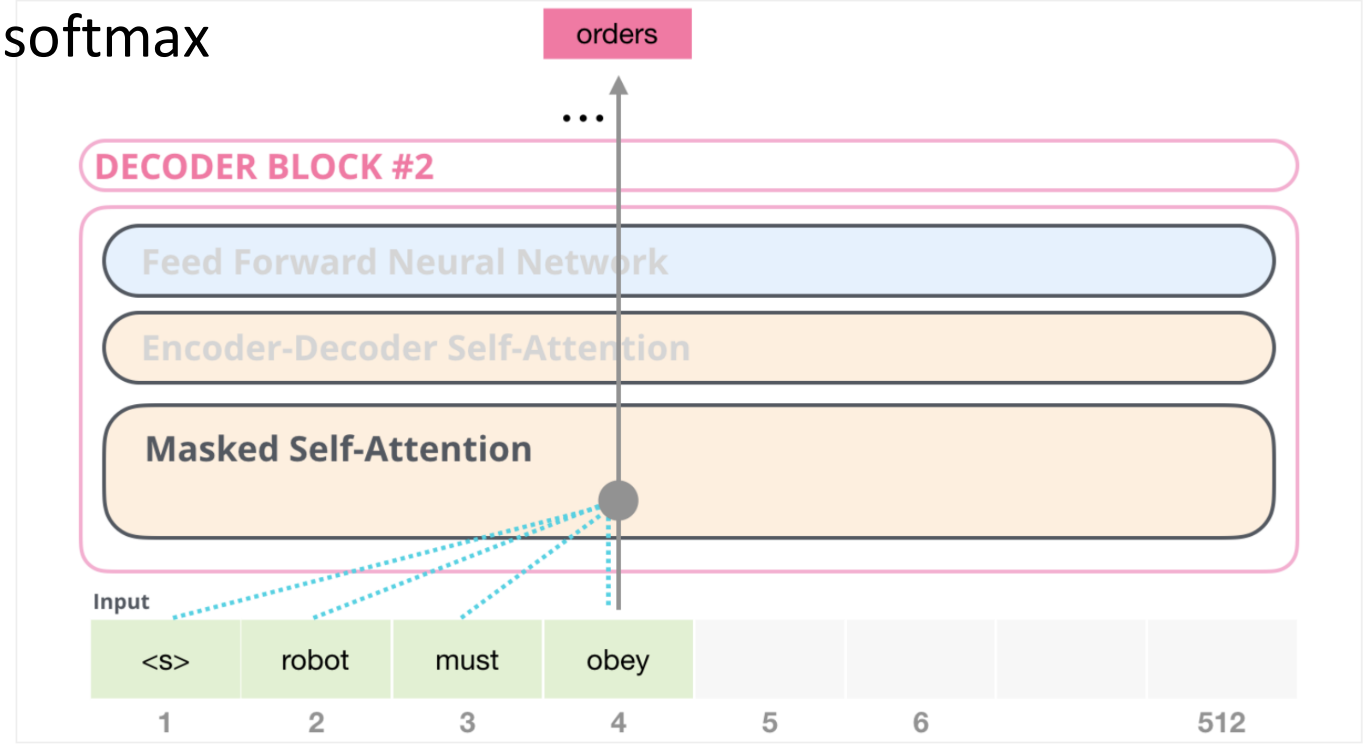
Transformers: Cross-Attention

- Why encoder-decoder attention ?
  - Recall: same as before, but K, V come from encoder
  - Actually more traditional, but… **intuition**:



Transformers: Decoder Masking

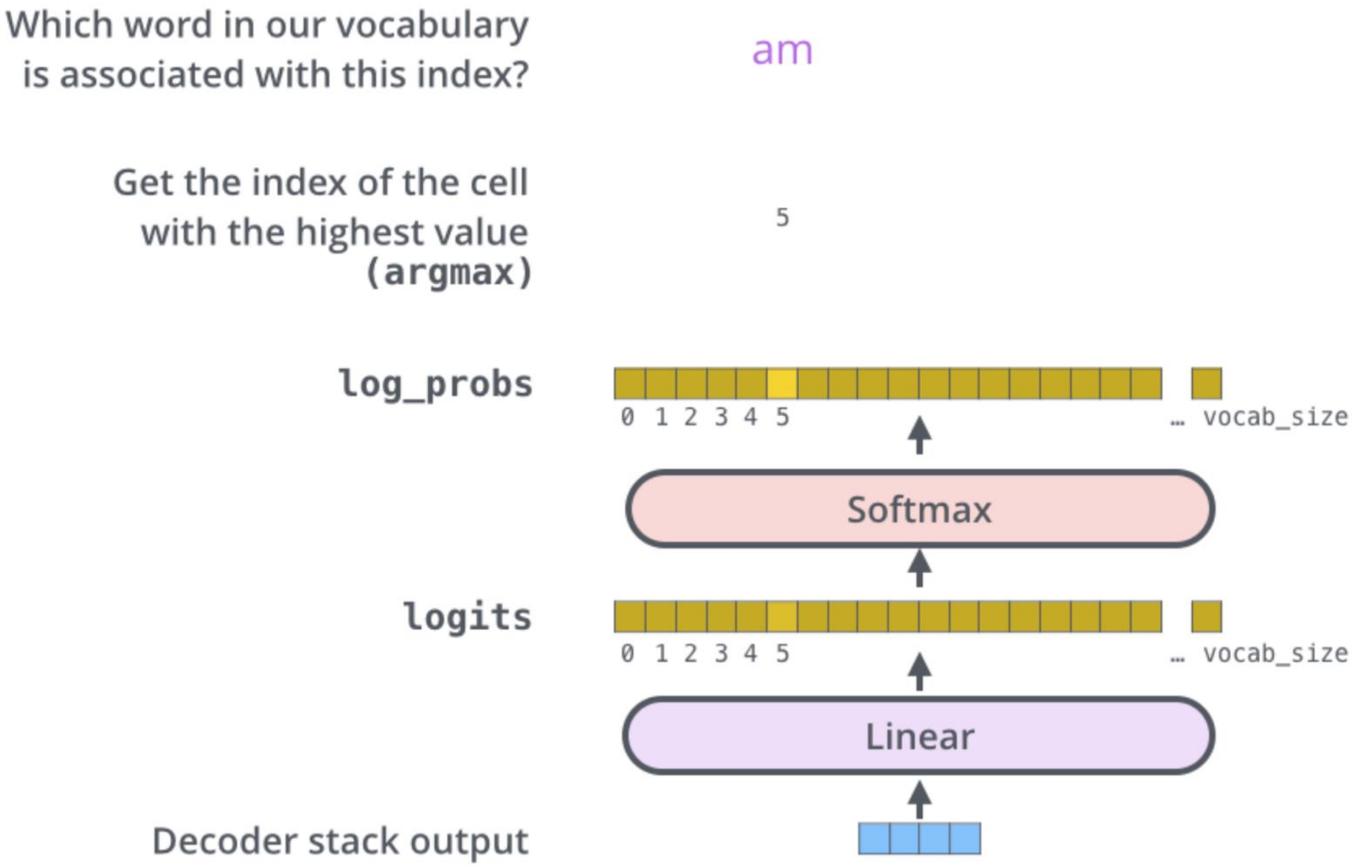
- One more interesting bit!
  - At the decoder level, self-attention changes a bit:
  - Masked instead: block *future* words from being attended to
    - **Important in training (**if we don't---**model could look at future tokens** during training**, but this is impossible in inference)**
    - **How to mask?** Add $-\infty$ before softmax



Transformers: Last Layers

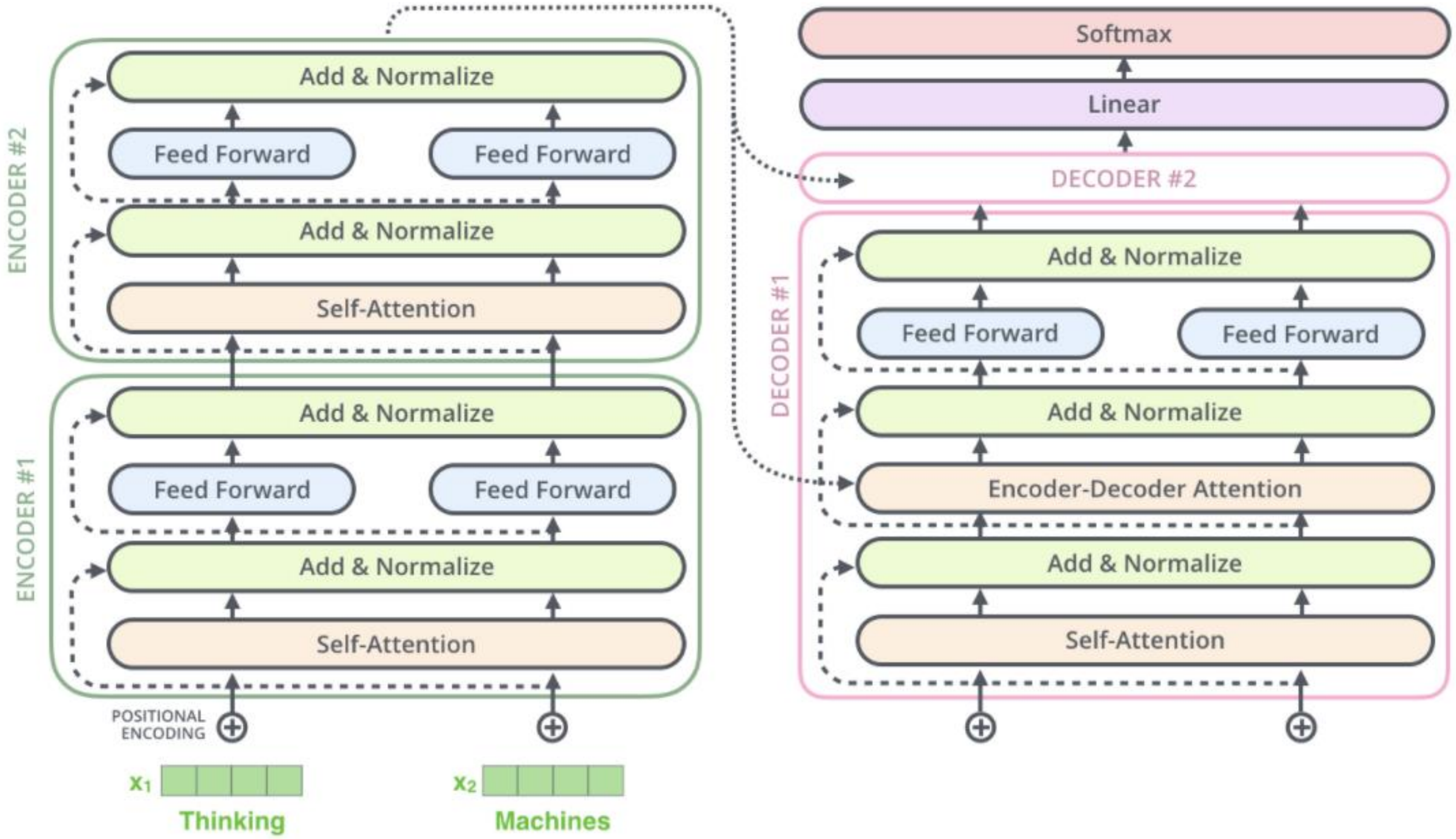
- Next let's look at the end. Similar to a CNN,
  - 1. Linear layer
  - 2. Softmax

  Get probabilities of words



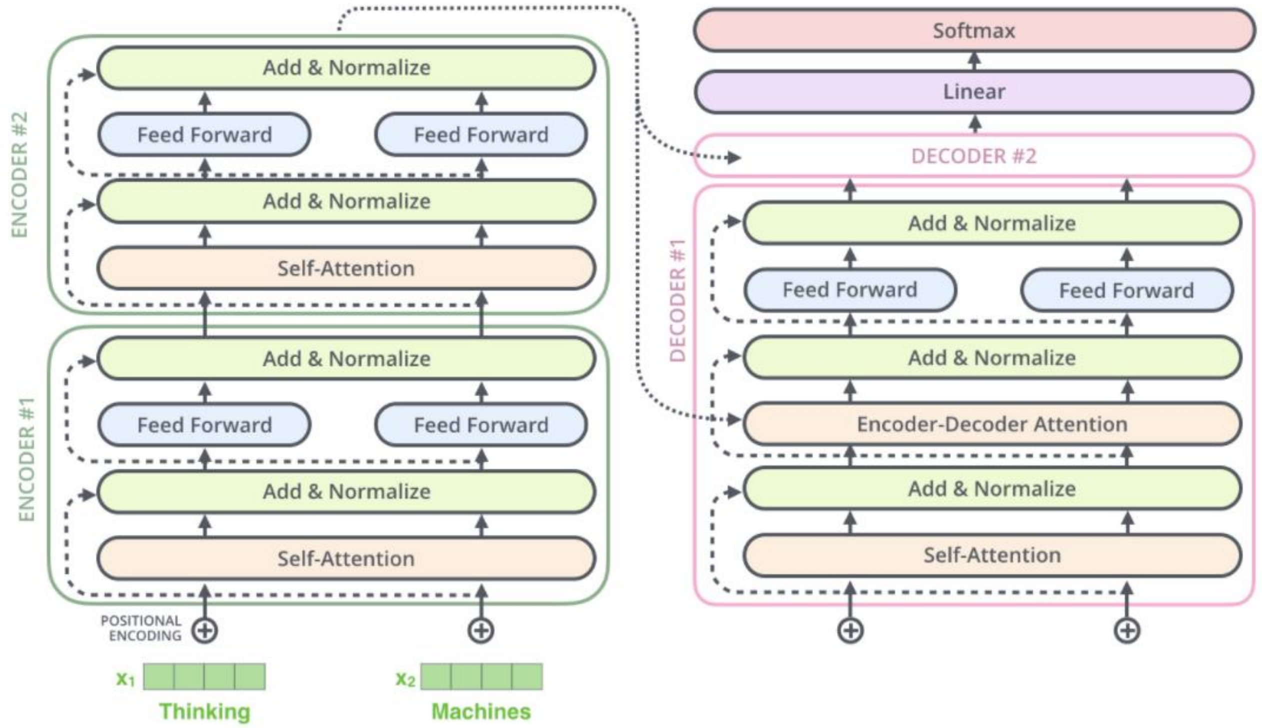
Transformers: Putting it All Together

- What does the full architecture look like?



Transformers: Training

- Data: standard datasets (WMT English-German)
  - Note: **supervised task**. Soon: switch to self-supervised
  - ~5 million pairs of sentences for this dataset
  - Training procedure not special: cross-entropy loss, Adam optimizer



Outline

- **Finish up last time: full Transformer architecture**
  - Encoder layer, decoder layer, full original Transformer architecture
- **Encoder-only models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Preview of decoder-only models**
  - GPT architecture, examples

Why Encoder-Decoder?

Wanted two things for translation:

- 1) **Outputs** in natural language
- 2) Tight alignment with **input**

What happens if we relax these?

1. Encoder-only models
2. Decoder-only models

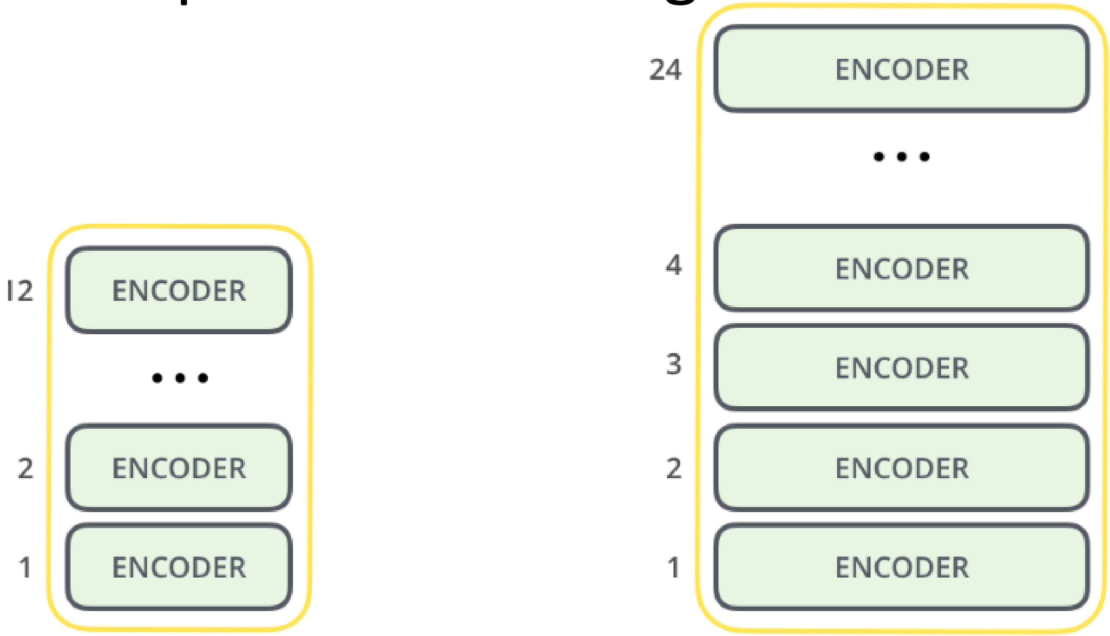
Encoder-Only Models: BERT

Let’s get rid of the first part

- 1) **Outputs** in natural language
- 2) Tight alignment with **input**

- So **not** a generative model → get representations
  - Like we talked about in self-supervised learning

- Rip away decoders
  - Just stack encoders



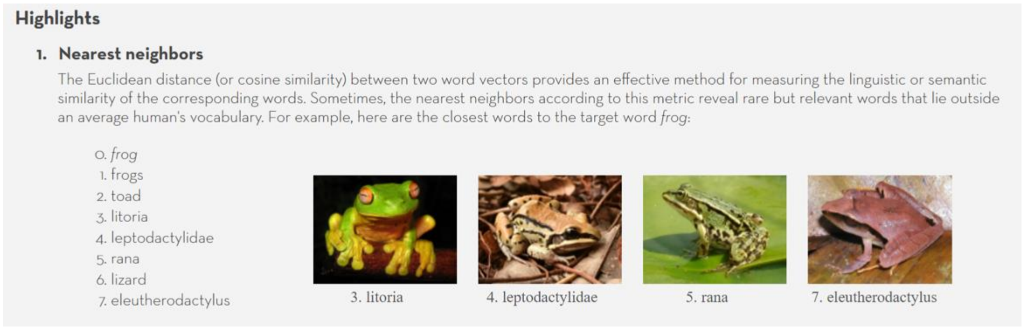
$\text{BERT}_{\text{BASE}}$ $\text{BERT}_{\text{LARGE}}$

Interlude: Contextual Embeddings

Q: Why is it called "BERT"?

- **A**: In a sense, follows up ELMo

- Story:
  - **2013**: "Dense" word embeddings (**Word2Vec, Glove**)
  - Downside: fixed representations per word
    - "Bank": building or riverside?
  - Need: contextual representations
    - Using language model-like techniques
    - 2018: ELMo, BERT
    - ELMo: uses LSTMs, BERT uses transformers



Highlights

1. **Nearest neighbors**

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

3. litoria 4. leptodactylidae 5. rana 7. eleutherodactylus

https://nlp.stanford.edu/projects/glove/

Interlude: Contextual Embeddings

Q: Why is it called “BERT”?

- **A**: In a sense, follows up ELMo

BERT acronym:

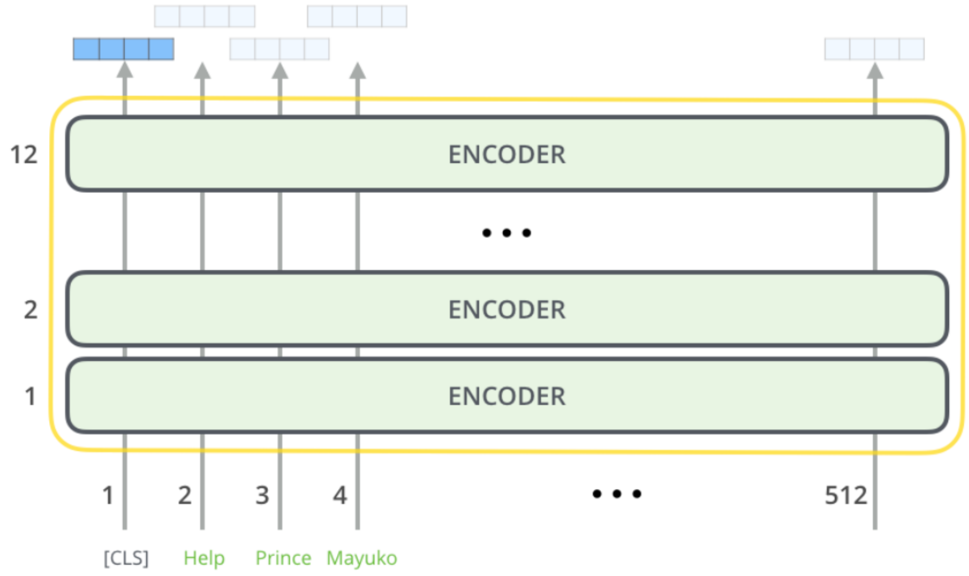
- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.
- ERT should make sense,

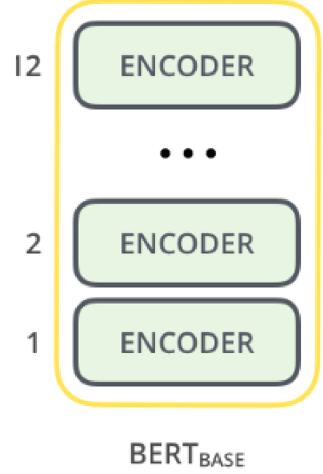
- Bidirectional: no causal masks, look at both sides of a word!
- Captured in self-attention block

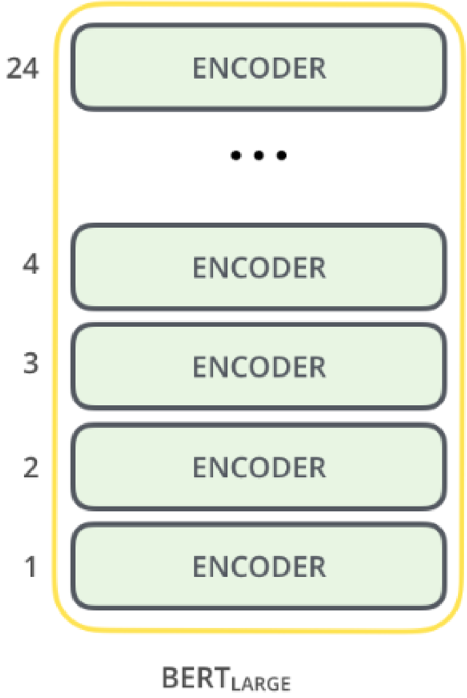
BERT: Forward Pass

BERT architecture

- Rip away decoders
  - Just stack encoders





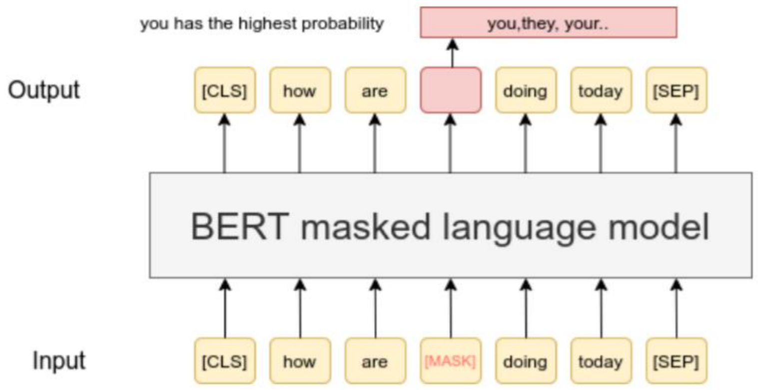


BERT: Training

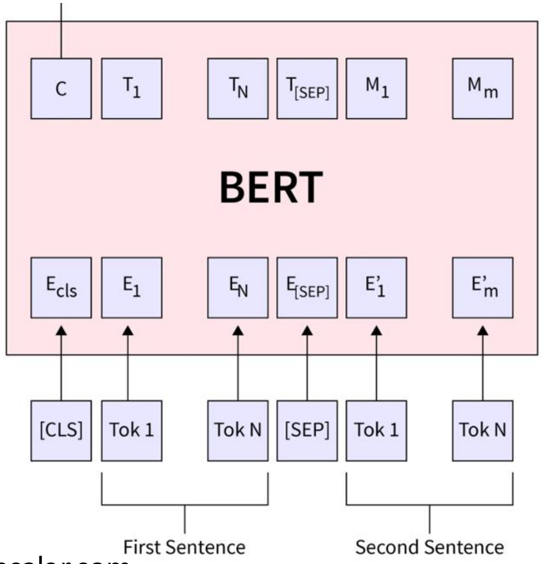
Training is more interesting!

- Pretraining. Then fine-tuning on task of interest

- Back to **self-supervised learning!**
- Two tasks for **pretraining**.



sbert.net



scaler.com

1. Masked Language Modeling

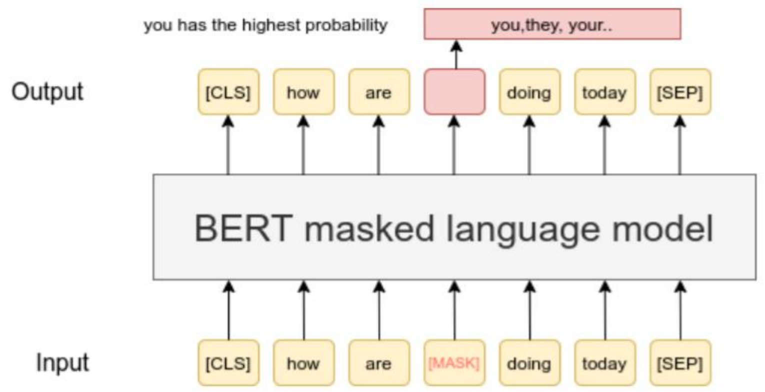
2. Next Sentence Prediction

BERT: Training Task 1

Masked Language Modeling Task

- Use [MASK] token for word to be predicted

- Which words to mask?
  - Original paper: 15% of words at random
  - But… of these
    - 10% of the time, no [MASK], flip word randomly
    - 10% of the time leave word unchanged

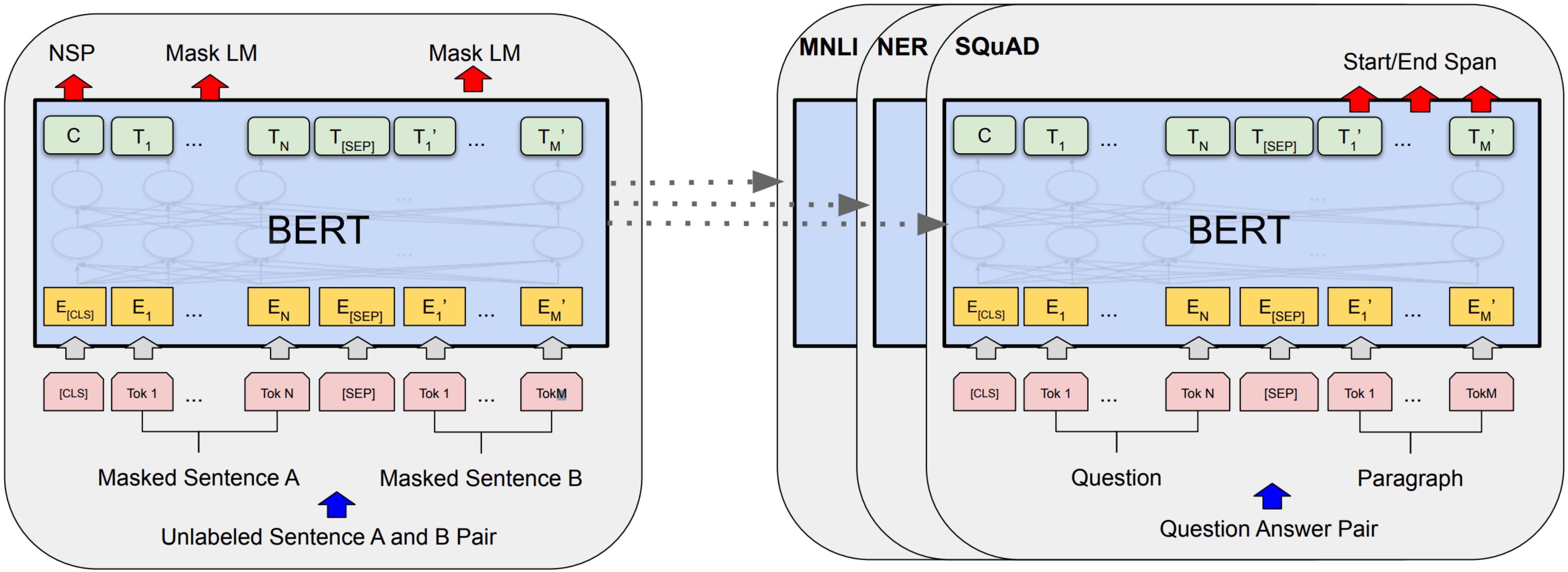


sbert.net

BERT: Training

Training is more interesting,

- Pretraining. Then fine-tuning on task of interest



Pre-training

Fine-Tuning

Devlin et al

Outline

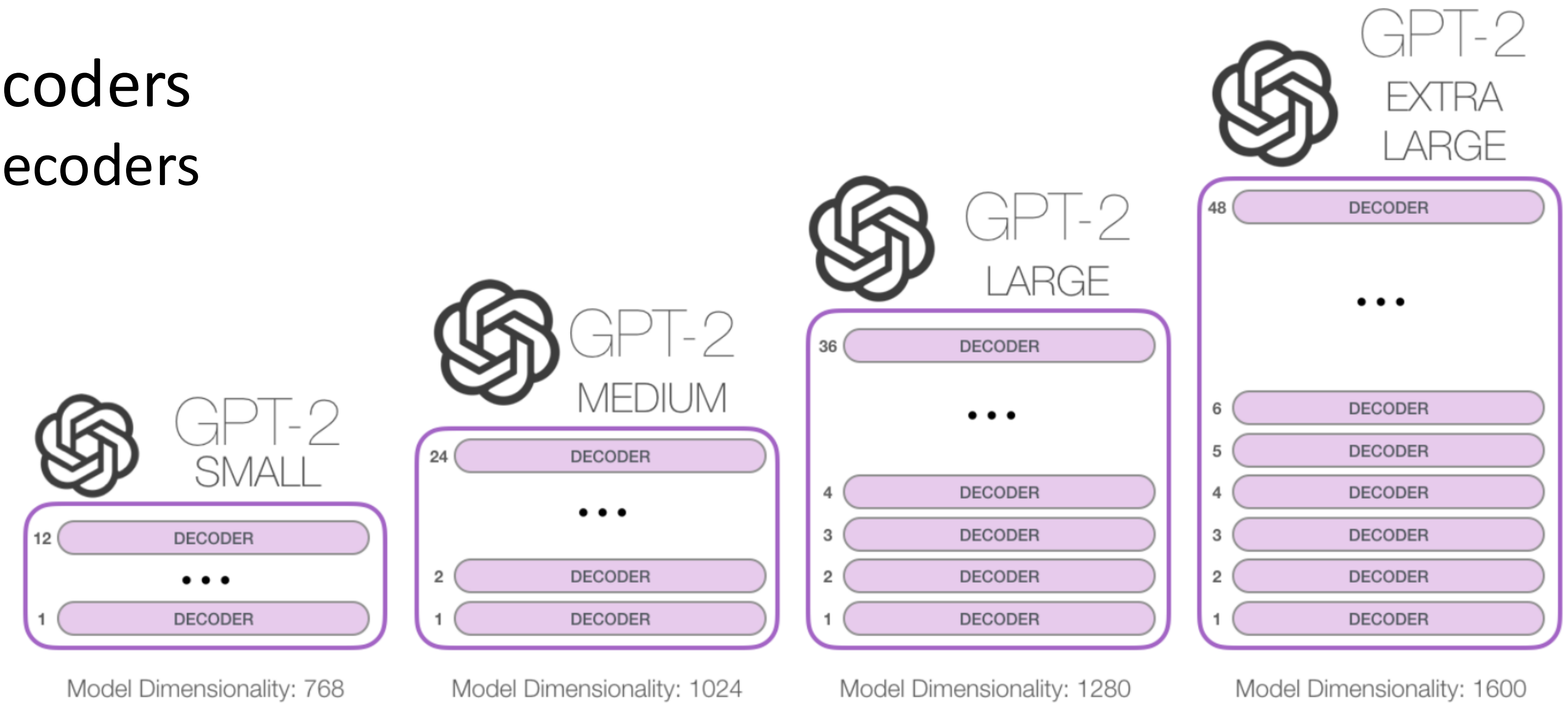
- **Finish up last time: full Transformer architecture**
  - Encoder layer, decoder layer, full original Transformer architecture
- **Encoder-only models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Preview of decoder-only models**
  - GPT architecture, examples

Decoder-Only Models: GPT

Let's get rid of the first part

- 1) **Outputs** in natural language
- 2) Tight alignment with **input**

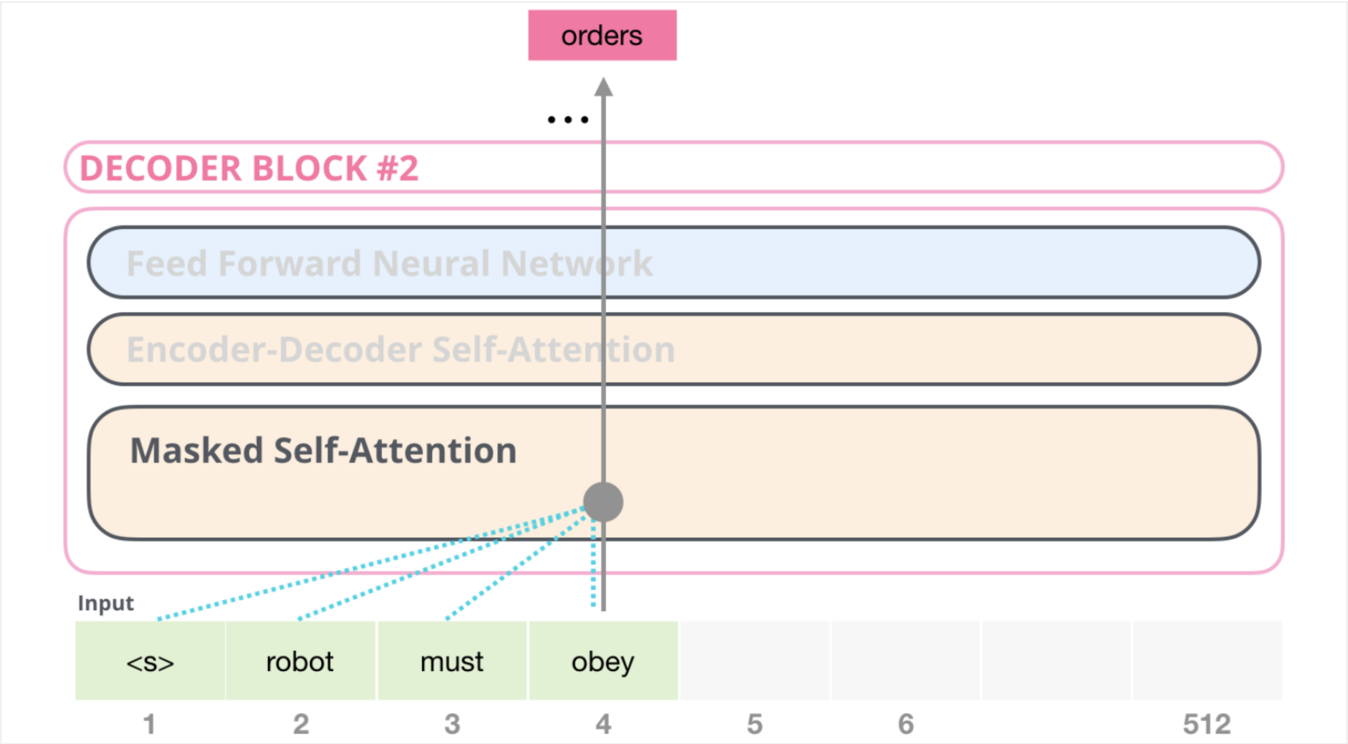
- Rip away encoders
  - Just stack decoders

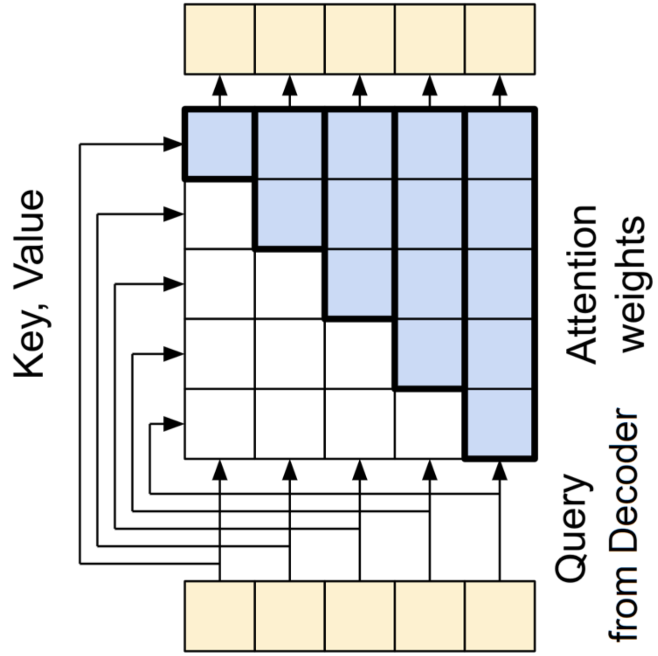


Decoder-Only Models: GPT

Rip away encoders

- Just stack decoders
- Use causal masking! NB: not a *mask token* like in BERT
- Training: **next-token prediction**





PyLessons