CS 760: Machine Learning
Large Language Models

Fred Sala

University of Wisconsin-Madison

Dec. 9, 2021

Announcements

- **Logistics**:
  - HW 8 / Project Due Dec. 14
  - Final: Dec. 20th
  - **Surveys** --- thank you!

- Class roadmap:

Thurs., Dec 9	**Large Language Models**
Tues., Dec 14	Fairness & Ethics
Monday, Dec 20	**Final Exam**

Training Human Intelligence Updates

- 6 months of training
  - ~100 billion neurons
  - Compare to today’s model parameter counts

Outline

- **Language Models & NLP**
  - k-gram models, RNN review, word embeddings, attention
- **Transformer Model**
  - Properties, architecture breakdown
- **Transformer-based Models**
  - BERT, GPTs, Foundation Models

Outline

- **Language Models & NLP**
  - k-gram models, RNN review, word embeddings, attention
- **Transformer Model**
  - Properties, architecture breakdown
- **Transformer-based Models**
  - BERT, GPTs, Foundation Models

Language Models: History

- Pre-date neural networks
- Basic idea: use probabilistic models to **assign a probability to a sentence**

$$P(W) = P(w_1, w_2, \ldots, w_n) \text{ or } P(w_{\text{next}} | w_1, w_2 \ldots)$$

- Goes back to Shannon
  - Information theory: letters

Zero-order approximation	XFOML RXKHRJFFJUJ ALPWXFWJXYJ FFJEYVJCQSGHYD QPAAMKBZAACIBZLKJQD
First-order approximation	OCRO HLO RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL
Second-order approximation	ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE
Third-order approximation	IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE
First-order word approximation	REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE

Language Models: History

- Classic approach: make Markov-type assumptions:

$$P(w_i|w_{i-1}w_{i-2}\ldots w_1) = P(w_i|w_{i-1}w_{i-2}\ldots w_{i-k})$$

- Ex: k=1: Bigram model

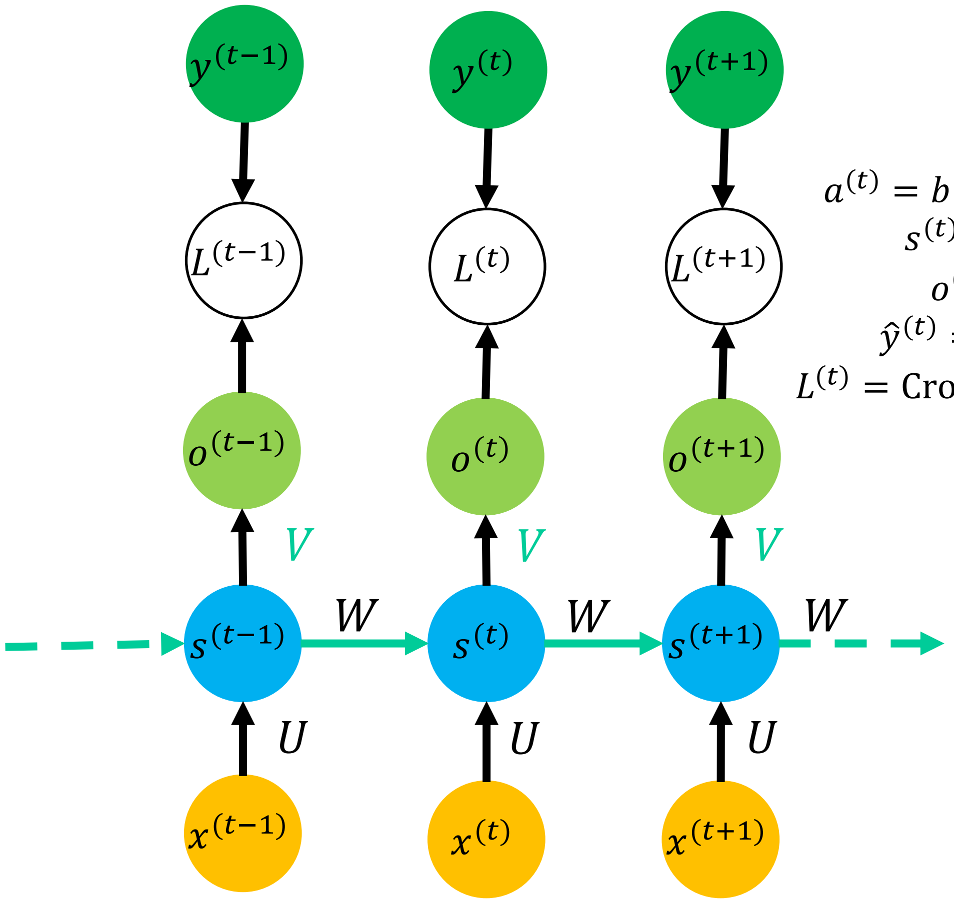
$$P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|w_{n-1})$$

- Example:

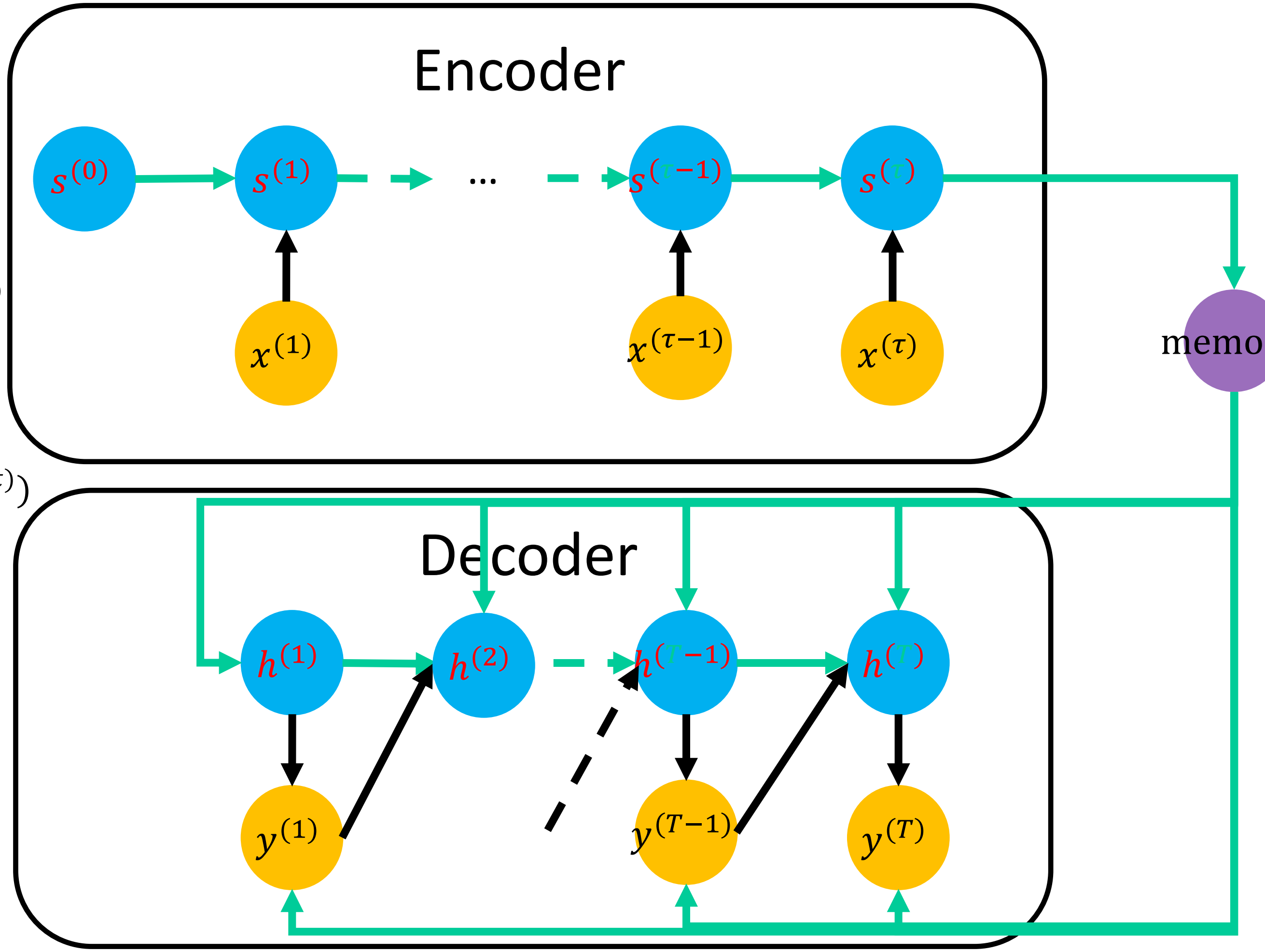
```
texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler,
house, said, mr., gurria, mexico, 's, motion, control, proposal, without,
permission, from, five, hundred, fifty, five, yen outside, new, car, parking,
lot, of, the, agreement, reached this, would, be, a, record, november
```

Language Models: RNN Review

- Classical RNN model / Encoder-Decoder variant:

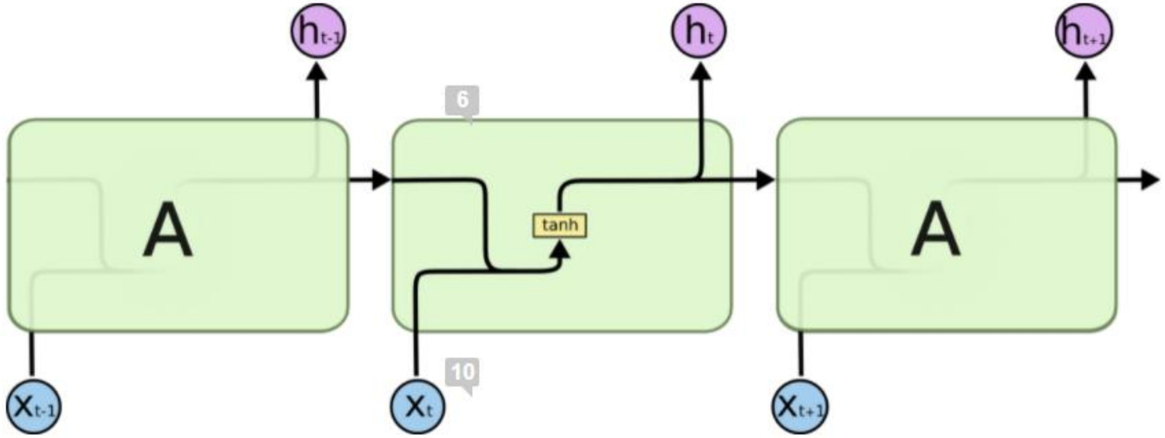


$$a^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$$
$$s^{(t)} = \tanh\left(a^{(t)}\right)$$
$$o^{(t)} = c + Vs^{(t)}$$
$$\hat{y}^{(t)} = \text{softmax}\left(o^{(t)}\right)$$
$$L^{(t)} = \text{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})$$

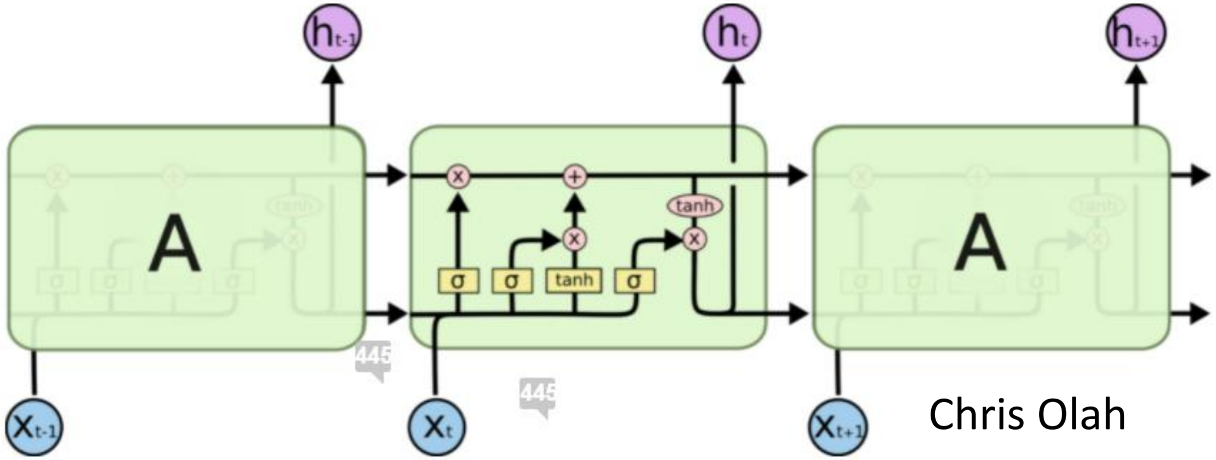


Language Models: LSTM Review

- RNN: can write structure as:



- Long Short-Term Memory: deals with problem. Cell:



Chris Olah

Language Models: Word Embeddings

- One way to encode words: one-hot vectors
  - Want something smarter…

Distributional semantics: account for relationships

- Representations should be close/similar to other words that appear in a similar context

Dense vectors:

$$\text{dog} = \begin{bmatrix}0.13 & 0.87 & -0.23 & 0.46 & 0.87 & -0.31\end{bmatrix}^T$$

$$\text{cat} = \begin{bmatrix}0.07 & 1.03 & -0.43 & -0.21 & 1.11 & -0.34\end{bmatrix}^T$$

AKA **word embeddings**

Training **Word Embeddings**

Many approaches (super popular 2010-present)

- Word2vec: a famous approach
- What's our likelihood?

Windows of length 2a

$$L(\theta) = \prod_{t=1}^{T} \prod_{-a \leq j \leq a} P(w_{t+j} | w_t, \theta)$$

Our word vectors (weights)

All positions

Training **Word Embeddings**

Word2vec likelihood
$$L(\theta) = \prod_{t=1}^{T} \prod_{-a \leq j \leq a} P(w_{t+j} | w_t, \theta)$$

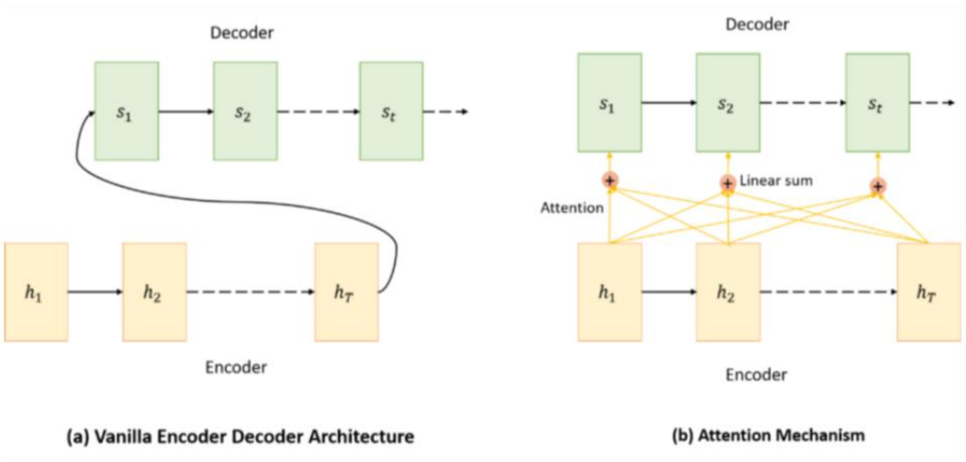
- Maximize this; what's the probability?
  - Two vectors per word. v_w, u_w for center/context (o is context word, c is center)

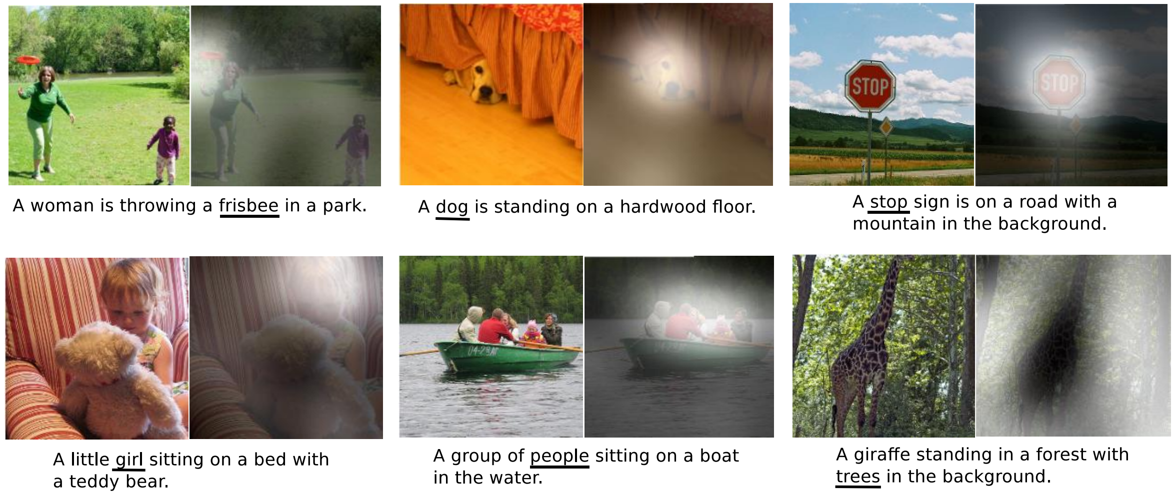
Softmax

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Language Models: Attention

- One challenge: dealing with the hidden state
  - Everything gets compressed there
  - Might lose information
- Solution: **attention** mechanism
  - Tells us where to focus. Note: usable in other models too!

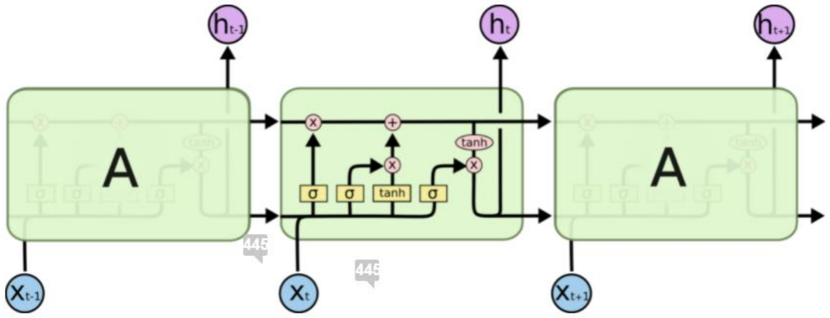




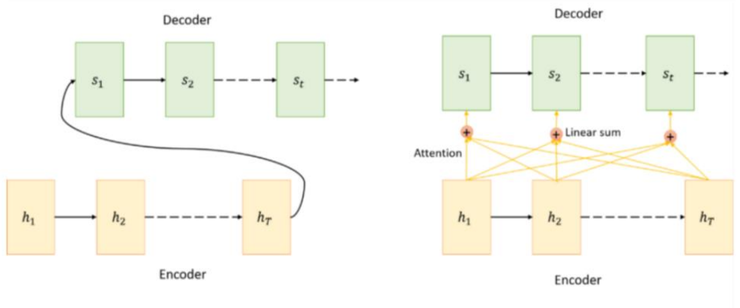
Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Language Models: Putting it All Together

- Before mid-2017: best language models
  - Use encoder/decoder architectures based on **RNNs**
  - Use **word embeddings** for word representations
  - Use attention mechanisms



$$\text{dog} = \begin{bmatrix} 0.13 & 0.87 & -0.23 & 0.46 & 0.87 & -0.31 \end{bmatrix}^T$$



(a) Vanilla Encoder Decoder Architecture

(b) Attention Mechanism

Break & Quiz

Outline

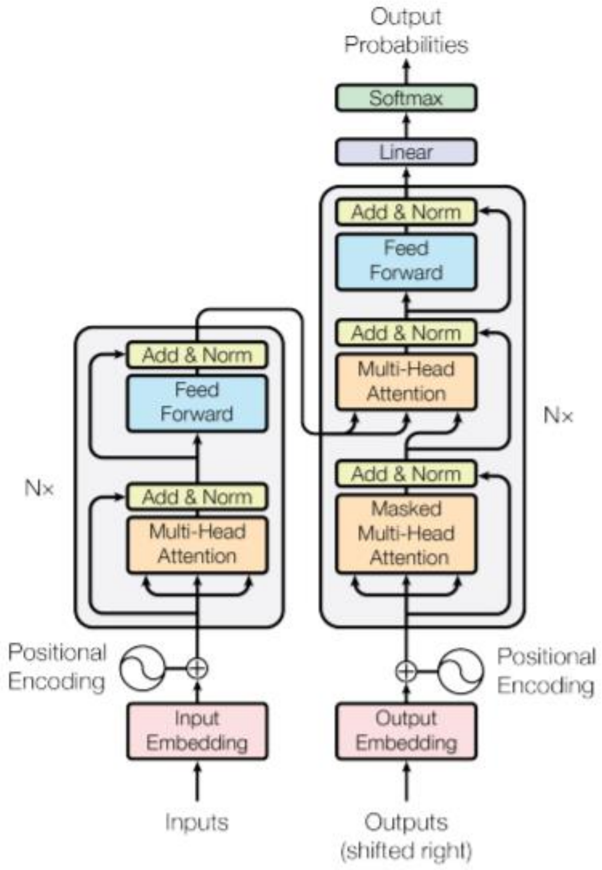
- **Language Models & NLP**
  - k-gram models, RNN review, word embeddings, attention
- **Transformer Model**
  - Properties, architecture breakdown
- **Transformer-based Models**
  - BERT, GPTs, Foundation Models

Transformers: Idea

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

- Architecture
  - We’ll cover this step-by-step

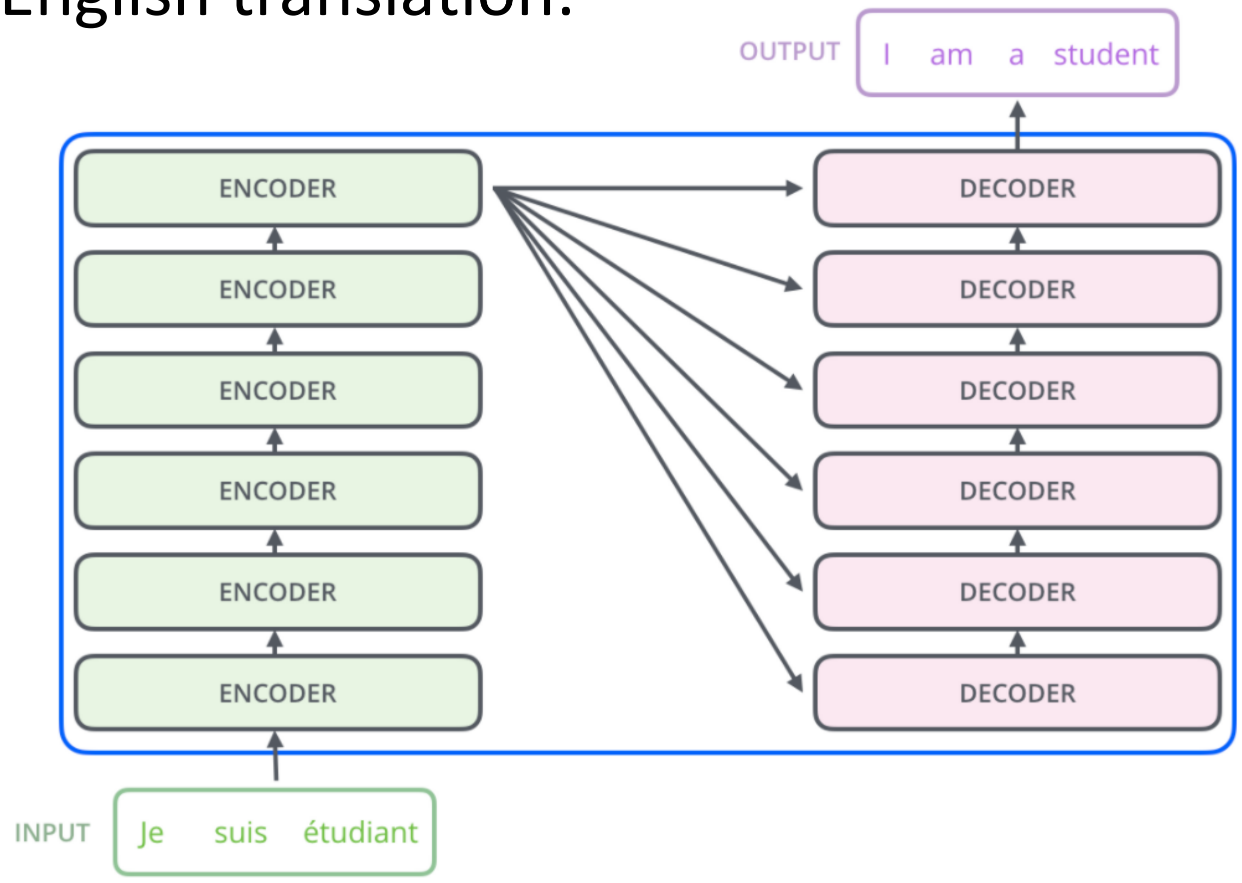
- Results:
  - Best results on translation tasks.



Vaswani et al. ‘17

Transformers: Architecture

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

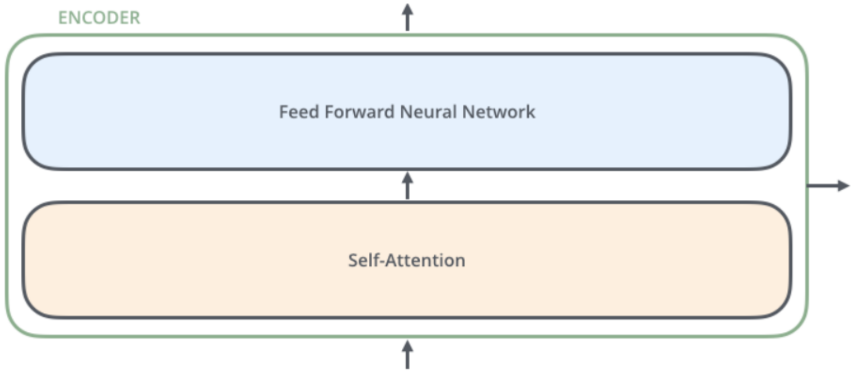


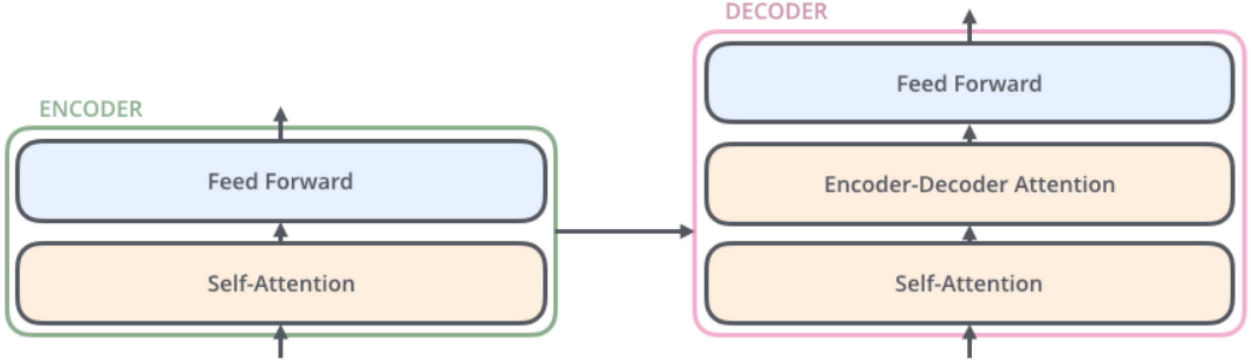
Note: All visualizations are due to Jay Alammar

Excellent resource: https://jalammar.github.io/illustrated-transformer/

Transformers: Architecture

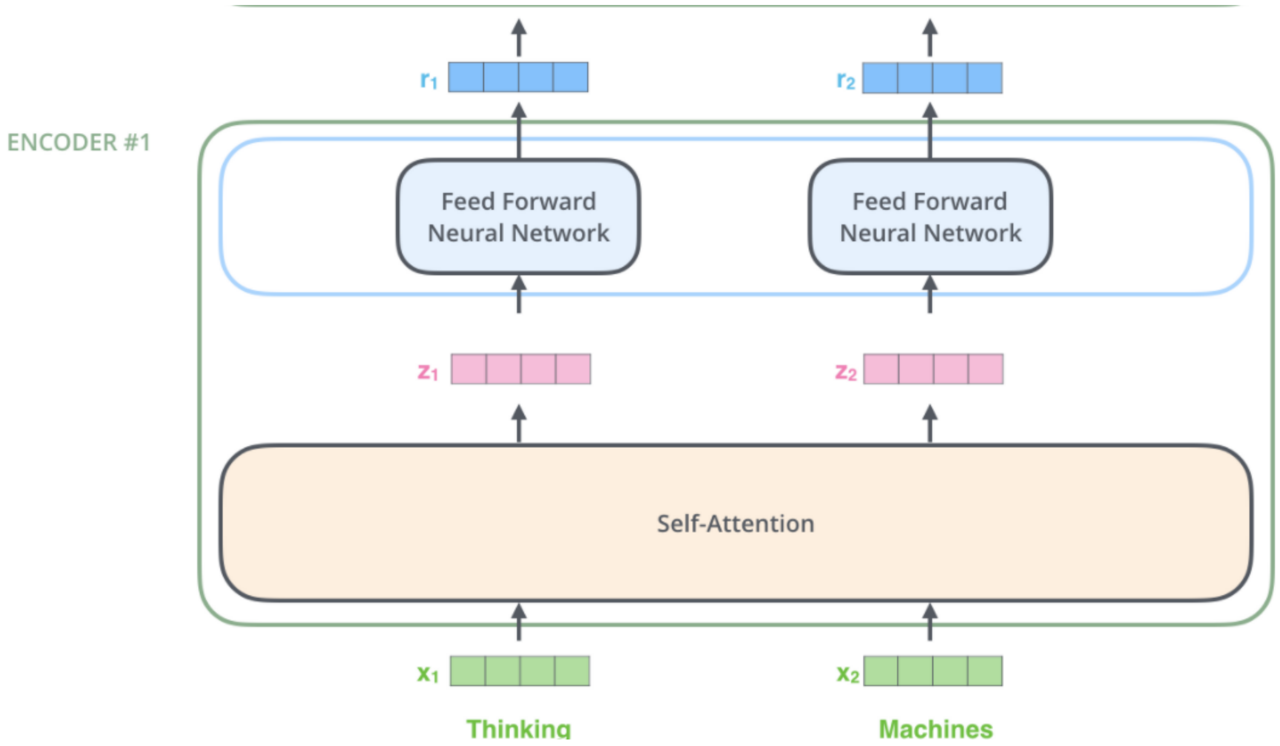
- Sequence-sequence model with **stacked** encoders/decoders:
    - What’s inside each encoder/decoder unit?





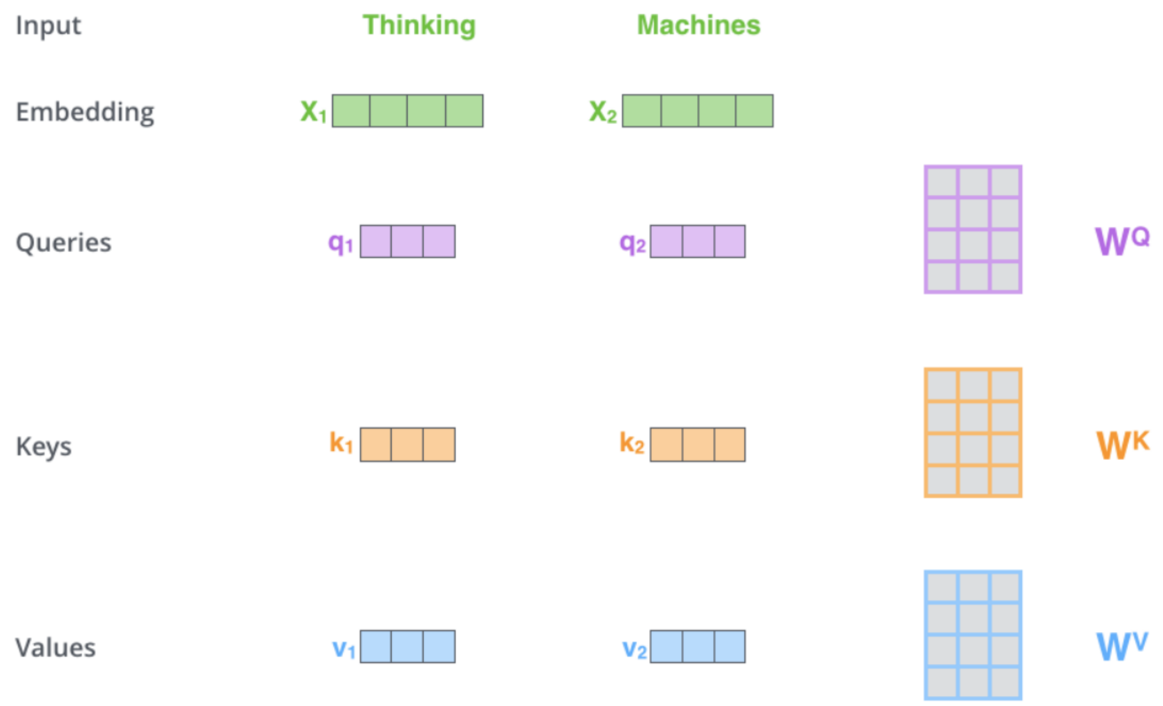
Transformers: Insider an Encoder

- Let’s take a look at the encoder. Two components:
  - 1. **Self-attention** layer
  - 2. “Independent” **feedforward nets**



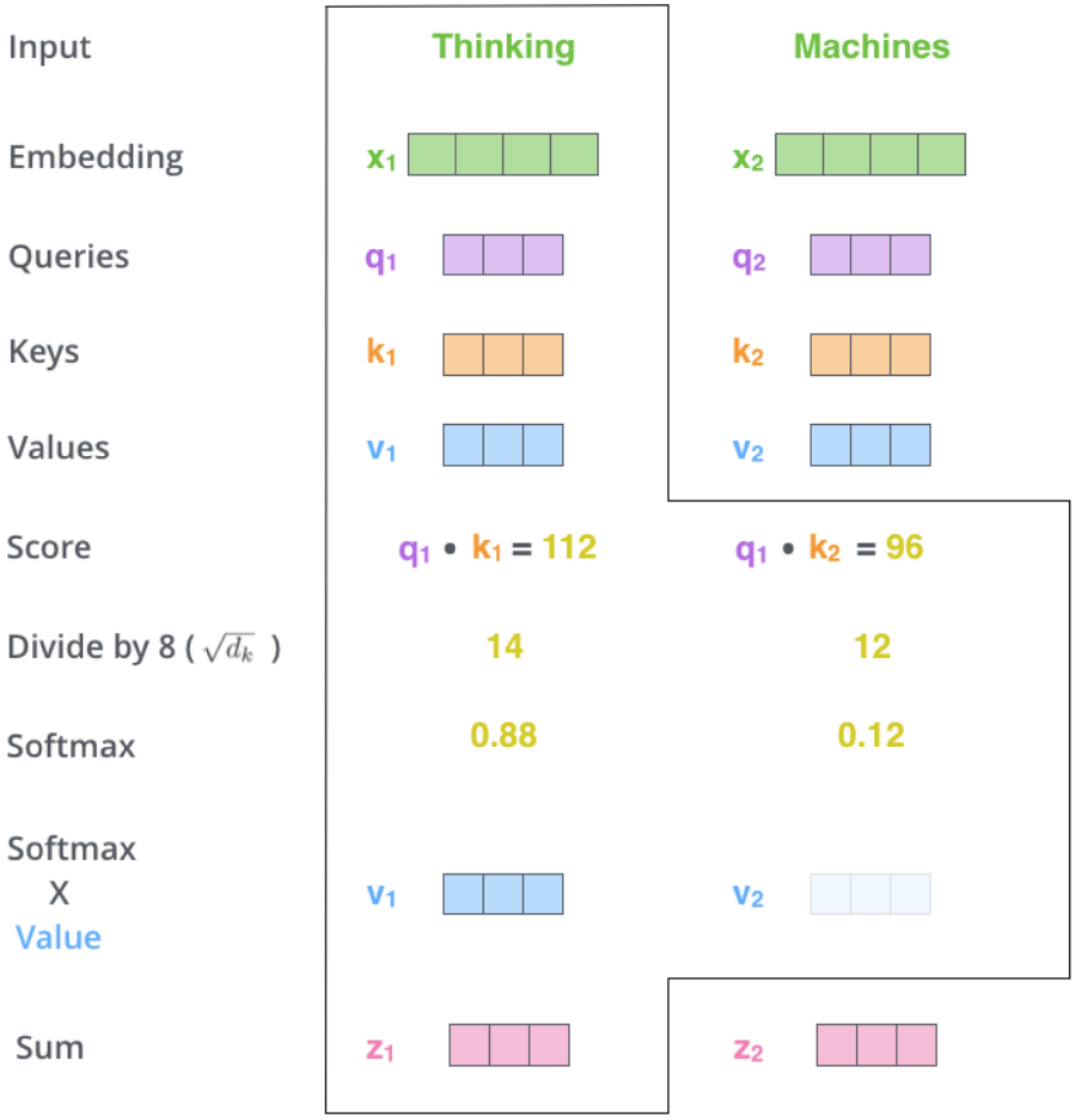
Transformers: Self-Attention

- Self-attention is the key layer in a transformer stack
  - Get 3 vectors for each embedding: **Query**, **Key**, **Value**
  - We'll combine query at i with each key j (j=1,…), and run softmax
  - Compute values vectors by adding weights (from softmax)



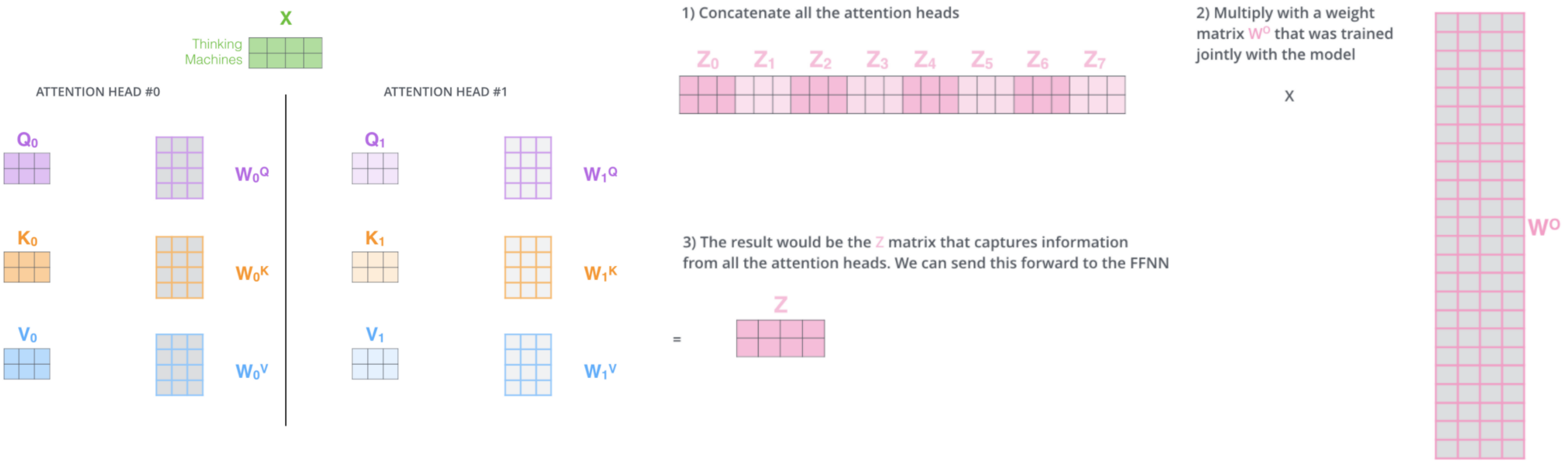
Transformers: Self-Attention

- Self-attention is the key layer in a transformer stack
  - Illustration. Recall the three vectors for each embedding: **Query**, **Key**, **Value**
  - The sum values are the outputs for this layer
  - Send these to feedforward NNs
    - Note: matrix versions in practice



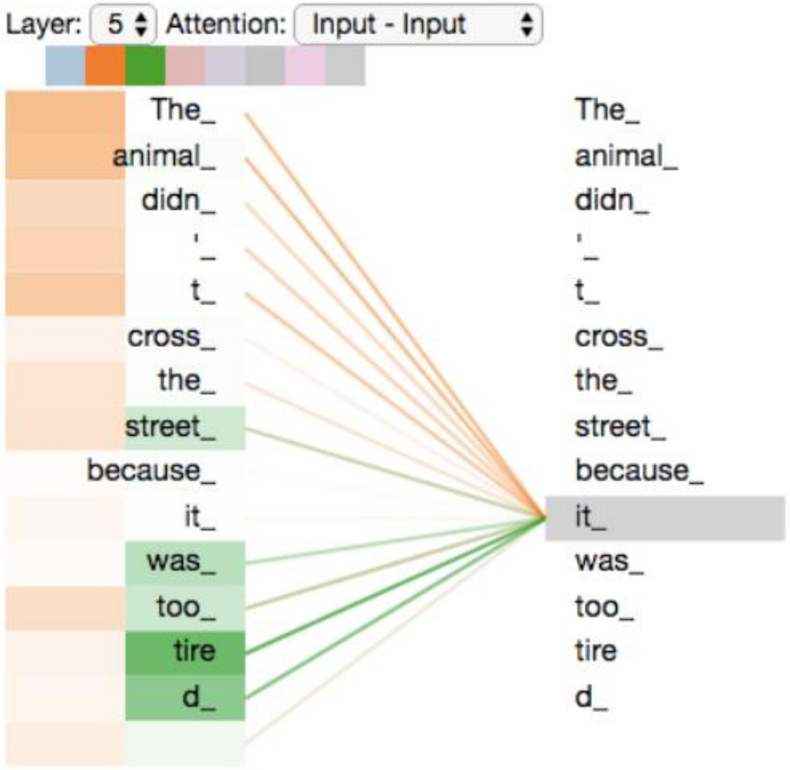
Transformers: Multi-Headed Attention

- We can do this multiple times in parallel
  - Called multiple heads
  - Need to combine the resulting output sums



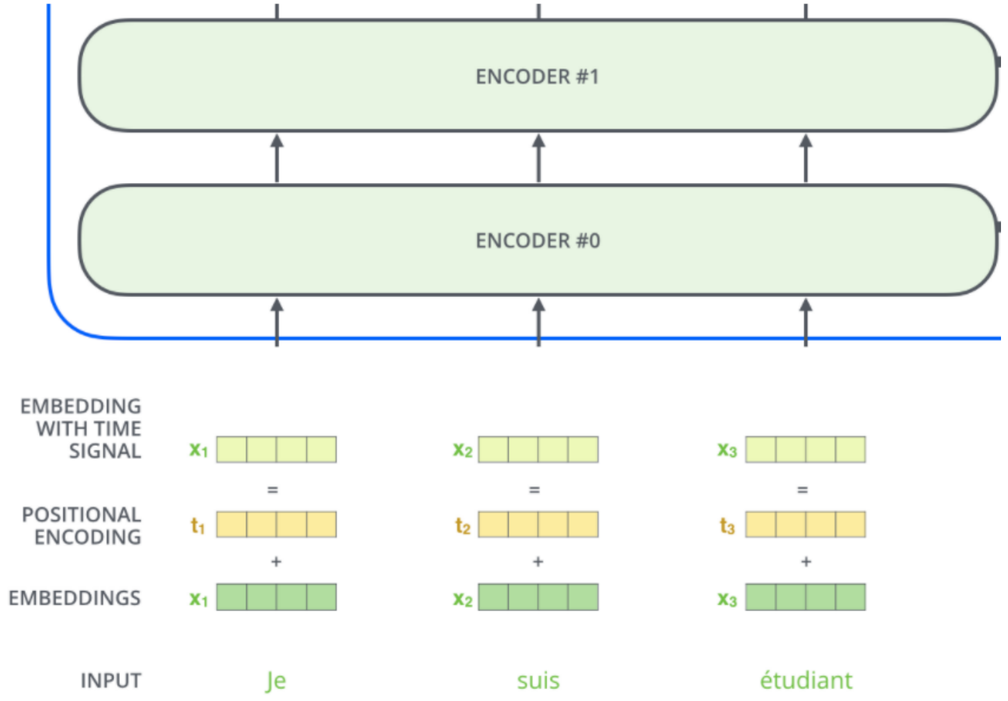
Transformers: Attention Visualization

- Attention tells us where to focus the information
  - Illustration for a sentence:



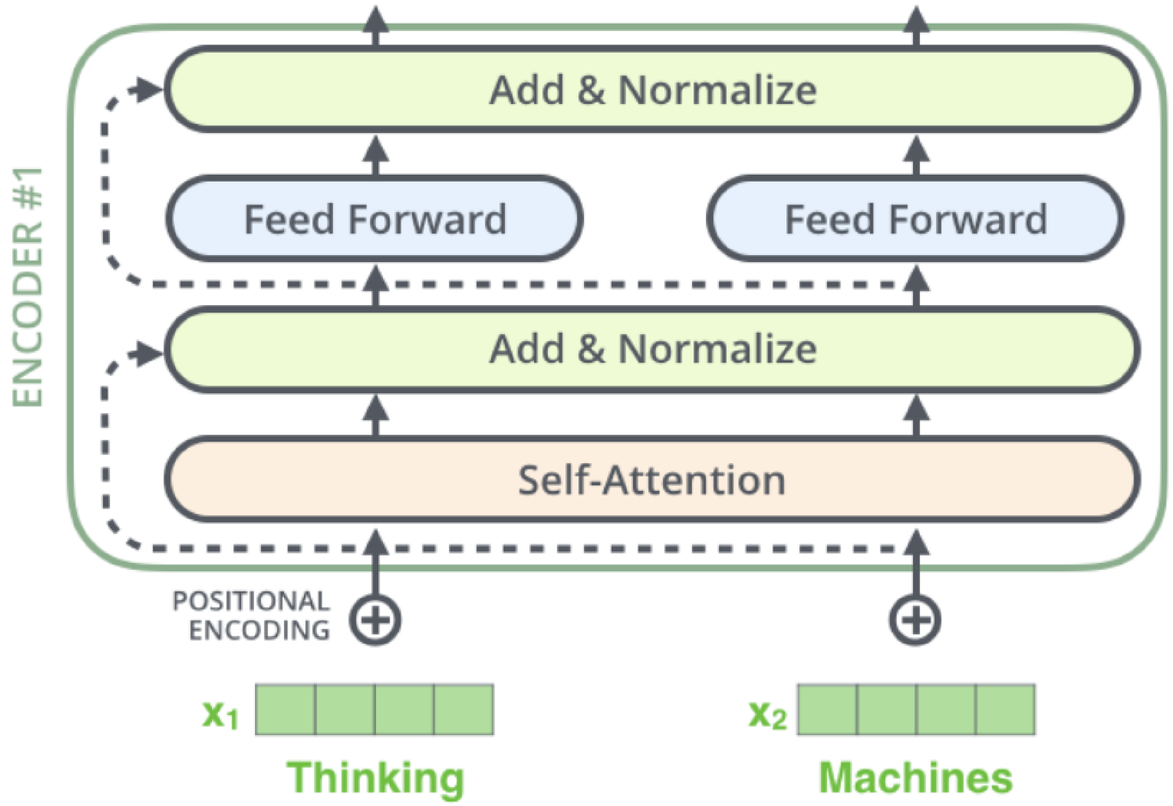
Transformers: Positional Encodings

- One thing we haven't discussed: the order of the symbols/elements in the sequence
  - Clearly important. **Q**: How to include?
  - **A**: add a vector containing a special positional formula's embedding



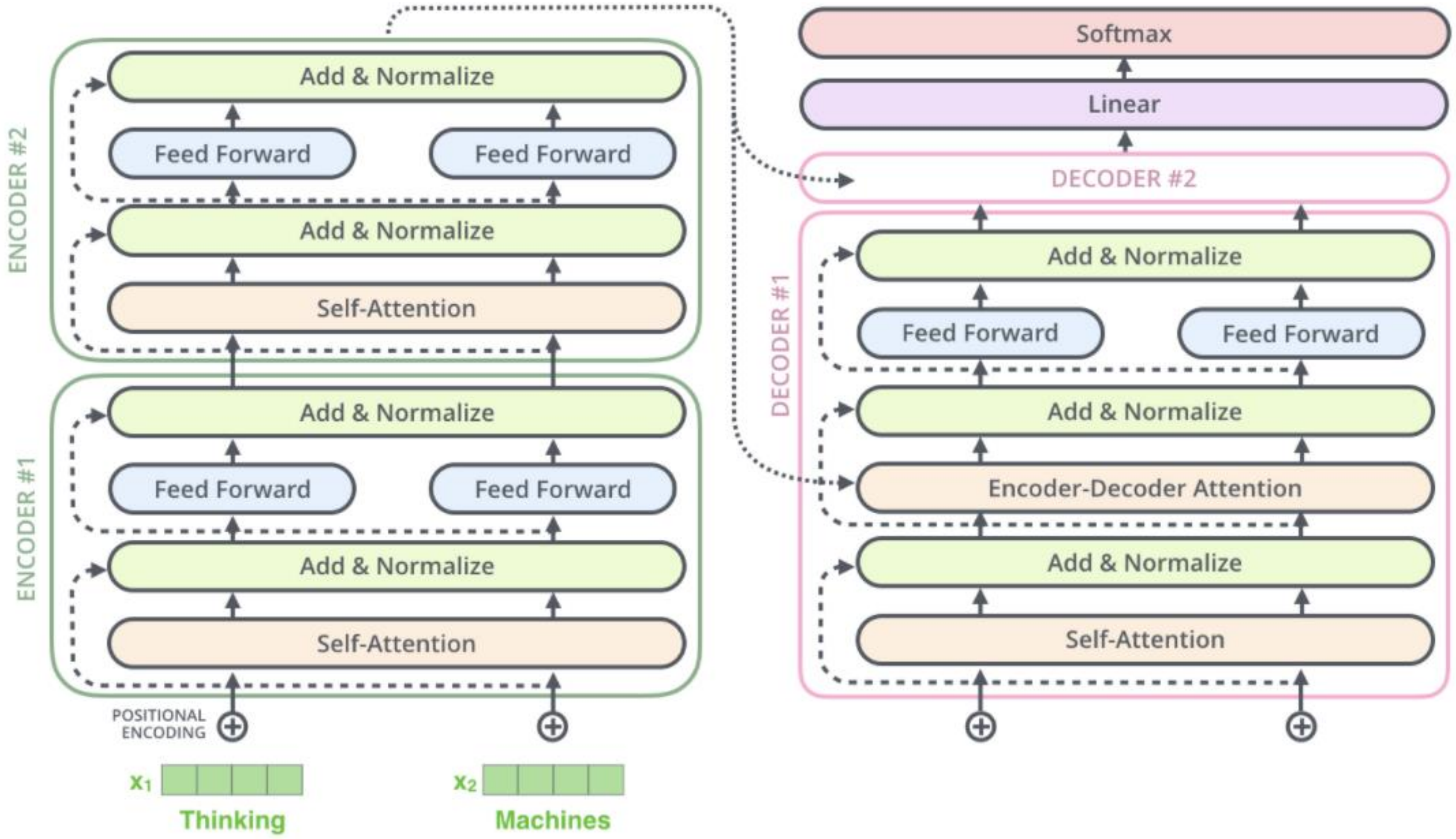
Transformers: More Tricks

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



Transformers: Putting it All Together

- What does the full architecture look like?



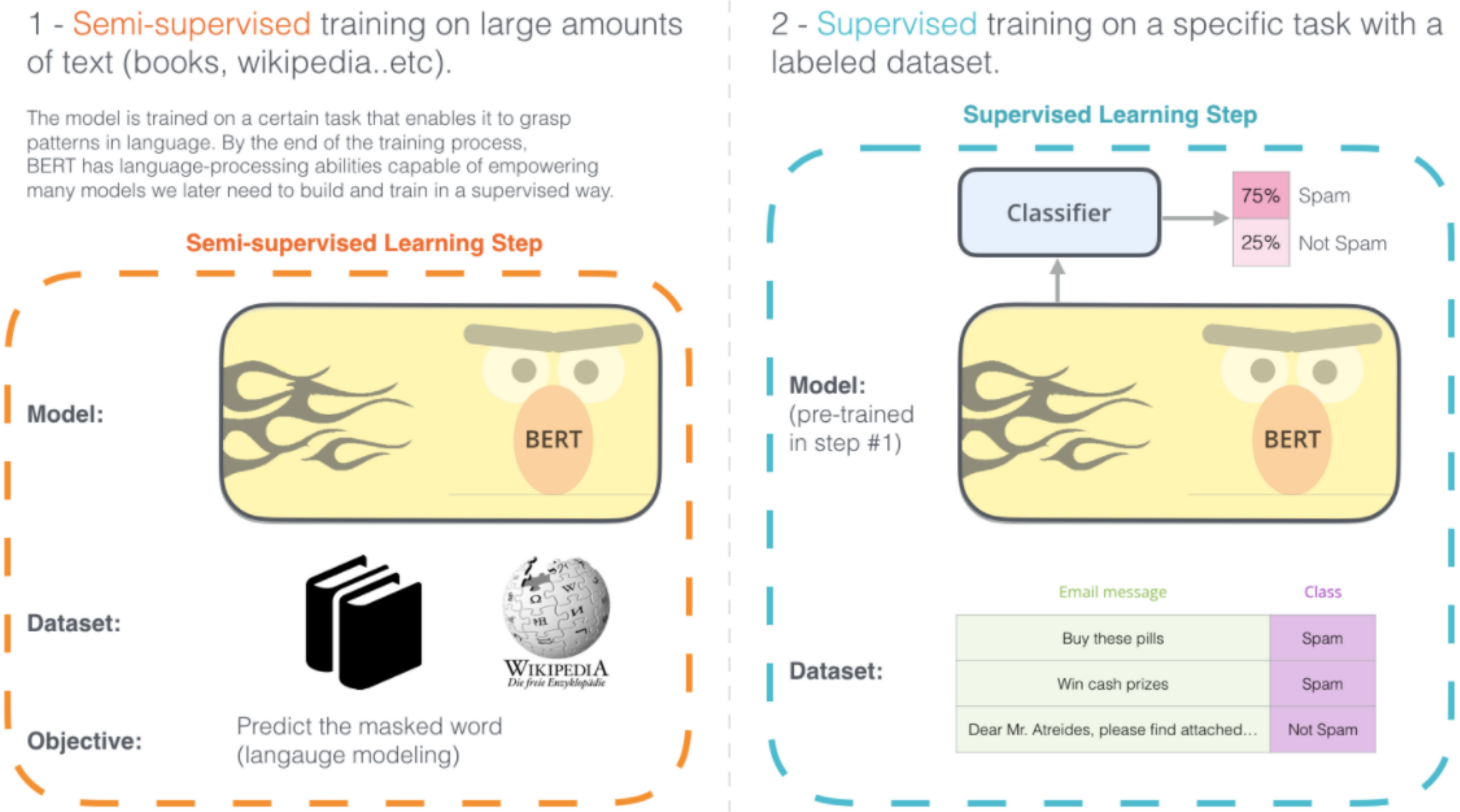
Break & Quiz

Outline

- **Language Models & NLP**
  - k-gram models, RNN review, word embeddings, attention
- **Transformer Model**
  - Properties, architecture breakdown
- **Transformer-based Models**
  - BERT, GPTs, Foundation Models

Transformer-Based Models: **BERT**

- The transformer model works very well…
  - Why not apply it to **everything** in NLP?
  - BERT does this. Two-step process to use for a particular task



BERT: Concepts

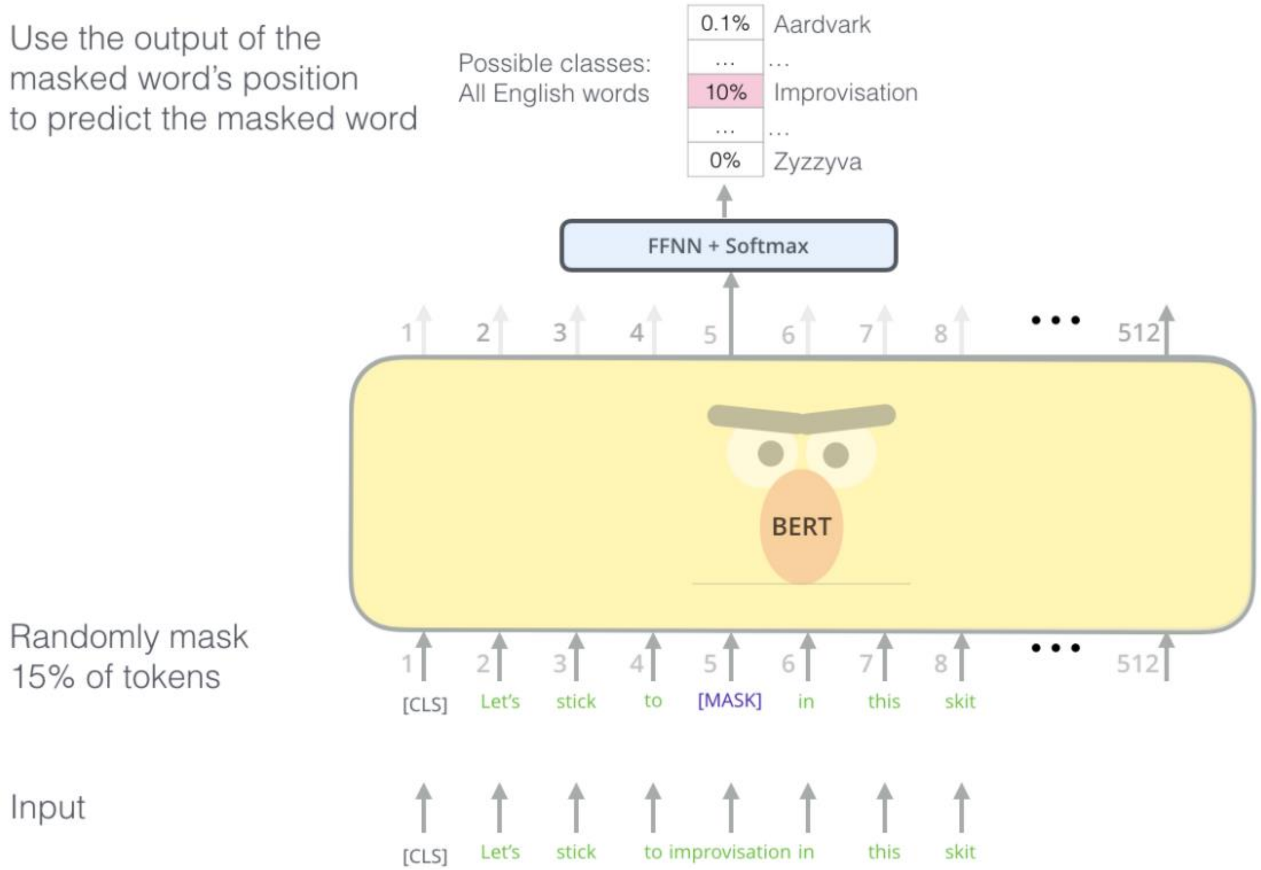
- What makes BERT work? A bunch of ideas:
  - 1. Use the **Transformer** architecture
    - Encoder stacks in particular (not doing sequence-sequence)
  - 2. **Pre-training** on corpora
    - Then fine-tune for a particular task
  - 3. **Scale**: BERT-Large has 340 million parameters

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\mathrm{BERT}_{\mathrm{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\mathrm{BERT}_{\mathrm{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Results: Devlin et al, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

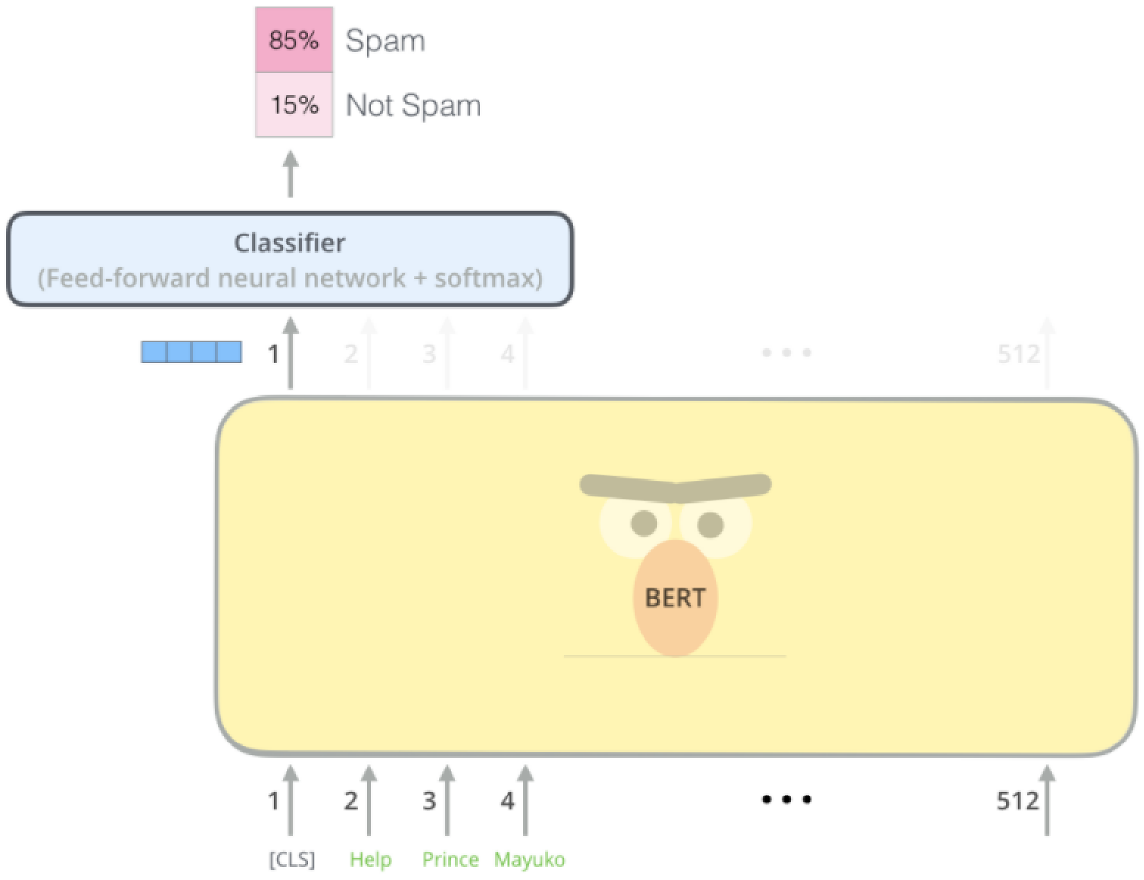
BERT: Training

- BERT is trained on a simple tasks on a huge amount of data:
  - Recall our pretext tasks in self-supervised learning
  - **Masked word prediction:**



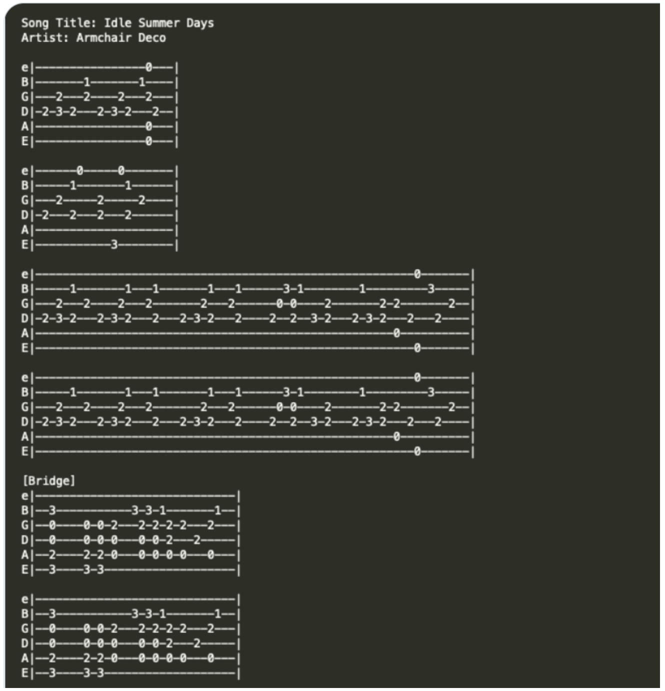
BERT: Classification

- Then, fine-tune on a particular task
  - Example: **binary classification**, spam VS not spam
  - Also use for word embeddings. Note: contextual, unlike word2Vec



GPT Series of Models

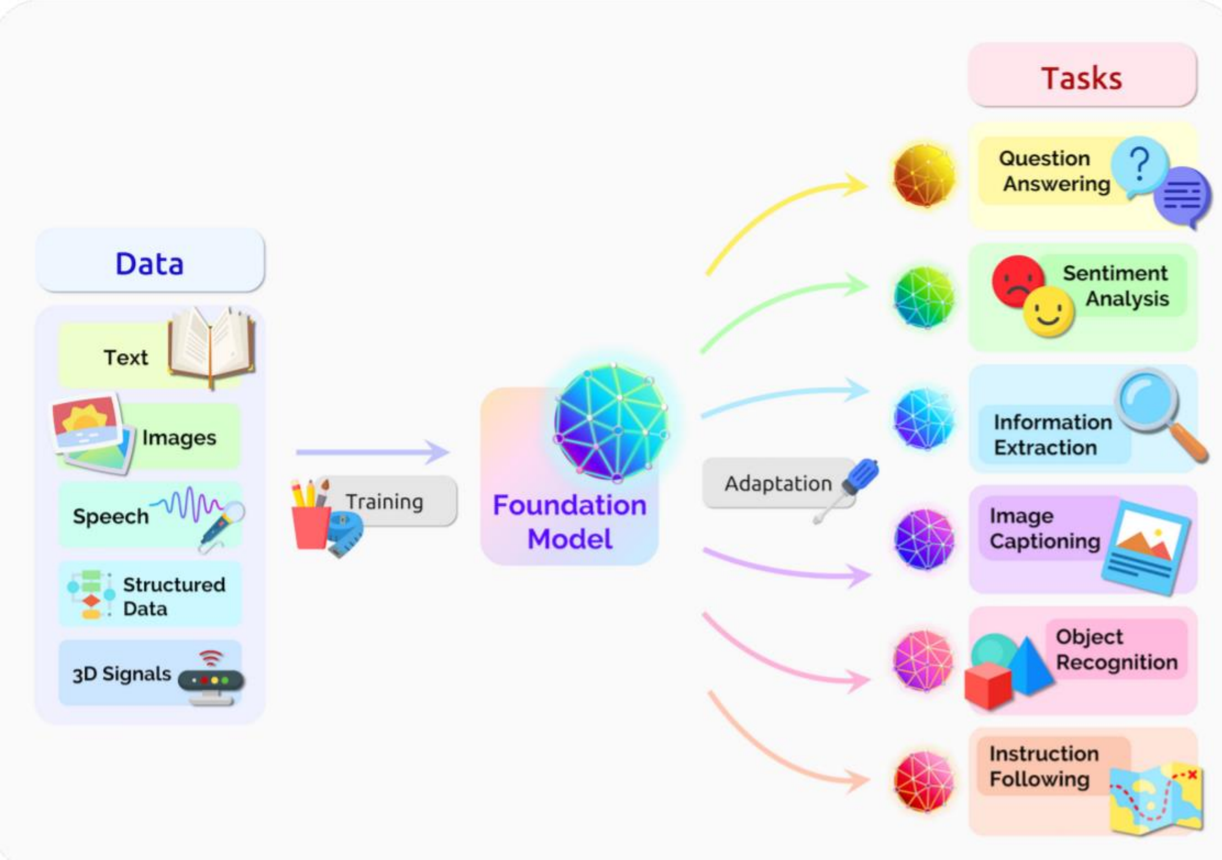
- **GPT**: **G**enerative **P**re-trained **T**ransformer
  - Also built on top of transformer model architecture
  - Essentially the decoder part only
- Goal: generate text (possibly from a **prompt**)
  - Not fine-tuning for a particular task.
- Scale: huge!
  - GPT-3: 175 billion parameters



Amanda Askell

Foundation Models

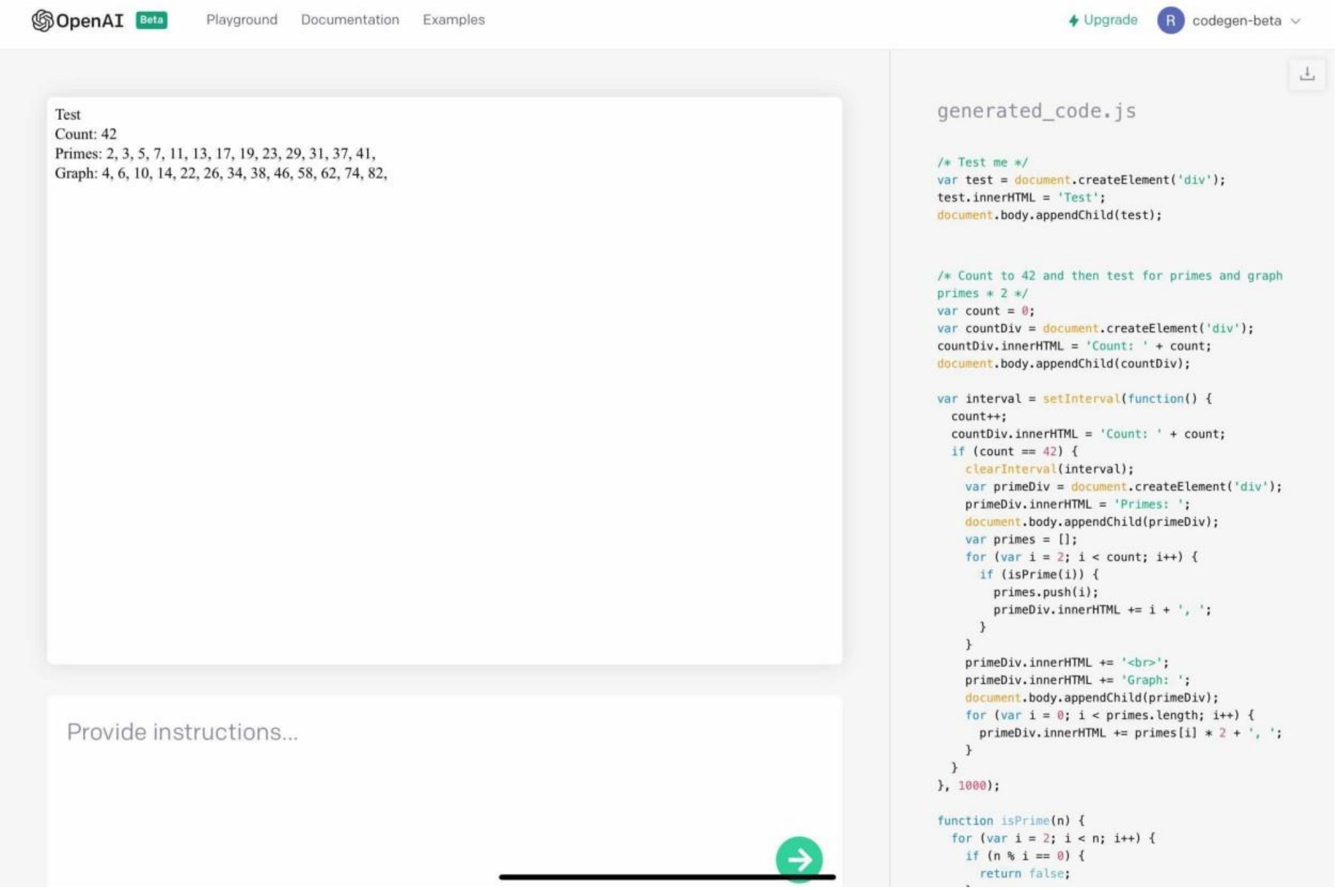
- Many more large scale models
  - Not just focused on text



Bommasani et al, “On the Opportunities and Risks of Foundation Models”

Codex

- Codex: a variant of GPT-3 based on source code
  - Outputs code. Ex: show primes



Russell Foltz-Smith

DALL-E

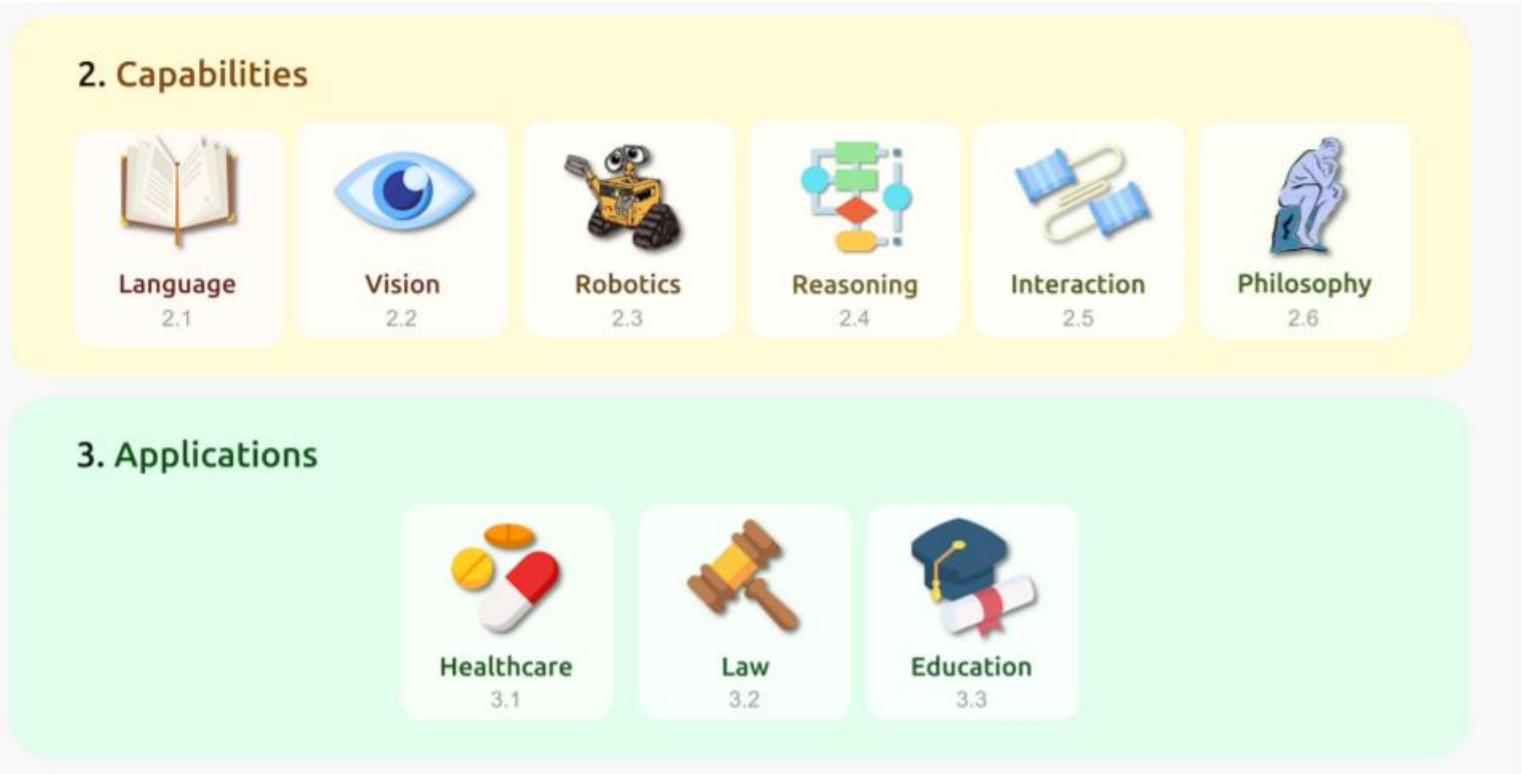
- Create images from text
  - Prompt: “an armchair in the shape of an avocado. . . .”

https://openai.com/blog/dall-e/

  - Note: several online demos. Try it yourself!

Conclusion

- "Foundation" models based on transformers and beyond
  - Huge, expensive to train, challenging in various ways… but
  - Remarkably powerful for a vast number of tasks.
  - The future of AI??



Bommasani et al, "On the Opportunities and Risks of Foundation Models"

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Jay Alammar