CS 839: Foundation Models
Models II

Fred Sala

University of Wisconsin-Madison

Sept. 21, 2023

Announcements

- **Logistics:**
  - Check out some of the posted papers!
  - Homework will start next week

- Class roadmap:

Thursday Sept. 21	Models II
Tuesday Sept. 23	Prompting I
Thursday Sept. 28	Prompting II
Tuesday Oct. 3	Reasoning & Chain-of-Thought
Thursday Oct. 5	In-Context Learning: Practice and Theory

Mostly Language Models

Outline

- **From Last Time: Encoder-only Models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Decoder-only Models**
  - Example: GPT, architecture, basic functionality
- **Variations and Advancements**
  - Scaling, upgrades to positional encodings, etc

Questions/Clarifications From Last Time

1. BERT initialization

- Random

2. **Differences** between decoders in **encoder-decoder models** and **decoder-only models**

- Gets rid of cross-attention, still use masked self-attention

3. **Weight-tying** input and output embeddings

- Trick to help performance (better usage of information, # parameters)

Outline

- **From Last Time: Encoder-only Models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Decoder-only Models**
  - Example: GPT, architecture, basic functionality
- **Variations and Advancements**
  - Scaling, upgrades to positional encodings, etc

Why Encoder-Decoder?

Wanted two things for translation:

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

What happens if we relax these?

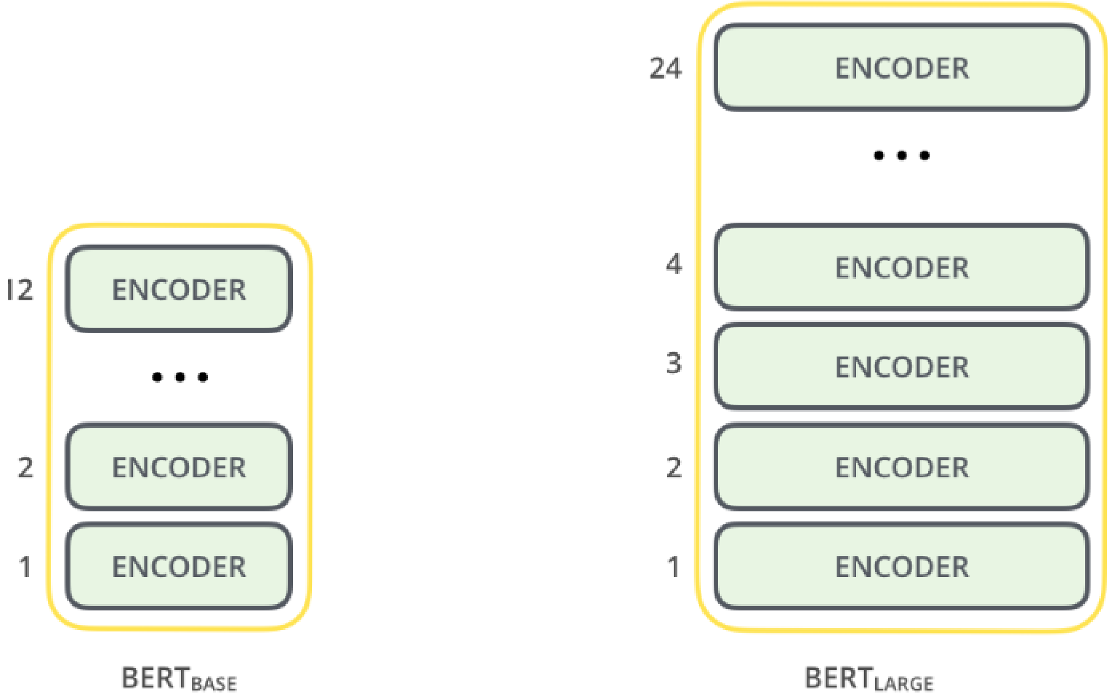
1. Encoder-only models
2. Decoder-only models

Encoder-Only Models: BERT

Let's get rid of the first part

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

- Rip away decoders
  - Just stack encoders



Interlude: Word Embeddings

Q: Why is it called “BERT”?

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

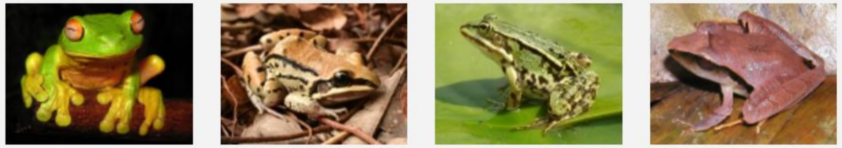
- Story:
  - **2013**: “Dense” word embeddings (**Word2Vec**, **Glove**)
  - Capture some information
  - Surprising properties!

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

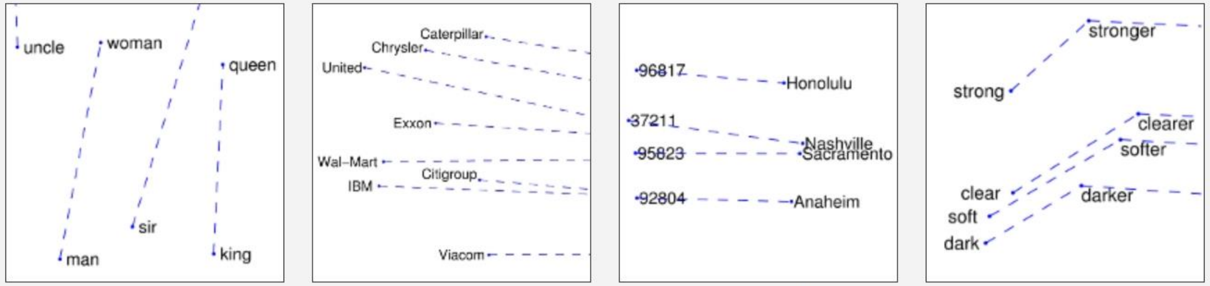


3. litoria 4. leptodactylidae 5. rana 7. eleutherodactylus

2. **Linear substructures**

   The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

   In order to capture in a quantitative way the nuance necessary to distinguish *man* from *woman*, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.



man - woman company - ceo city - zip code comparative - superlative

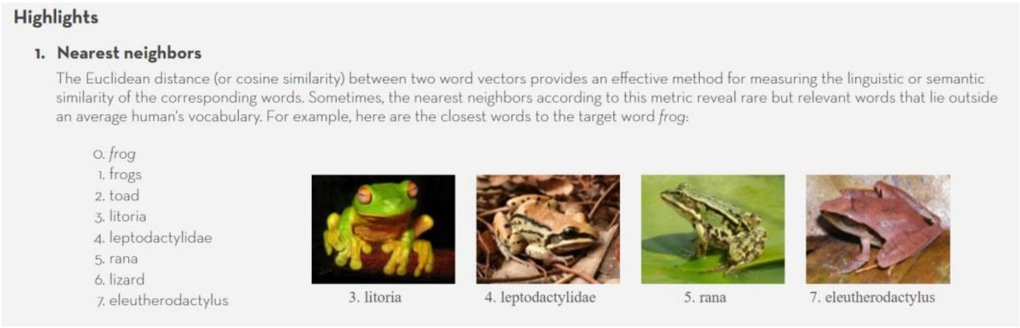
https://nlp.stanford.edu/projects/glove/

Interlude: Word 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?
  - Capturing context---one direction not sufficient!
    - *I went to the bank* *to deposit a check*
    - *I went to the bank* *by the riverside*



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

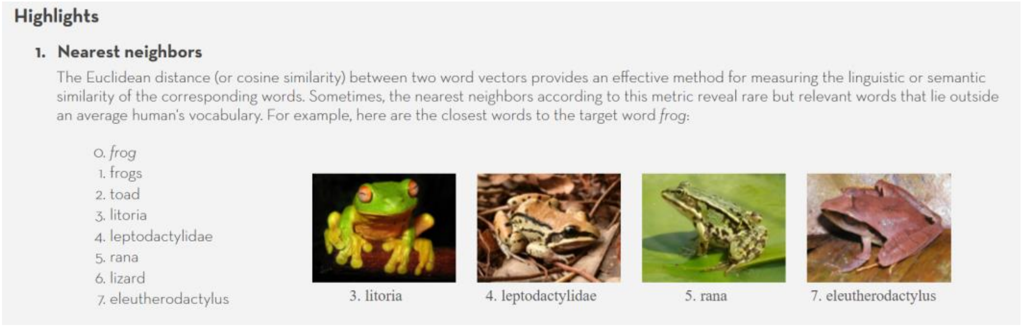
https://nlp.stanford.edu/projects/glove/

Interlude: Word 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
    - **Bidirectional**!
    - 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*:

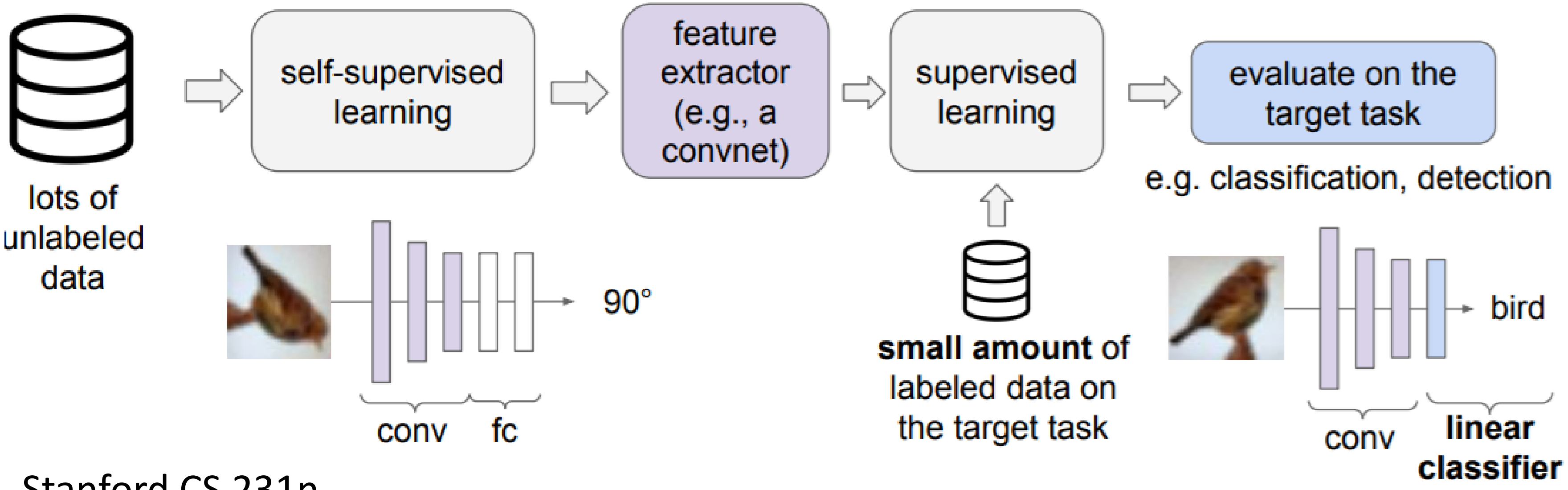
0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

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

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

Interlude: Back to Self-Supervised Learning

- These representations aren't for a *particular* task
- Once we fix a task, we can
  - Fine-tune them
  - Freeze them and build on top of them

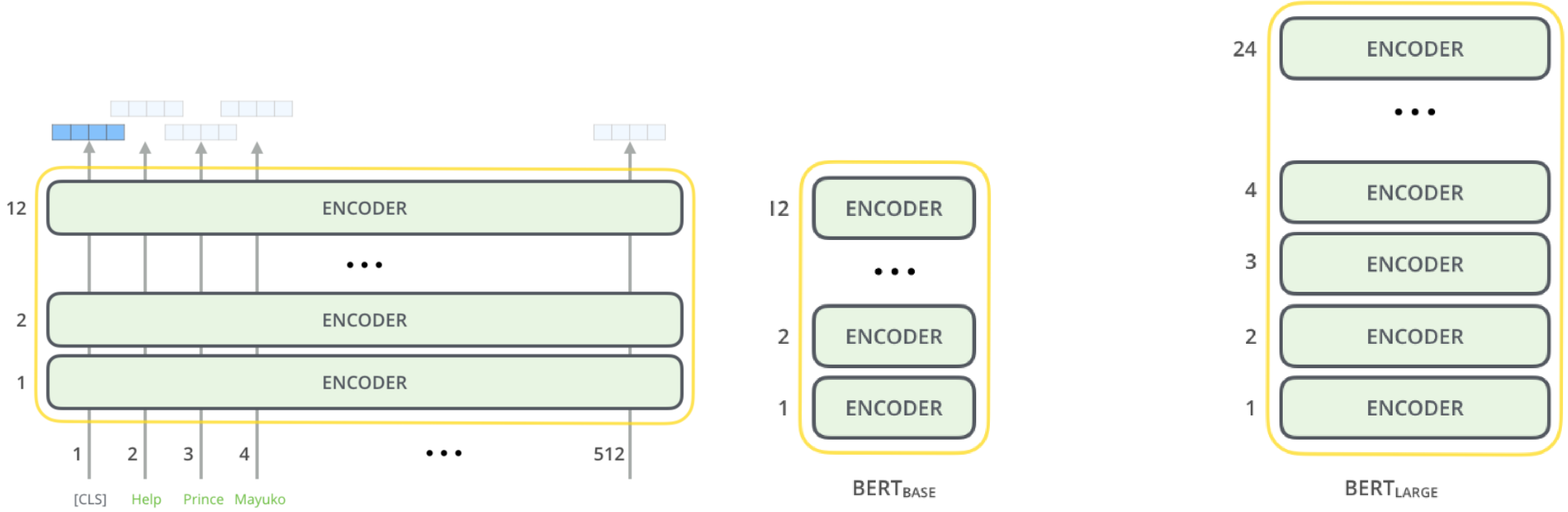


Stanford CS 231n

BERT: Forward Pass

BERT architecture

- Rip away decoders
  - Just stack encoders

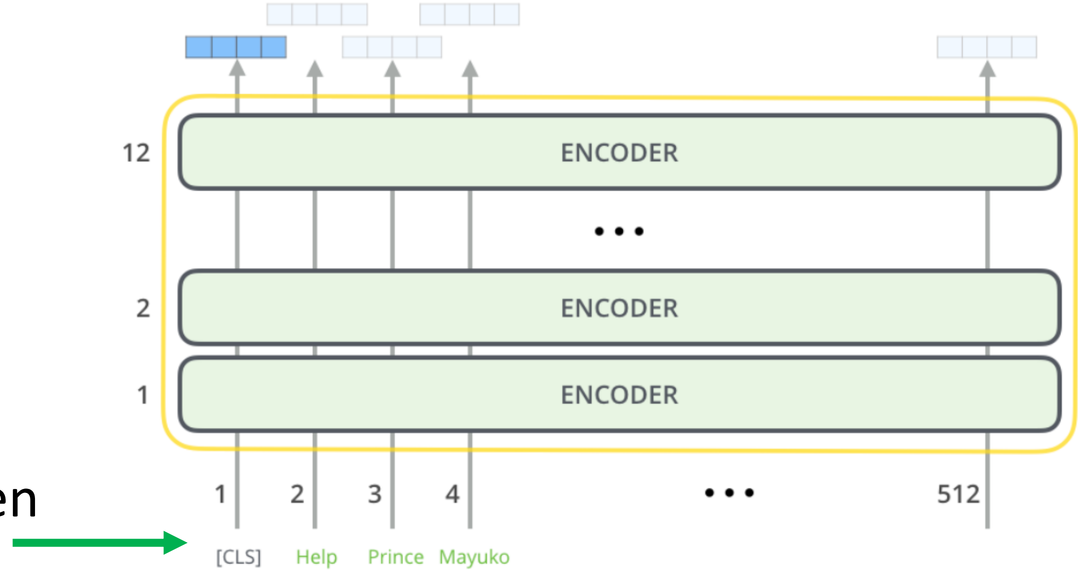


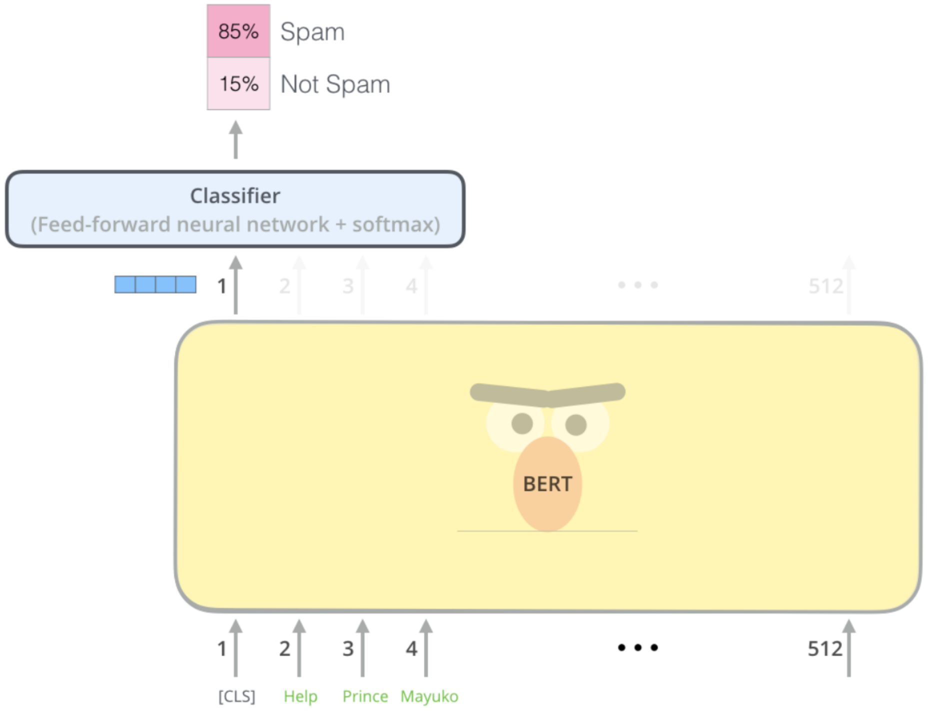
BERT: Using It

How to use it for e.g., classification?

- Special token/word [CLS]
  - Output representation here can be trained on (add classifier!)

CLS token



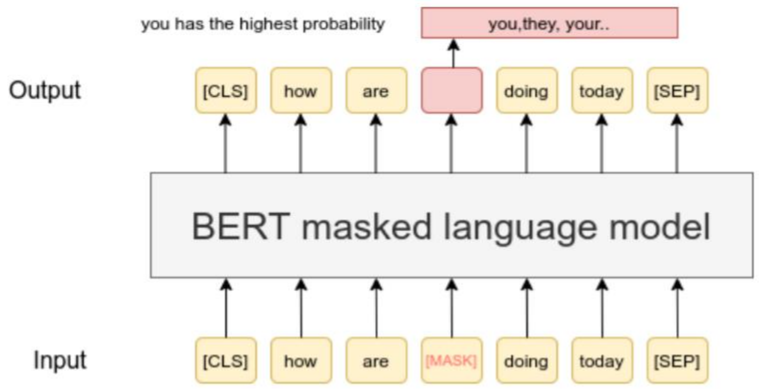


BERT: Training

Training is more interesting!

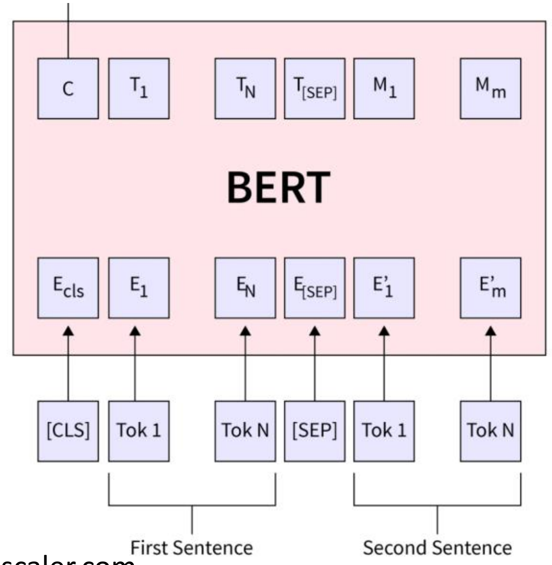
- Pretraining. Then fine-tuning on task of interest

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



sbert.net

1. Masked Language Modeling



scaler.com

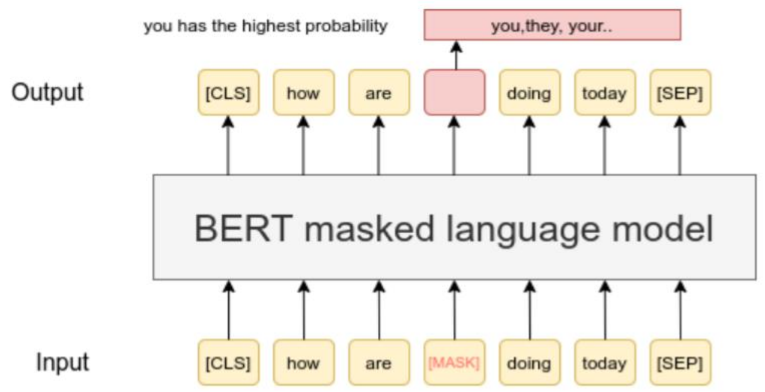
2. Next Sentence Prediction

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

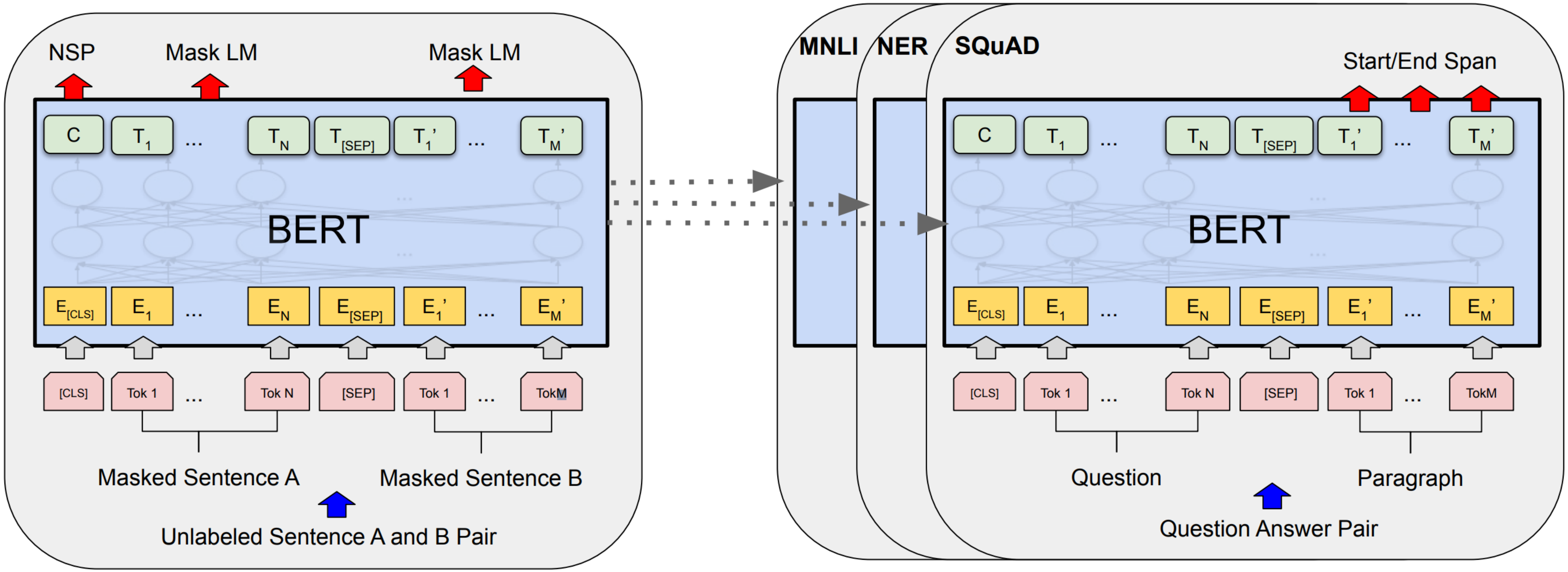


sbert.net

BERT: Fine-Tuning

Training is more interesting,

- Pretraining. Then fine-tuning on task of interest



Pre-training

Fine-Tuning

Devlin et al

BERT: Variations

Lots of work!

- Examples:
  - **RoBERTa**: better trained, better performance
    - 10x more data, no next-sentence prediction pretraining task
  - **SpanBERT:** masking *spans*, not just tokens!
  - **ALBERT:** parameter reduction (but same or better perf.)
    - How? Parameter tying/sharing cross layer, factorization

- Specializations: multilingual, domain-specific
  - BioBERT etc.

Break & Questions

Outline

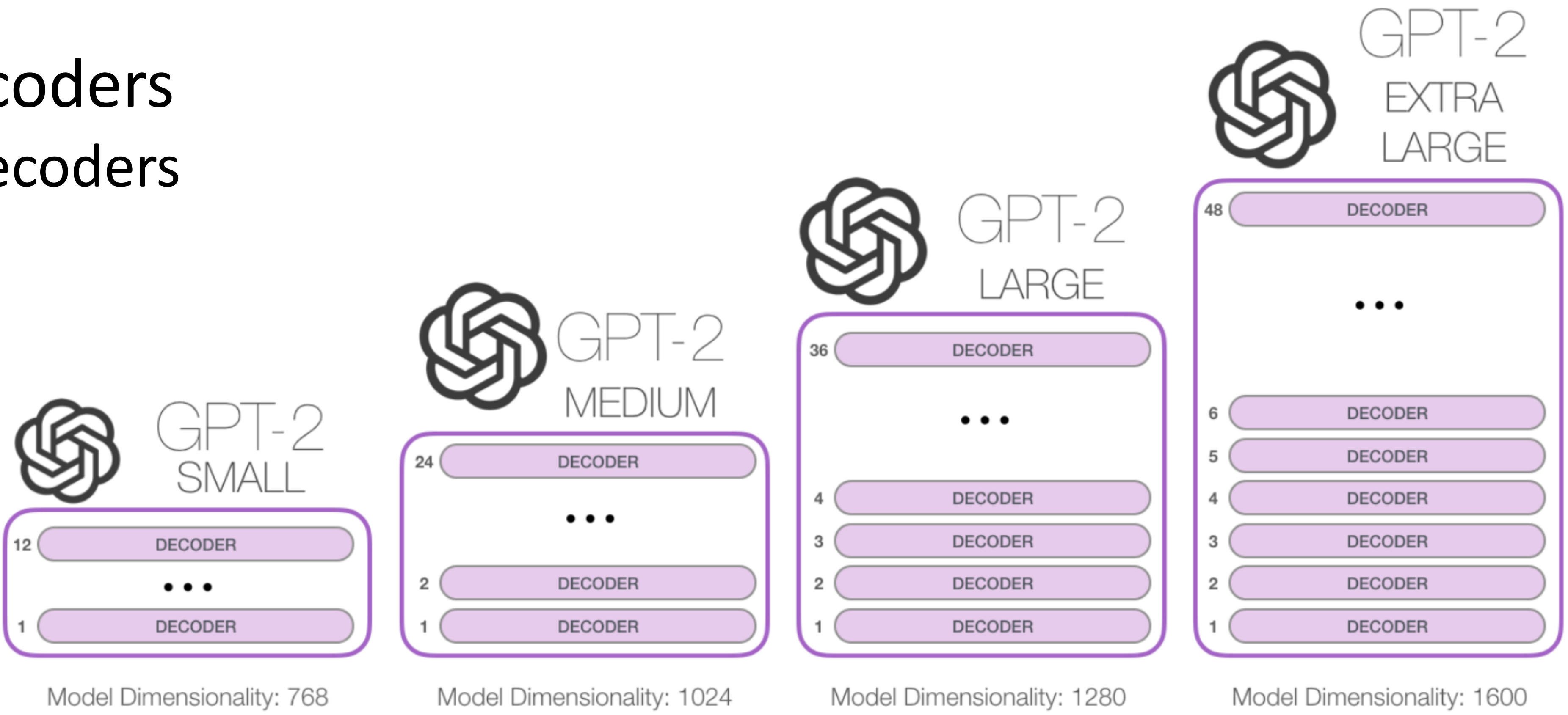
- **From Last Time: Encoder-only Models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Decoder-only Models**
  - Example: GPT, architecture, basic functionality
- **Variations and Advancements**
  - Scaling, upgrades to positional encodings, etc

Decoder-Only Models: GPT

Let's get rid of the first part

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

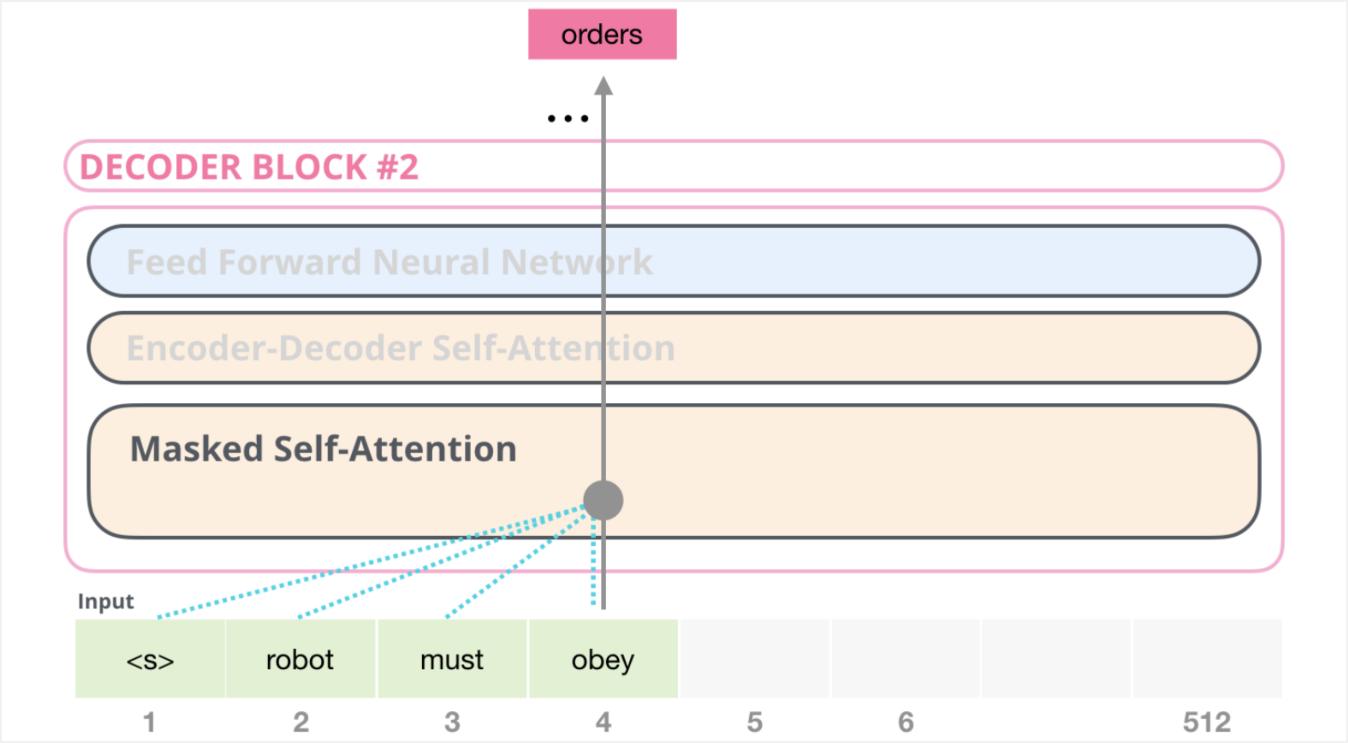
- Rip away encoders
  - Just stack decoders

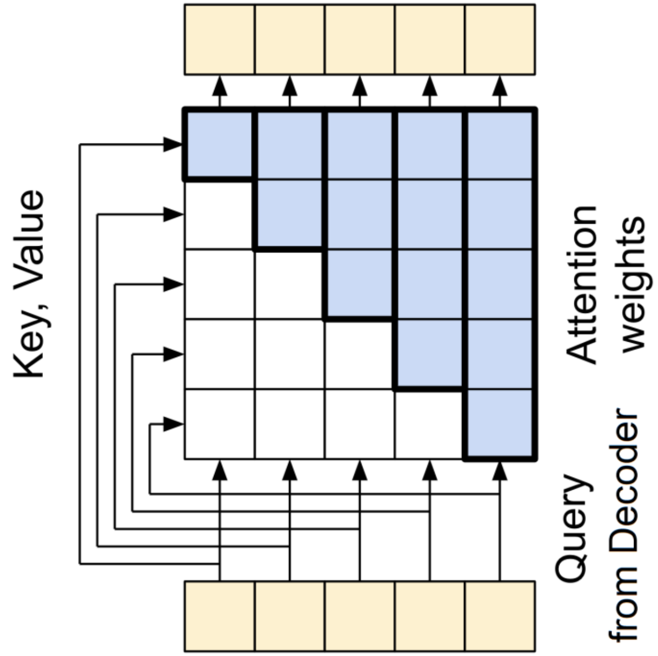


Decoder-Only Models: GPT

Rip away encoders

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



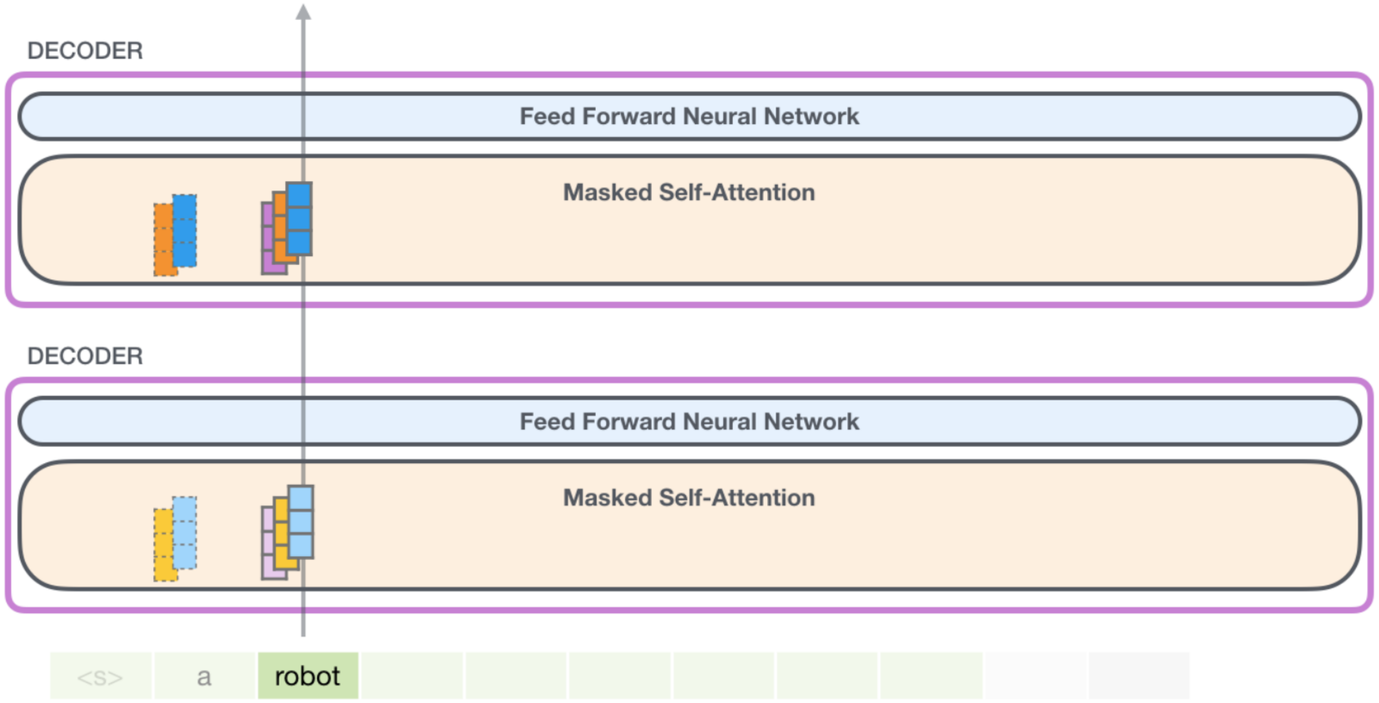


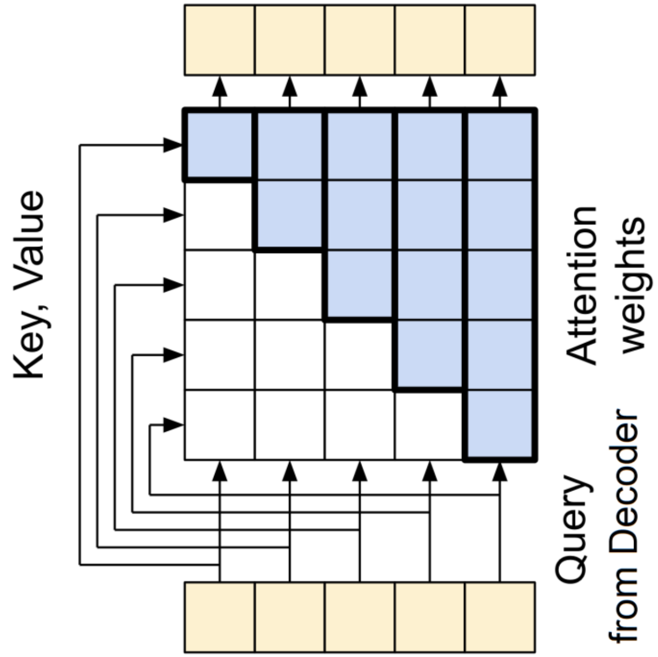
PyLessons

Decoder-Only Models: GPT

Rip away encoders

- Just stack decoders
- Decoders: get rid of **encoder** aspects (masked self-attention only)





PyLessons

From GPT2 to GPT3

Mainly make things larger!

- 96 decoder blocks (getting very tall)
- Context size: **2048**
- 175 billion parameters in total (800GB!)

Training data:

GPT-3 training data[1]:9

Dataset	# tokens	Proportion within training
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

https://en.wikipedia.org/wiki/GPT-3

Break & Questions

Outline

- **From Last Time: Encoder-only Models**
  - Example: BERT, architecture, multitask training, fine-tuning
- **Decoder-only Models**
  - Example: GPT, architecture, basic functionality
- **Variations and Advancements**
  - Scaling, upgrades to positional encodings, etc

Variations

Lots of new large language models

- Same basic idea for the architecture
- Example: PALM
  - Different activations, position embeddings, no biases
  - Scales!

Model	Layers	# of Heads	d_{model}	# of Parameters (in billions)	Batch Size
PaLM 8B	32	16	4096	8.63	$256 \rightarrow 512$
PaLM 62B	64	32	8192	62.50	$512 \rightarrow 1024$
PaLM 540B	118	48	18432	540.35	$512 \rightarrow 1024 \rightarrow 2048$

Chowdhery et al

Thank You!