Foundations: Large Language Models

Sebastian Schuster

Seminar “What do language models really understand”?
April 13, 2023

Plan for today

- Organizational matters
- What are (large) language models?
- The transformer architecture
- Two popular pre-trained models: BERT and GPT-3

Organizational matters

https://sebschu.github.io/lm-understanding-seminar/

Admission

- Everyone here should have received an email that they are admitted/ waitlisted — if not please talk to me after the seminar
- Registrants through CS seminar system: There may still be some changes to the list of participants from CS due to The Algorithm
- Waitlisted participants: List of seminar participants should be finalized next week
- If you are thinking about dropping the course, make up your mind now so that people on the waitlist can take it
- If you don’t make it off the waitlist, you are welcome to audit if you can find a seat

Structure of the course

- First three sessions:
  Lectures by me on foundations

- Remaining 10 sessions:
  2 student presentations on papers each week

  More on presentations next week!

Requirements

- **Everybody should read both papers before class**
- Optional (but probably helpful) to read papers in the first three weeks
- Starting in week 4, you have to submit a question/brief comment on each paper by the evening before the seminar (how to submit TBA)
- You'll get most out of this seminar by engaging in the discussions!

Grading criteria

- For students taking the seminar for **4 credits:**
  - Presentation: 66.6%
  - Questions/comments about readings: 33.3%
- For students taking the seminar for **7 credits:**
  - Presentation: 40%
  - Questions/comments about readings: 20%
  - Final paper: 40%
- For people who are not in the LST MS program: Ask your study advisor whether you can take the seminar for **4 credits.**

Schedule

Date	Topic	Papers	Presenter
04/13/2023	Foundations: Large Language Models	Devlin et al. (2019), Brown et al. (2020)	Sebastian
04/20/2023	Foundations: Fine-tuning and reinforcement learning from human feedback	Ouyang et al. (2022)	Sebastian
04/27/2023	Foundations: What does it mean to 'understand'? Methods for assessing understanding.	Bender and Koller (2020), Piantadosi and Hill (2022)	Sebastian
05/04/2023	Methods: Behavioral experiments and probing	Linzen et al. (2016), Tenney et al. (2019)	
05/11/2023	Negation	Ettinger (2020), Shivagunde et al. (2023) ?	
05/16/2023 (Special day/time!)	Compositionality	Kim and Linzen (2020), Qiu et al. (2022)	
05/18/2023	*no class* (public holiday)		
05/25/2023	Entity tracking / world models I	Li et al. (2021), Kim and Schuster (to appear)	
06/01/2023	Entity tracking / world models II	Toshniwal et al. (2021), Li et al. (2023)	
06/06/2023 (Special day/time!)	Discourse understading and connectives	Pandia and Ettinger (2021), Pandia et al. (2021)	
06/08/2023	*no class* (public holiday)		
06/15/2023	Pragmatic inferences	Hu et al. (2022), Ruis et al. (2022)	
06/22/2023	Metaphors / Figurative meaning	TBD	
06/29/2023	Grounding I	TBD	
07/06/2023	Grounding II	TBD	
07/13/2023	*no class*		
07/20/2023	*no class*		

Signup for presentation slots happening next week!

Things to note about schedule

- **We end early**: No lectures on July 13 and July 20!
- **2 public holidays**: No lectures on May 18 and Jun 8!
- **2 special meetings**:
  May 16 and Jun 6 8:15-9:45?

Contents

- **3 foundation lectures:**
  - (Large) language models
  - Recent developments: Finetuning and reinforcement learning on human feedback
  - Philosophical background: What does it mean to "understand"?
- 1 week: **foundational papers on evaluating LM capabilities** (syntax and semantics)
- 9 weeks: evaluating various aspects of understanding

Office hours

- Send me an email / a message on teams to schedule a meeting

Questions about organizational matters?

Language Models

What is a language model?

P(*next word* | *context*)

A conditional probability distribution over the **next word** from a fixed vocabulary,
given **a sequence of previous words**.

What is a language model?

$P(\textit{next word} \mid \text{"The cat"})$

Next word	P(next word | context)
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.004
…	
sat	0.1
…	
zebra	0.00007

Scoring words and sequences

Scoring words:

$$P(\textit{next word} \mid \textit{context})$$

Scoring sequences:

$$P(\textit{on a mat} \mid \textit{the cat sat})$$

$$= P(\textit{on} \mid \textit{the cat sat})$$

Generating texts

the cat

Next word	**P(next word | the cat)**
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.004
…	
sat	0.1
…	
zebra	0.00007

Generating texts

the cat sat

Next word	P(next word | the cat)
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.004
…	
sat	0.1
…	
zebra	0.00007

Generating texts

the cat sat

Next word	P(next word | the cat sat)
a	0.0000006
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.15
…	
sat	0.0001
…	
zebra	0.00007

Generating texts

the cat sat on

Next word	P(next word | the cat sat on)
a	0.2
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.0000003
…	
on	0.0000015
…	
sat	0.0001
…	
zebra	0.00007

Generating texts

the cat sat on a

Next word	P(next word | the cat sat on a)
a	0.000004
aardvark	0.000002
aarhus	0.0000001
…	
mat	0.1
…	
on	0.0000015
…	
sat	0.0001
…	
zebra	0.007

Generating texts

the cat sat on a mat

Where do the probabilities come from?

- **Pre-2015ish:**
  - **Counting** short sequences in large corpora
  - One problem: Estimates are very poor for very rare sequences/sequences that don’t appear in the corpus
- **Post-2015ish:**
  - **Neural language models**

A neural language model

Context — Context of previous words $w_1, w_2, \ldots, w_k$

? — Some mysterious neural network

$P\left(w_{k+1}\right)$ — Probability distribution over the next word $P\left(w_{k+1}\right)$

How to represent the context?

- Neural networks can only process **numerical inputs**
- We therefore need to represent context $w_1, w_2, \ldots, w_k$ using **numbers**
- One method — **one-hot encoding**: A vector such that one dimension corresponds to one word in vocabulary (= the *finite* set of words that can be encoded)
- The representation of a word is a vector with one 1 (hence one-hot) and 0 for all other dimensions

Word embeddings

- Alternative to one-hot encoding — **word embeddings:** Represent every word as a **continuous *d*-dimensional vector** (for example, 300-dimensional vector)
- **Learn these vectors** as part of training the language model
- Ideally, these **vectors are similar** (low cosine distance between vectors) **for words with similar meaning**
  - vectors for *cat* and *dog* should be closer together than vectors for *cat* and *marmalade*
  - In practice, this tends to happen

Word embeddings: Example

- Vocabulary V = {cat, dog, mat, on, sat, the}
- Dimension *d* = 5

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

- Values are now ***uninterpretable*** but ideally ***encode similarity***
- **Dense** (= not sparse) encoding — we use a fixed dimension independent of vocabulary size

Word embeddings: Remaining issue

- **Still difficult to represent words that are not in the vocabulary:**
  - **Solutions**:
    - Learning a vector for a special <*UNK*> word
    - Using **character based embeddings** (for example, embeddings for every letter in the alphabet)
    - Using **subword tokens**

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$

1. For each word w_i, look up word vectors in **embedding table** of dimension $|V| \times d$

   ➡ results in a list of word vectors $v_1, v_2, \ldots, v_k$ where v_i corresponds to the word vector for word w_i

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

2. We **stack** these vectors to form a matrix of dimension $d \times k$

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: The cat sat on the

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: **The** cat sat on the

1. Look up vectors:

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}$$

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: The **cat** sat on the

1. Look up vectors:

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}, \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix}$$

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: The cat **sat** on the

1. Look up vectors:

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}, \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix}, \begin{pmatrix} -1.079 \\ -0.612 \\ 0.594 \\ -1.057 \\ -1.186 \end{pmatrix}$$

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: The cat sat **on** the

1. Look up vectors:

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}, \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix}, \begin{pmatrix} -1.079 \\ -0.612 \\ 0.594 \\ -1.057 \\ -1.186 \end{pmatrix}, \begin{pmatrix} -0.262 \\ -0.923 \\ 1.097 \\ -0.724 \\ -1.078 \end{pmatrix}$$

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: The cat sat on **the**

1. Look up vectors:

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}, \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix}, \begin{pmatrix} -1.079 \\ -0.612 \\ 0.594 \\ -1.057 \\ -1.186 \end{pmatrix}, \begin{pmatrix} -0.262 \\ -0.923 \\ 1.097 \\ -0.724 \\ -1.078 \end{pmatrix}, \begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}$$

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

How to represent a context $w_1, w_2, \ldots, w_k$: Example

Input: The cat sat on the

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

1. Look up vectors:

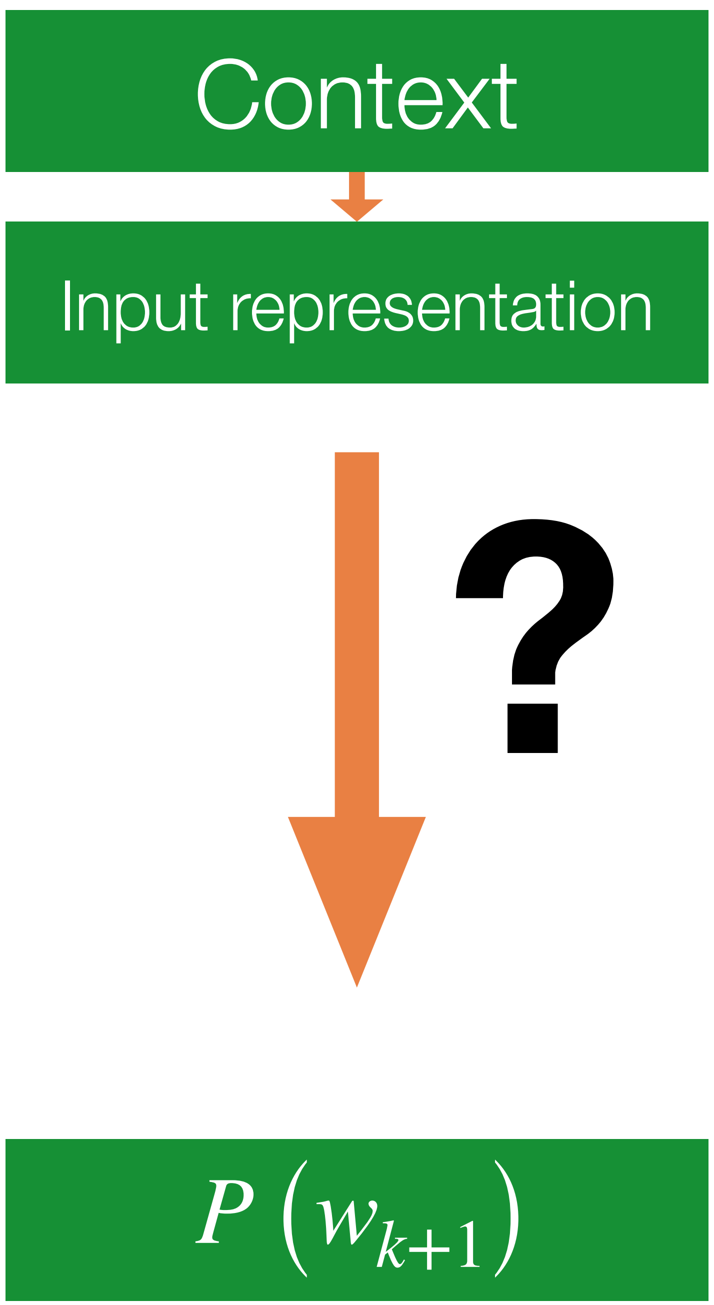
$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}, \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix}, \begin{pmatrix} -1.079 \\ -0.612 \\ 0.594 \\ -1.057 \\ -1.186 \end{pmatrix}, \begin{pmatrix} -0.262 \\ -0.923 \\ 1.097 \\ -0.724 \\ -1.078 \end{pmatrix}, \begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}$$

2. Stack vectors to form input matrix of dimension $d \times k$:

$$\begin{pmatrix} 0.544 & -0.023 & -1.079 & -0.262 & 0.544 \\ -0.678 & 1.354 & -0.612 & -0.923 & -0.678 \\ 0.604 & -0.553 & 0.594 & 1.097 & 0.604 \\ 0.944 & -0.367 & -1.057 & -0.724 & 0.944 \\ 0.632 & 0.975 & -1.186 & -1.078 & 0.632 \end{pmatrix}$$

input representation

A neural language model



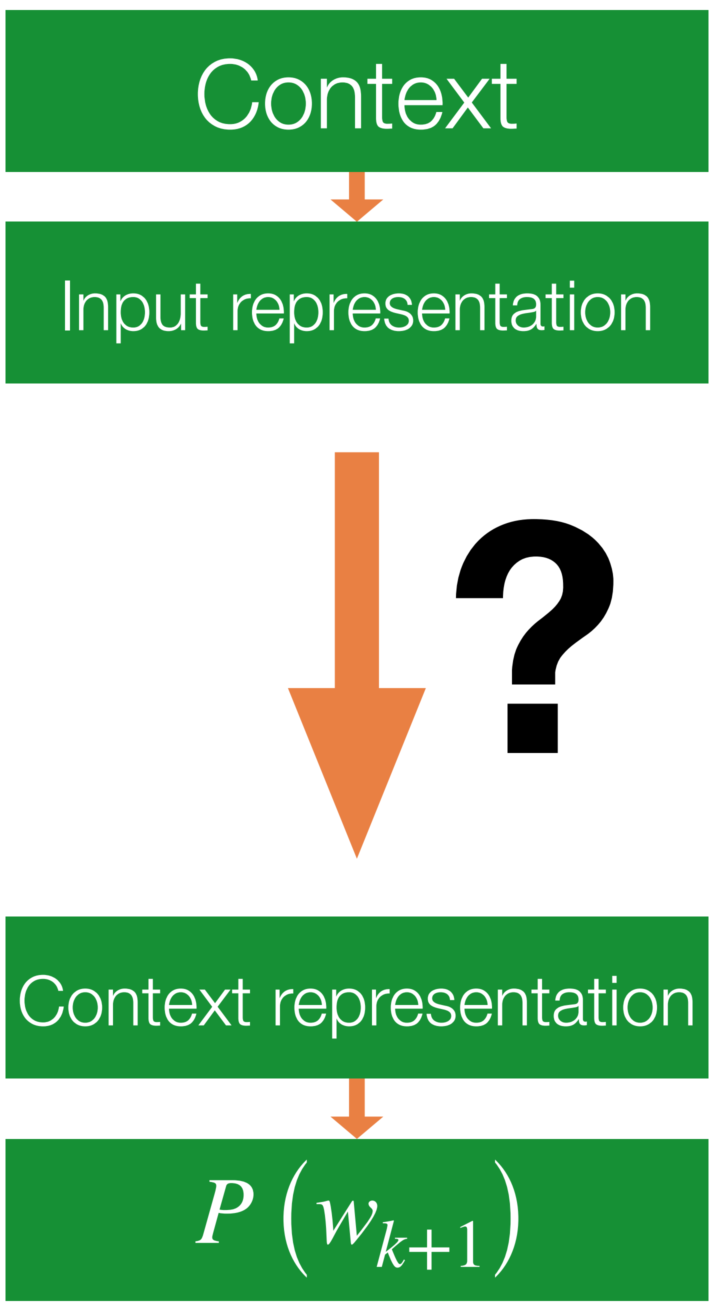
Context of previous words $w_1, w_2, \ldots, w_k$

Matrix with word embeddings

Some mysterious neural network

Probability distribution over the next word $P\left(w_{k+1}\right)$

A neural language model



Context of previous words $w_1, w_2, \ldots, w_k$

Matrix with word embeddings

Some mysterious neural network

Vector representing the context

Probability distribution over the next word $P\left(w_{k+1}\right)$

Computing the probability of the next word: SoftMax

Context representation	l-dimensional vector representing the context c
$P\left(w_{k+1}\right)$	Probability distribution over the next word $P\left(w_{k+1}\right)$

Weight matrix S:

	d_1	d_2	…	d_{l-1}	d_l
cat	-4.496	0.363	…	5.246	0.534
dog	-0.053	0.652	…	-1.370	-2.637
mat	0.610	0.079	…	0.750	-1.942
on	-0.262	-0.657	…	0.897	-1.577
sat	0.945	-0.864	…	-3.184	0.991
the	0.739	0.902	…	-5.206	3.288

$$P\left(w_{k+1} \mid c\right) = \frac{\exp s_{w_{k+1}} \cdot c}{\sum_{w' \in V} \exp s_{w'} \cdot c}$$

Computing the probability of the next word: SoftMax

Context representation

$P\left(w_{k+1}\right)$

l-dimensional vector representing the context c

Probability distribution over the next word $P\left(w_{k+1}\right)$

Dot product between weight for word w_{k+1} and context vector c

$$P\left(w_{k+1} \mid c\right) = \frac{\exp \boxed{s_{w_{k+1}} \cdot c}}{\sum_{w' \in V} \exp s_{w'} \cdot c}$$

Weight matrix S:

	d_1	d_2	…	d_{l-1}	d_l
cat	-4.496	0.363	…	5.246	0.534
dog	-0.053	0.652	…	-1.370	-2.637
mat	0.610	0.079	…	0.750	-1.942
on	-0.262	-0.657	…	0.897	-1.577
sat	0.945	-0.864	…	-3.184	0.991
the	0.739	0.902	…	-5.206	3.288

Computing the probability of the next word: SoftMax

Context representation

$P\left(w_{k+1}\right)$

l-dimensional vector representing the context c

Probability distribution over the next word $P\left(w_{k+1}\right)$

Dot product between weight for word w_{k+1} and context vector c

$$P\left(w_{k+1} \mid c\right) = \frac{\exp s_{w_{k+1}} \cdot c}{\sum_{w' \in V} \exp s_{w'} \cdot c}$$

Normalization so that $P\left(w_{k+1} \mid c\right)$ is a proper probability distribution

Weight matrix S:

	d_1	d_2	…	d_{l-1}	d_l
cat	-4.496	0.363	…	5.246	0.534
dog	-0.053	0.652	…	-1.370	-2.637
mat	0.610	0.079	…	0.750	-1.942
on	-0.262	-0.657	…	0.897	-1.577
sat	0.945	-0.864	…	-3.184	0.991
the	0.739	0.902	…	-5.206	3.288

Computing the probability of the next word: SoftMax

Context representation → $P\left(w_{k+1}\right)$

Context representation: l-dimensional vector representing the context c

$P\left(w_{k+1}\right)$: Probability distribution over the next word $P\left(w_{k+1}\right)$

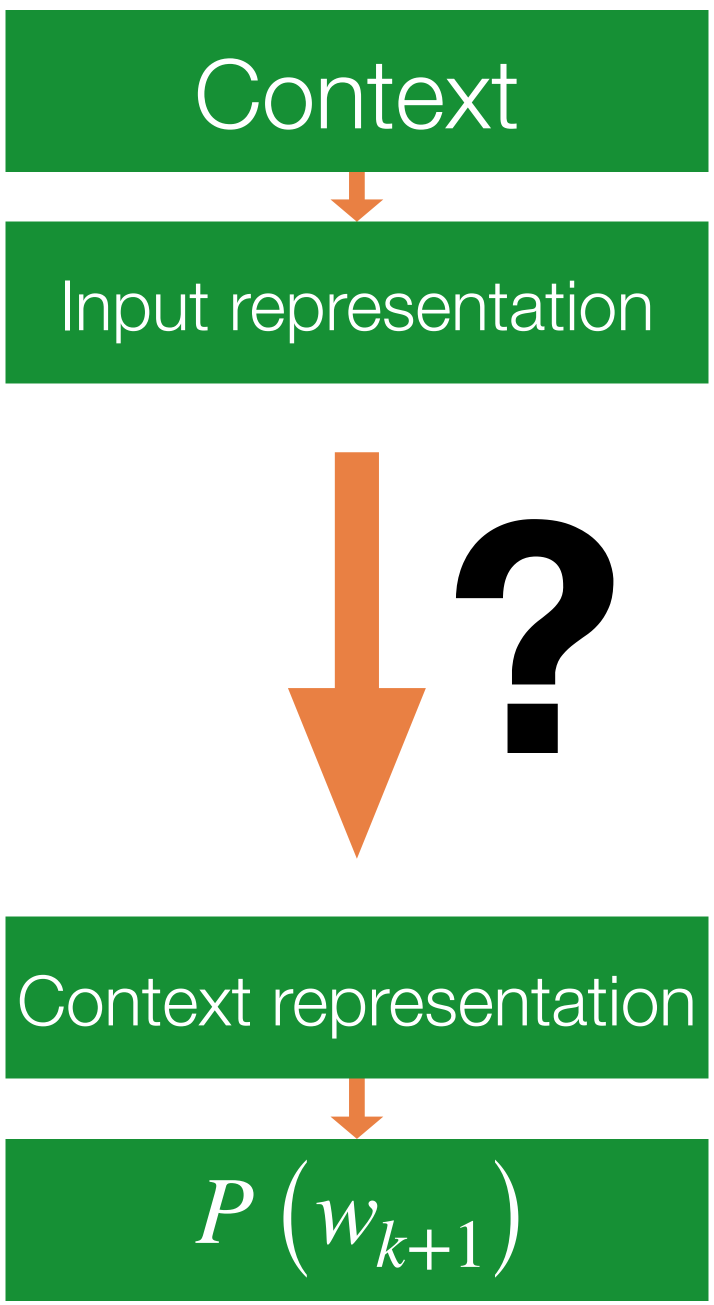
Weight matrix S:

	d_1	d_2	…	d_{l-1}	d_l
cat	-4.496	0.363	…	5.246	0.534
dog	-0.053	0.652	…	-1.370	-2.637
mat	0.610	0.079	…	0.750	-1.942
on	-0.262	-0.657	…	0.897	-1.577
sat	0.945	-0.864	…	-3.184	0.991
the	0.739	0.902	…	-5.206	3.288

mat row $= s_{\text{mat}}$

$$P\left(\text{mat} \mid c\right) = \frac{\exp s_{\text{mat}} \cdot c}{\sum_{w' \in V} \exp s_{w'} \cdot c}$$

A neural language model



Context of previous words $w_1, w_2, \ldots, w_k$

Matrix with word embeddings

Some mysterious neural network

Vector c representing the context

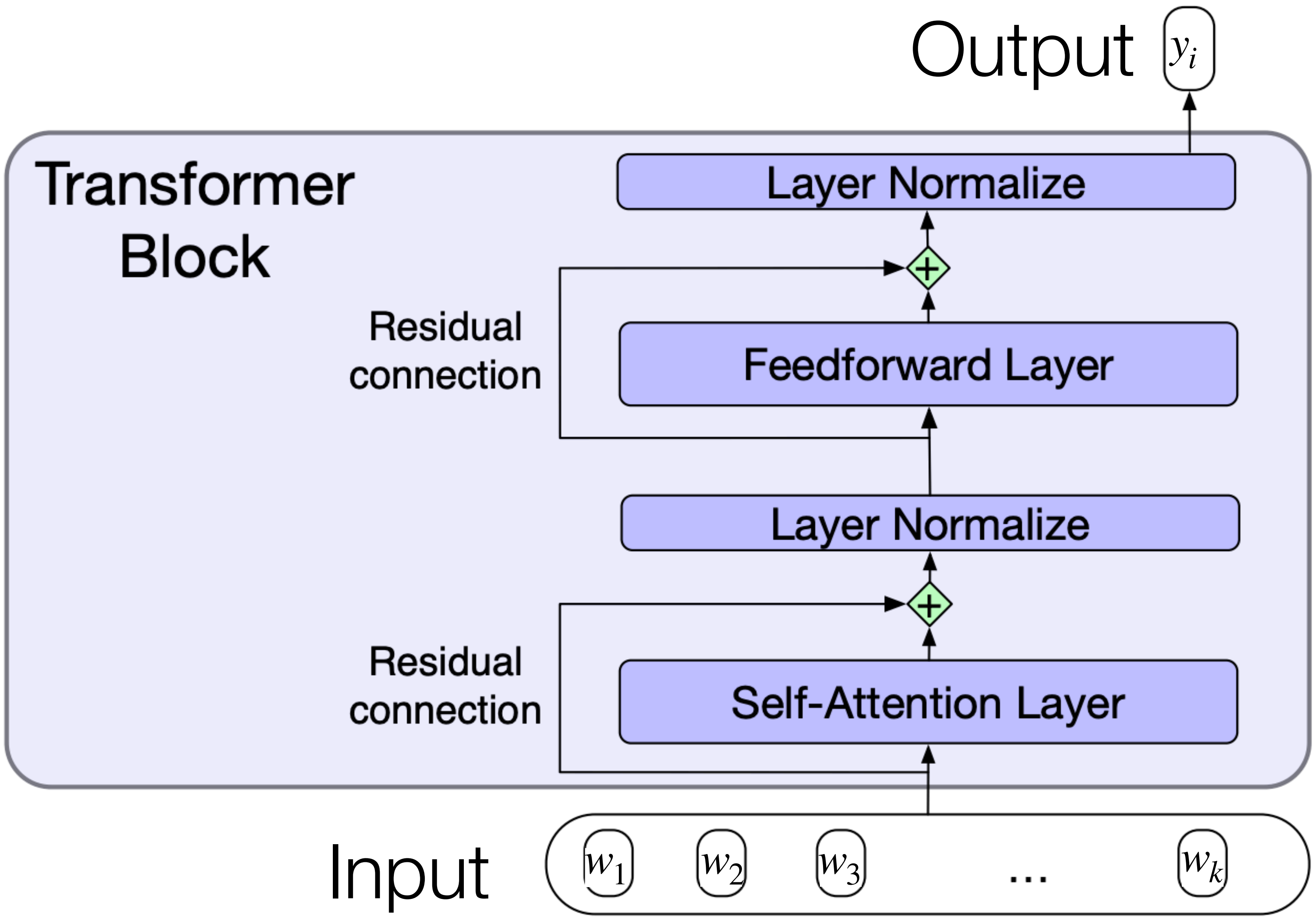
Probability distribution over the next word $P\left(w_{k+1}\right)$

The Transformer Architecture

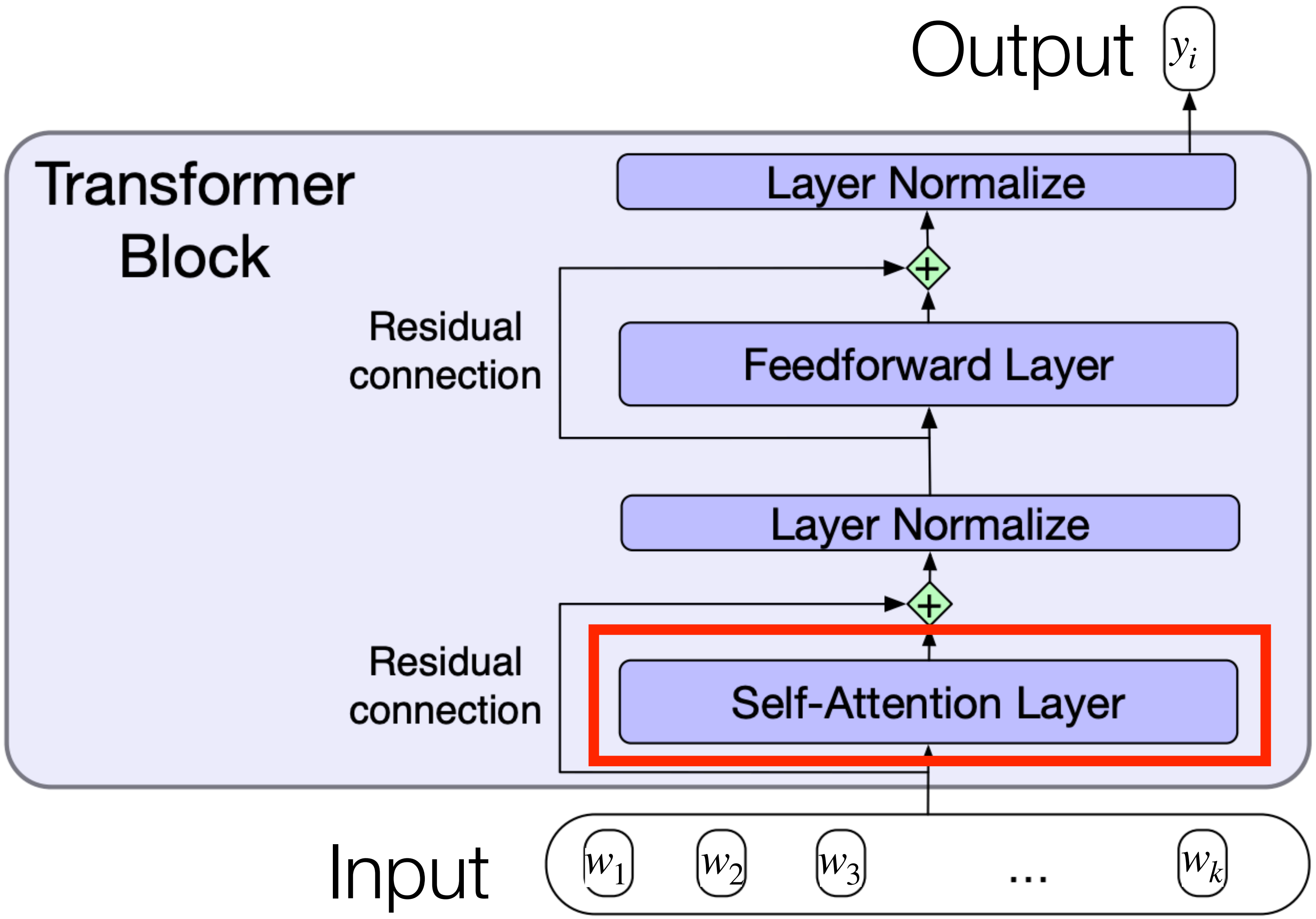
Transformer models

- Transformer models take an input representation and output a context representation
- They explicitly model **word order** and **dependencies between words**

Overall architecture



Overall architecture



Self-attention

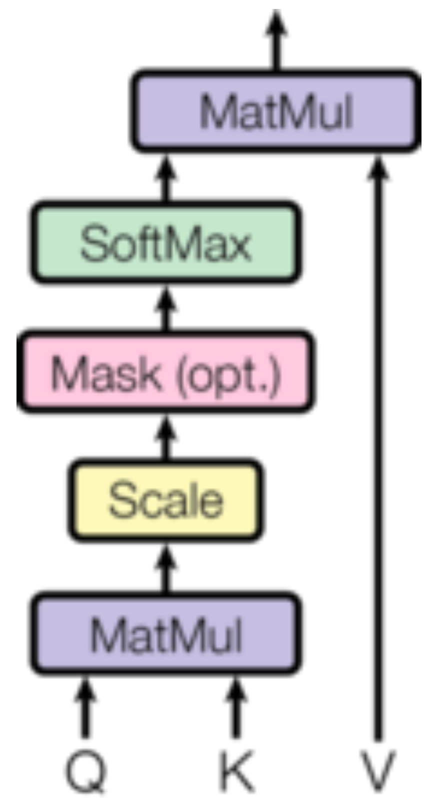
- **Intuition:** the output representation y_i of a word w_i should be a combination of **its own representation** and the representations of **other words that it depends on** (syntactically, in terms of meaning, …)

- We do this by computing an **attention vector** α_i

- The output representation y_i is a weighted sum of all the input representations

$$y_i = \sum_{0 \leq j \leq k} \alpha_{ij} w_j$$

Scaled dot-product attention



queries Q

keys K

$$\alpha_i = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)_i$$

scaling factor

Multi-head attention: Motivation

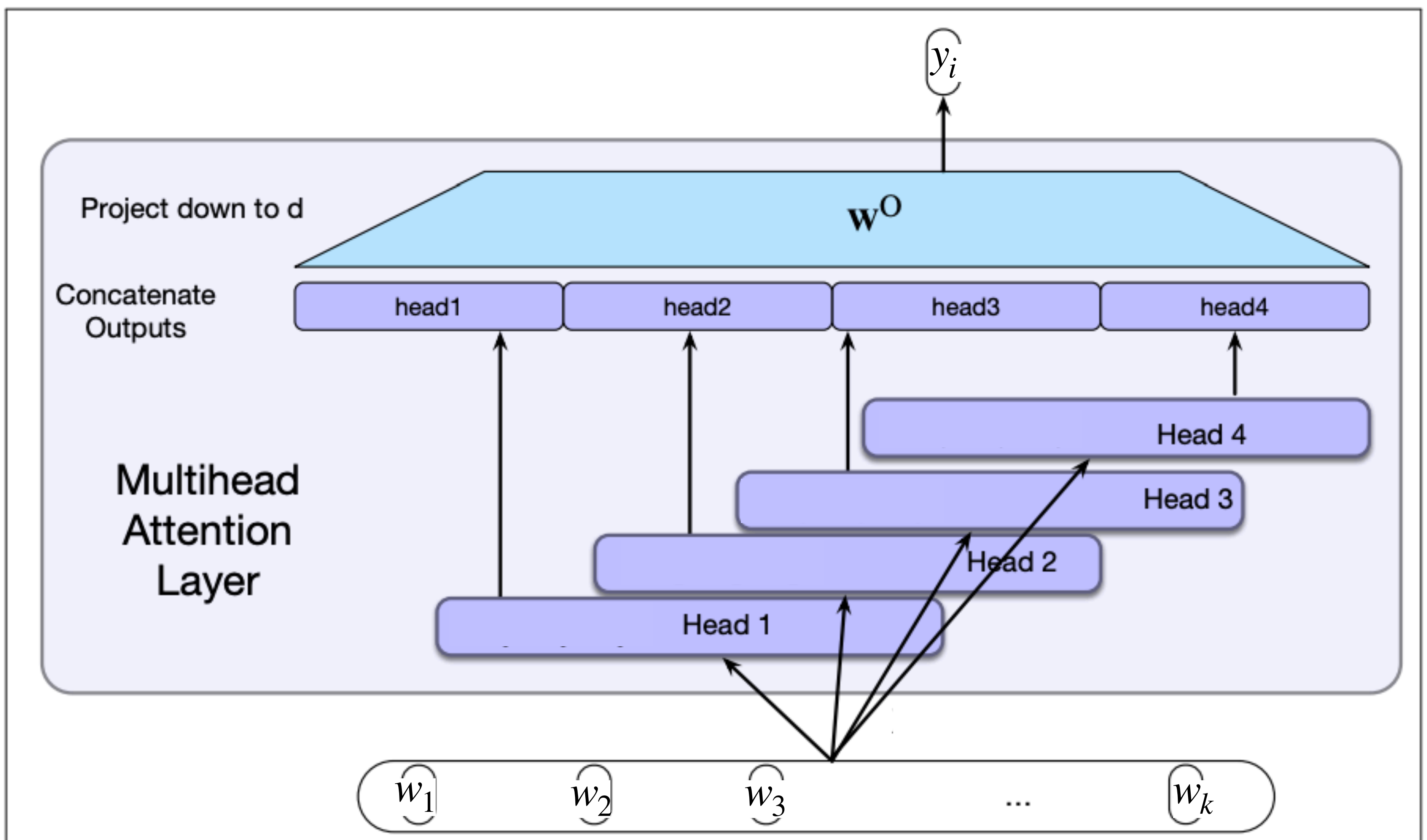
- There are **multiple types of dependencies** between words:
  - Syntactic: "The **keys** to the cabinet **are** on the table"
  - Arguments of a verb: "The **dog** **chases** the **cat**"
  - Co-reference: "The **student** saw **herself** in the mirror"
- A single self attention mechanism cannot really capture all these different types of dependencies

Multi-head attention

- Transformers generally use **multiple attention mechanisms** (called heads) within a single layer
- Each of these attention mechanisms uses **its own set of parameters** to compute the attention vector α_i
- We therefore compute a separate attention vector for each of the p heads: $\alpha_i^{(1)}, \alpha_i^{(2)}, \ldots, \alpha_i^{(p)}$
  and compute a separate weighted output representation for each head: $y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(p)}$

Multi-head attention

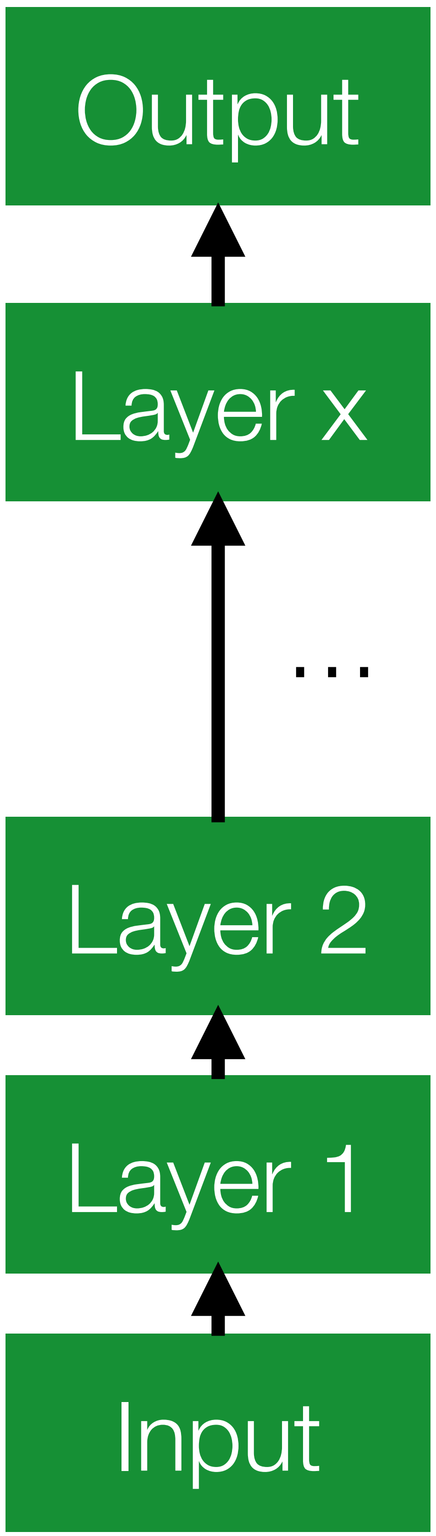
- The weighted output representations $y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(p)}$ are then **concatenated and projected down** to the same dimension as the input representation



Multiple layers

- Instead of just doing all these transformations once, Transformer models usually consist of multiple layers (in practice, usually somewhere between 5 and 20 layers)

- The input of layer l is the output of layer $l-1$



Modeling word order

- The model so far does not encode anything about word order

- We sum over all word representations (weighted by the attention weights) when computing an output representations

    - “the dog chases the cat” and “the cat chases the dog” once again have the same representation

- Solution: **positional embeddings**

How to represent a context $w_1, w_2, \ldots, w_k$ with positional embeddings

Input: The cat sat on the

1. Look up vectors:

$$\begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}, \begin{pmatrix} -0.023 \\ 1.354 \\ -0.553 \\ -0.367 \\ 0.975 \end{pmatrix}, \begin{pmatrix} -1.079 \\ -0.612 \\ 0.594 \\ -1.057 \\ -1.186 \end{pmatrix}, \begin{pmatrix} -0.262 \\ -0.923 \\ 1.097 \\ -0.724 \\ -1.078 \end{pmatrix}, \begin{pmatrix} 0.544 \\ -0.678 \\ 0.604 \\ 0.944 \\ 0.632 \end{pmatrix}$$

	d_1	d_2	d_3	d_4	d_5
cat	-0.023	1.354	-0.553	-0.367	0.975
dog	-0.053	0.644	-0.245	-0.322	1.056
mat	-0.753	-0.679	0.755	0.054	0.750
on	-0.262	-0.923	1.097	-0.724	-1.078
sat	-1.079	-0.612	0.594	-1.057	-1.186
the	0.544	-0.678	0.604	0.944	0.632

2. Stack vectors to form input matrix of dimension $d \times k$:

$$\begin{pmatrix} 0.544 & -0.023 & -1.079 & -0.262 & 0.544 \\ -0.678 & 1.354 & -0.612 & -0.923 & -0.678 \\ 0.604 & -0.553 & 0.594 & 1.097 & 0.604 \\ 0.944 & -0.367 & -1.057 & -0.724 & 0.944 \\ 0.632 & 0.975 & -1.186 & -1.078 & 0.632 \end{pmatrix}$$

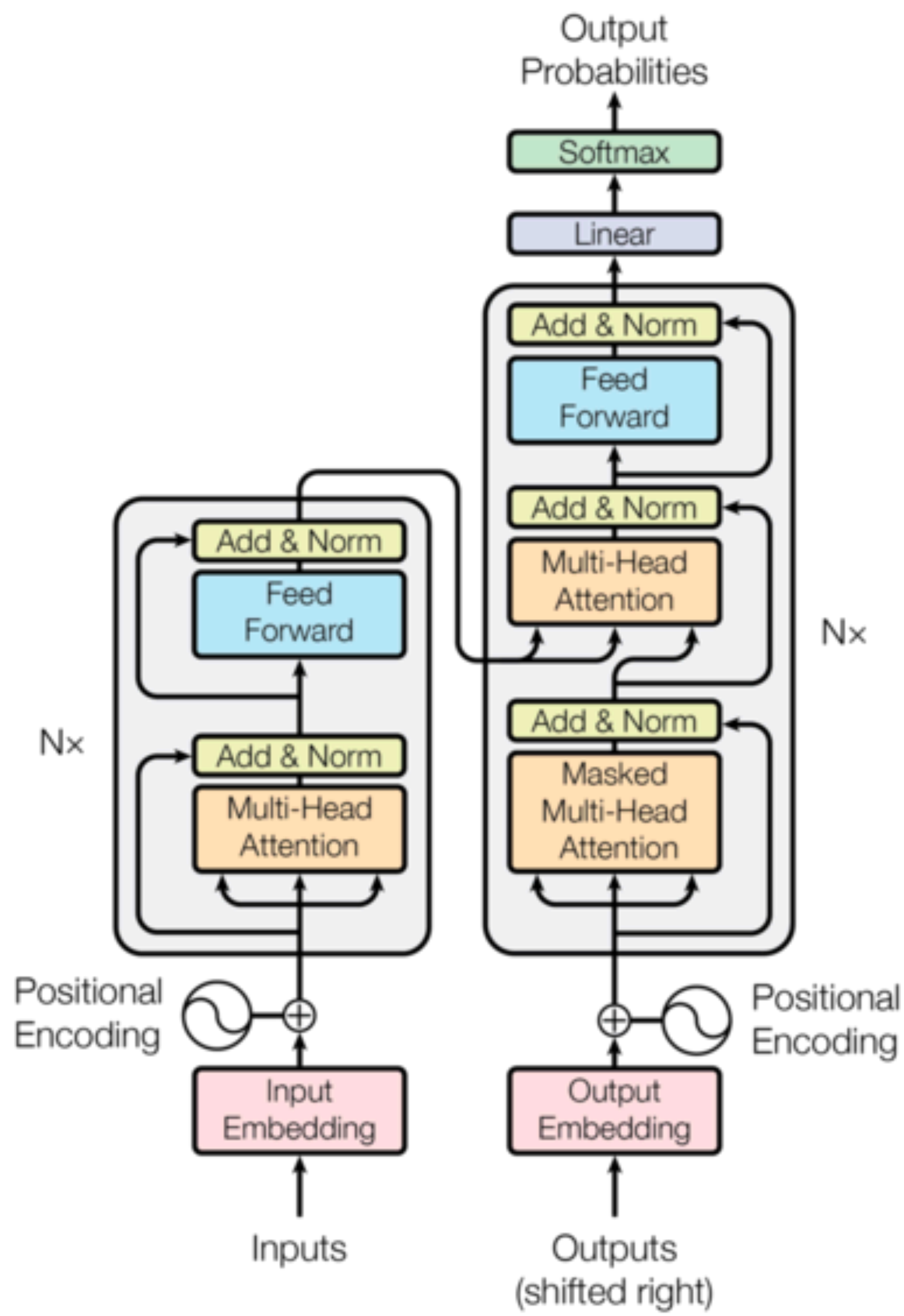
3. Add parameters indicating the position of each word:

$$\begin{pmatrix} 0.544 & -0.023 & -1.079 & -0.262 & 0.544 \\ -0.678 & 1.354 & -0.612 & -0.923 & -0.678 \\ 0.604 & -0.553 & 0.594 & 1.097 & 0.604 \\ 0.944 & -0.367 & -1.057 & -0.724 & 0.944 \\ 0.632 & 0.975 & -1.186 & -1.078 & 0.632 \\ 0.323 & 1.343 & 3.343 & -1.232 & 2.232 \\ -1.234 & 0.448 & -0.379 & -1.114 & -0.593 \end{pmatrix}$$

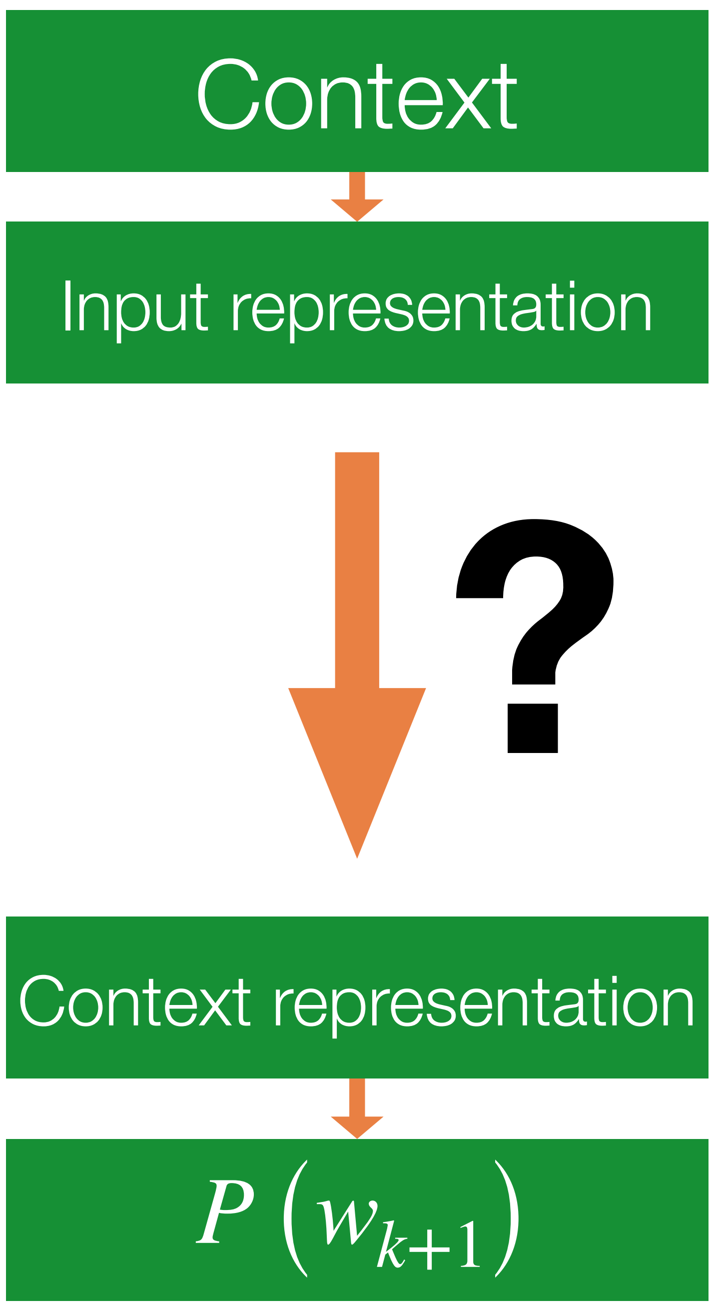
positional embeddings

	d_1	d_2
1	0.323	-1.234
2	1.343	0.448
3	3.343	-0.379
4	-1.232	-1.114
5	2.232	-0.593
6	0.534	-1.988

Putting it all together: The full Transformer model



A neural language model



Context of previous words $w_1, w_2, \ldots, w_k$

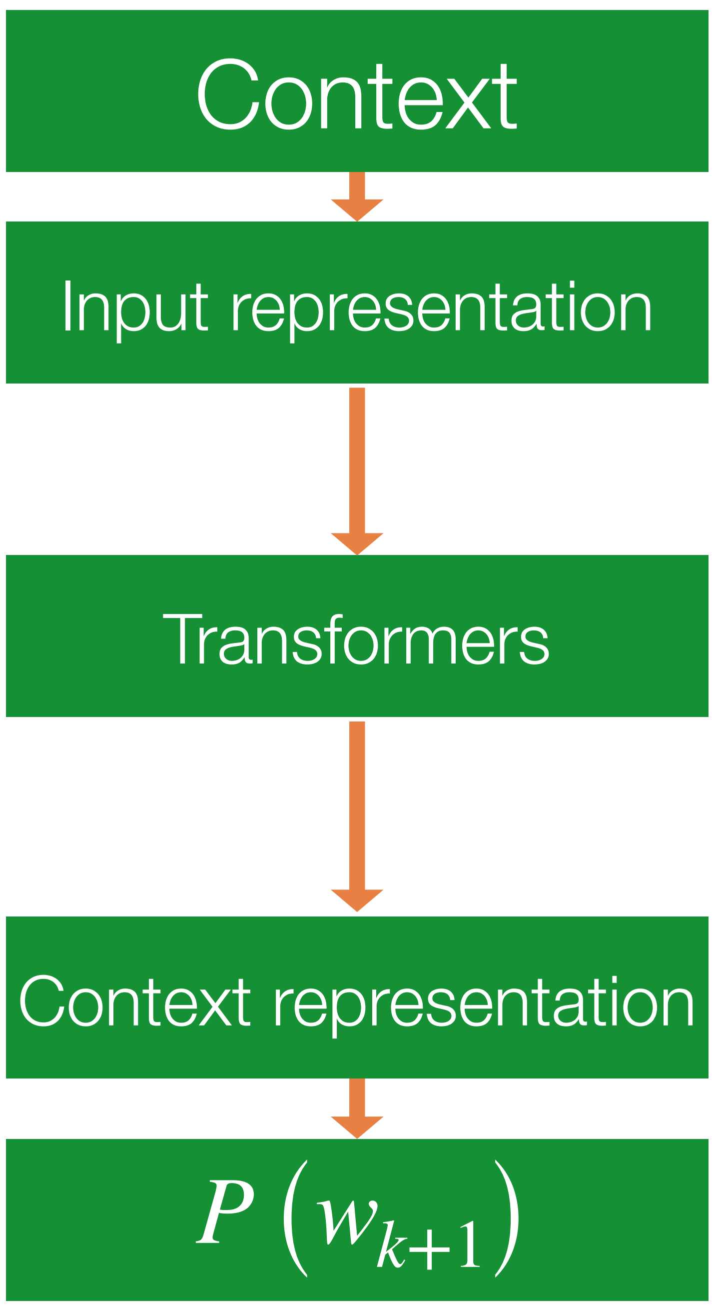
Matrix with word embeddings

Some mysterious neural network

Vector c representing the context

Probability distribution over the next word $P\left(w_{k+1}\right)$

A neural language model



Context of previous words $w_1, w_2, \ldots, w_k$

Matrix with word embeddings

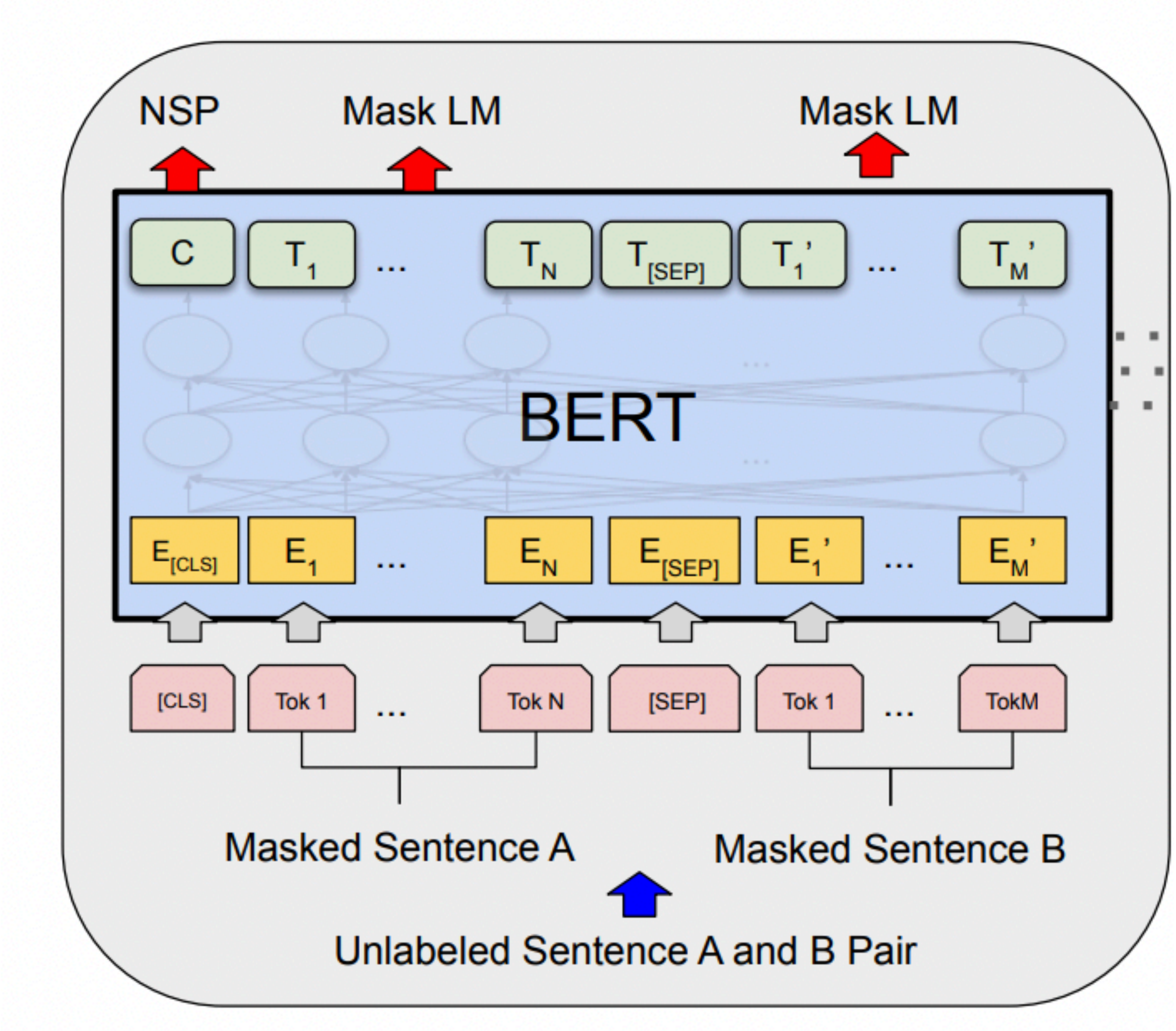
Several layers of multi-head attention
(+ some other things)

Vector c representing the context

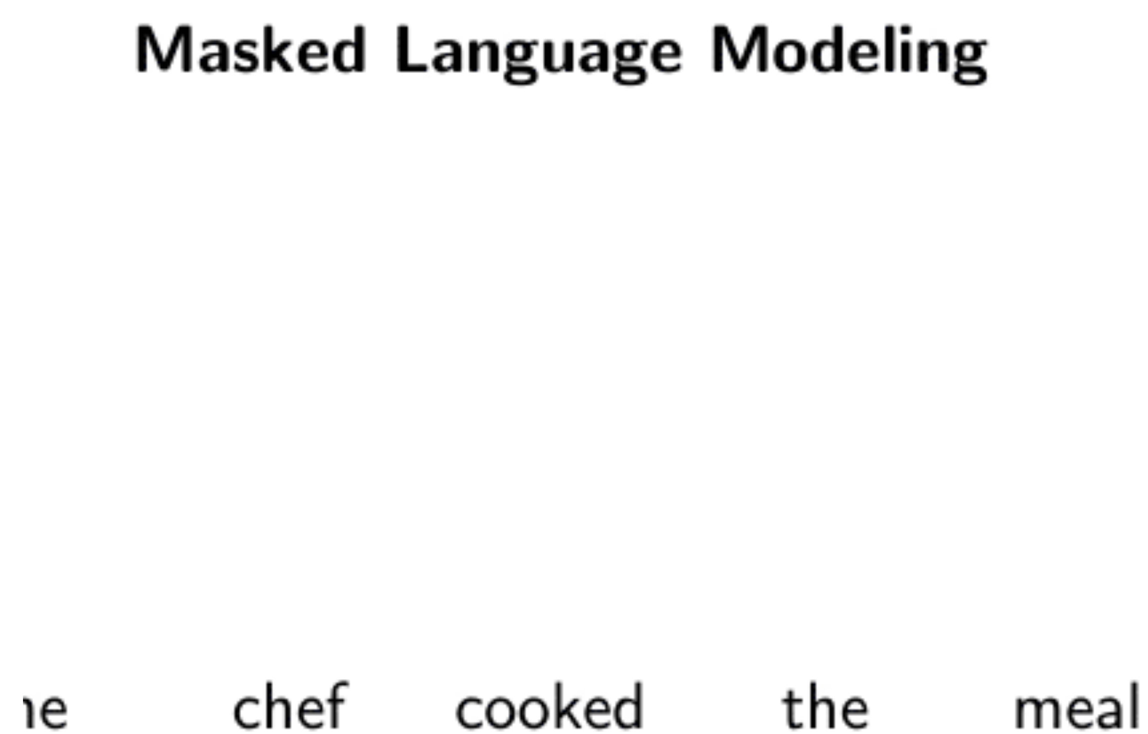
Probability distribution over the next word $P\left(w_{k+1}\right)$

BERT

Bidirectional Encoder Representations from Transformers (BERT)



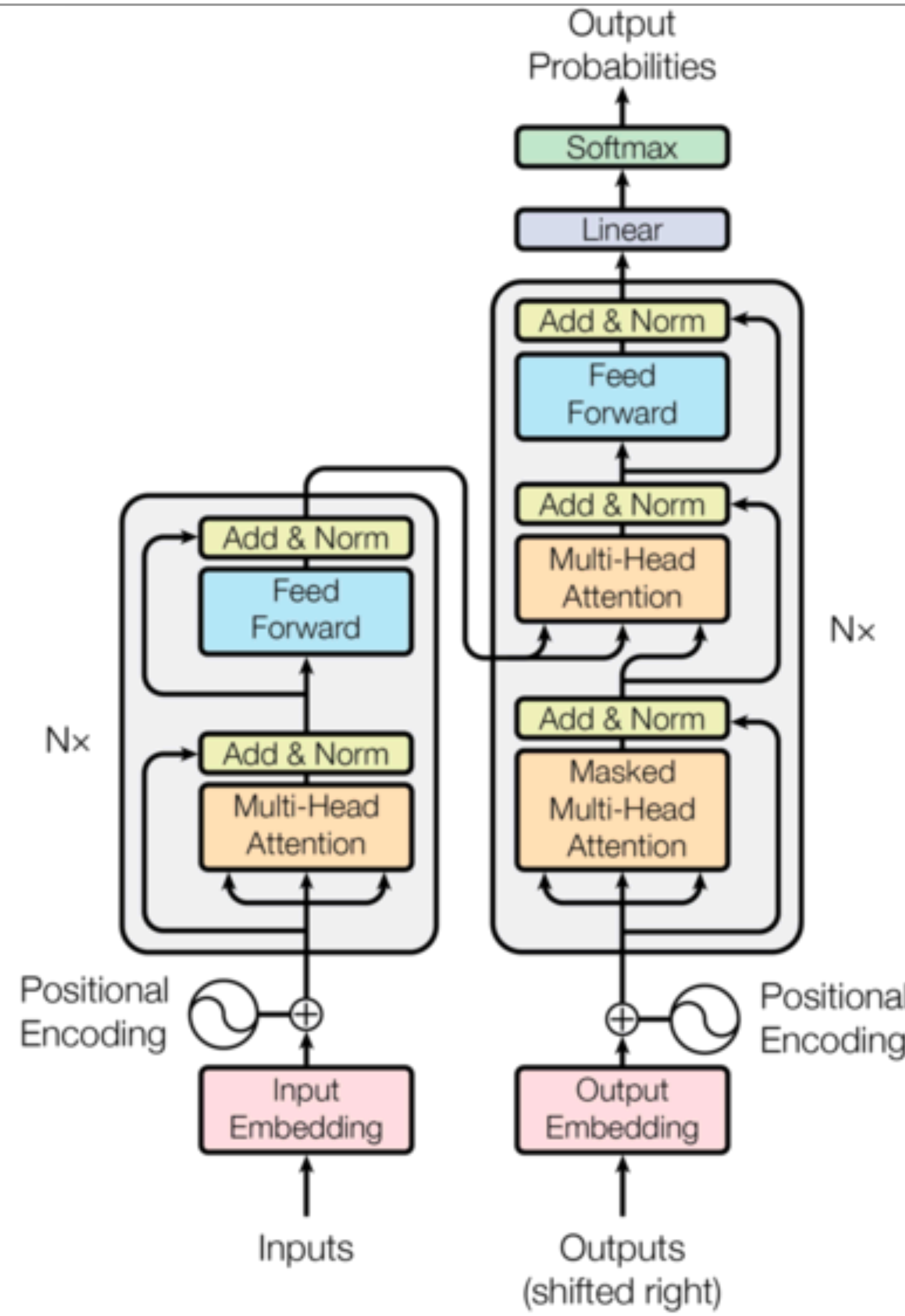
Training



Trained on:
BooksCorpus (800M words)
English Wikipedia (2,500M words).

GPT-3

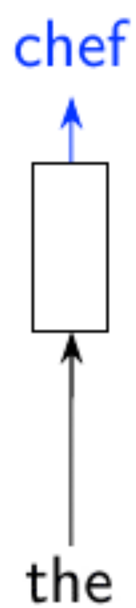
Generative Pretrained Transformer 3 (GPT-3)

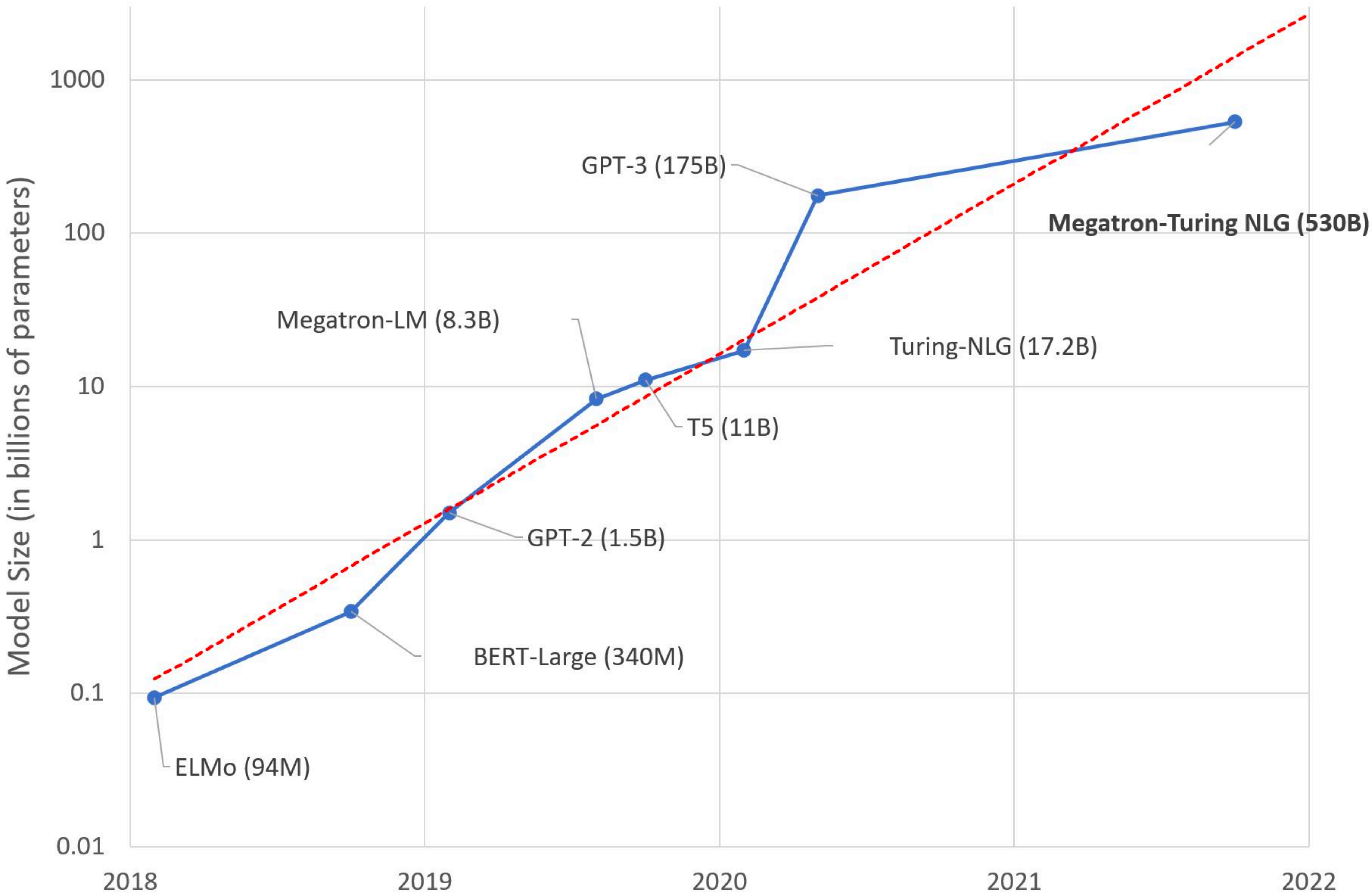


Trained on
~ 400B tokens!

Training

Language Modeling

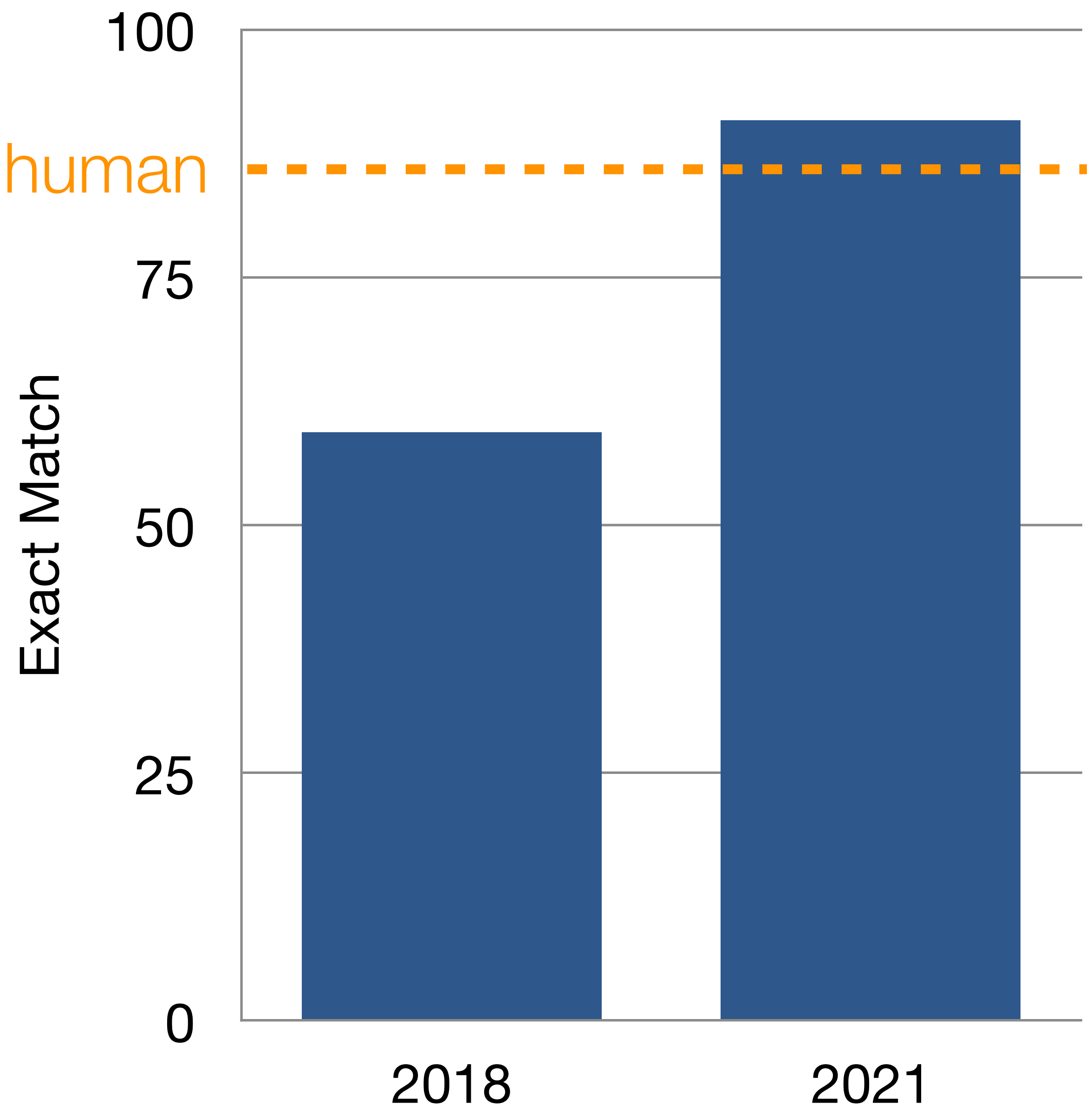




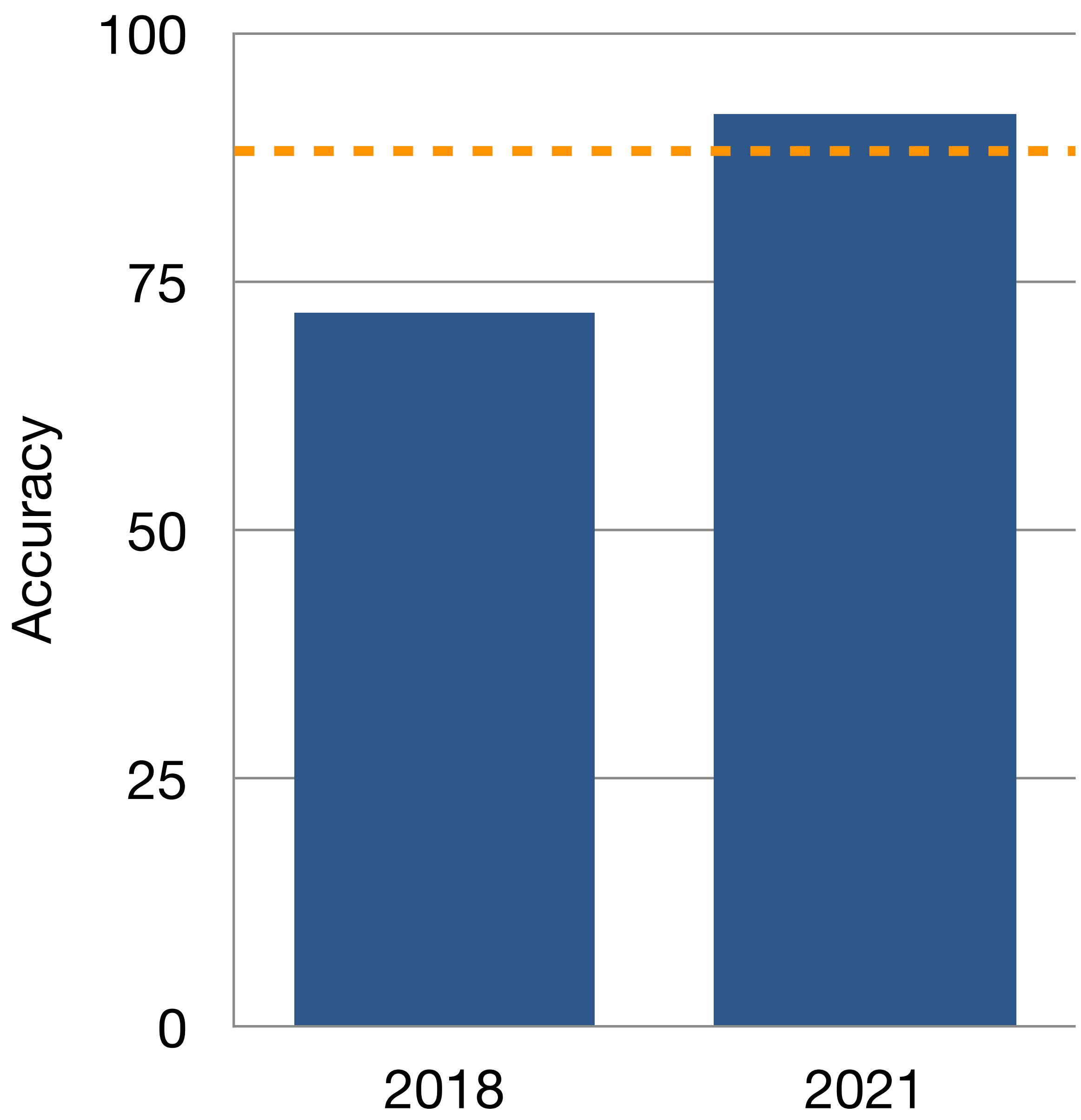
https://huggingface.co/blog/large-language-models

Progress in NLU

SQuAD 2.0



MNLI



Does this mean these models exhibit deep understanding abilities?!?