

Formal Techniques for Neural-symbolic Modeling: Lecture 2

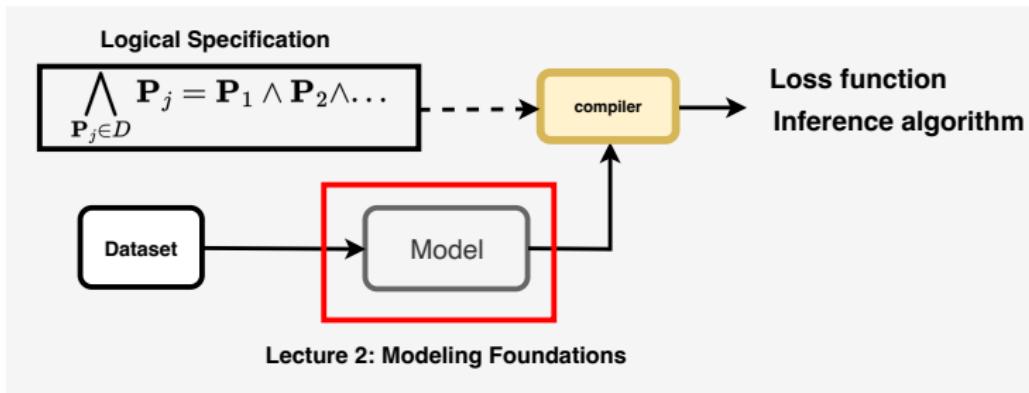
Kyle Richardson^a, Vivek Srikumar^b

Allen Institute for Artificial Intelligence (AI2)^a,
University of Utah^b

August 2023



Neural-symbolic modeling: big picture



Modeling with pre-trained transformers

USER Write me a poem about the ESSLLI summer school

ASSISTANT

In the heart of summer's golden glow,
Where curious minds wander and grow,
There lies a place of learning delight,
The haven that is ESSLLI, shining bright.

From distant lands they gather, young and old,
Seeking knowledge's treasure, untold.
Linguists, logicians, philosophers too,
Together they explore ideas anew.

Like a tapestry woven with care,
Each strand unique, yet all in flair.
Days filled with lectures and workshops vast,
Weaving knowledge, creating a future that lasts.

Modeling with pre-trained transformers

USER	Write me a poem about the ESSLLI summer school
ASSISTANT	<p>In the heart of summer's golden glow, Where curious minds wander and grow, There lies a place of learning delight, The haven that is ESSLLI, shining bright.</p>
	<p>From distant lands they gather, young and old, Seeking knowledge's treasure, untold. Linguists, logicians, philosophers too, Together they explore ideas anew.</p>
	<p>Like a tapestry woven with care, Each strand unique, yet all in flair. Days filled with lectures and workshops vast, Weaving knowledge, creating a future that lasts.</p>

How do these models (e.g., ChatGPT) work?

Goals for this lecture

1. What are *language models*? Classical approaches, limitations
2. What are *contextual models*? Differences with classical approaches
3. What is *attention* and how do *transformers* work?
4. How are these models trained and used in practice? Large-scale *pre-training* and *fine-tuning*.

Language Modeling Basics

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway is to the west of Sweden}$

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway}$ is to the west of Sweden

How to compute? The chain rule:

$$p(w_1, w_2, \dots, w_m) = \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}})$$

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway is to the west of Sweden}$

How to compute? The chain rule:

$$p(w_1, w_2, \dots, w_m) = \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}})$$

$$\begin{aligned} p(t) &= p(\text{Norway} \mid \overbrace{\text{<BOS>}}^{\text{start symbol}}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ &\quad p(\text{the} \mid \text{Norway is to}) \times p(\text{west} \mid \text{Norway is to the}) \times \\ &\quad p(\text{of} \mid \text{Norway is to the west}) \times p(\text{Sweden} \mid \text{Norway is to the west of}) \times \\ &\quad p(\overbrace{\text{<EOS>}}^{\text{end symbol}} \mid \text{Norway is to the west of Sweden}) \end{aligned}$$

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway is to the west of Sweden}$

How to compute? The chain rule:

$$p(w_1, w_2, \dots, w_m) = \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}})$$

$$\begin{aligned} p(t) = & p(\text{Norway} \mid \overbrace{\text{<BOS>}}^{\text{start symbol}}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ & p(\text{the} \mid \text{Norway is to}) \times p(\text{west} \mid \text{Norway is to the}) \times \\ & p(\text{of} \mid \text{Norway is to the west}) \times p(\text{Sweden} \mid \text{Norway is to the west of}) \times \\ & p(\overbrace{\text{<EOS>}}^{\text{end symbol}} \mid \text{Norway is to the west of Sweden}) \end{aligned}$$

Estimation: learning these probabilities from example text.

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway is to the west of Sweden}$

How to compute? The chain rule:

$$p(w_1, w_2, \dots, w_m) = \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}})$$

$$\begin{aligned} p(t) = & p(\text{Norway} \mid \overbrace{\text{<BOS>}}^{\text{start symbol}}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ & p(\text{the} \mid \text{Norway is to}) \times p(\text{west} \mid \text{Norway is to the}) \times \\ & p(\text{of} \mid \text{Norway is to the west}) \times p(\text{Sweden} \mid \text{Norway is to the west of}) \times \\ & p(\overbrace{\text{<EOS>}}^{\text{end symbol}} \mid \text{Norway is to the west of Sweden}) \end{aligned}$$

Is this a hard problem? Involves commonsense knowledge,

$$p(\text{Sweden} \mid \text{Norway is.. west of}) > p(\text{Iceland} \mid \text{Norway is..west of})$$

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway is to the west of Sweden}$

How to compute? The chain rule:

$$p(w_1, w_2, \dots, w_m) = \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}})$$

$$\begin{aligned} p(t) = & p(\text{Norway} \mid \overbrace{\text{<BOS>}}^{\text{start symbol}}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ & p(\text{the} \mid \text{Norway is to}) \times p(\text{west} \mid \text{Norway is to the}) \times \\ & p(\text{of} \mid \text{Norway is to the west}) \times p(\text{Sweden} \mid \text{Norway is to the west of}) \times \\ & p(\underbrace{\text{<EOS>}}_{\text{end symbol}} \mid \text{Norway is to the west of Sweden}) \end{aligned}$$

Is this a hard problem? Involves deep linguistic knowledge,

$$p(\text{him} \mid \text{John asked Mary to wash}) > p(\text{himself} \mid \text{John ... Mary to wash})$$

Basic Language Modeling

- ▶ **objective:** for text w_1, w_2, \dots, w_m , give $p(X_1 = w_1, X_2 = w_2, \dots, X_m = w_m)$

e.g., $t = \text{Norway is to the west of Sweden}$

How to compute? The chain rule:

$$p(w_1, w_2, \dots, w_m) = \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}})$$

$$\begin{aligned} p(t) = & p(\text{Norway} \mid \overbrace{\text{<BOS>}}^{\text{start symbol}}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ & p(\text{the} \mid \text{Norway is to}) \times p(\text{west} \mid \text{Norway is to the}) \times \\ & p(\text{of} \mid \text{Norway is to the west}) \times p(\text{Sweden} \mid \text{Norway is to the west of}) \times \\ & p(\overbrace{\text{<EOS>}}^{\text{end symbol}} \mid \text{Norway is to the west of Sweden}) \end{aligned}$$

Generation: Can be used to generate text (*decoding*).

Basic Language Modeling: the goals

- ▶ Given a **training** and **evaluation** dataset (consisting of texts $w_1, w_2, \dots w_n$), we can measure the following quantities:

Basic Language Modeling: the goals

- ▶ Given a **training** and **evaluation** dataset (consisting of texts w_1, w_2, \dots, w_n), we can measure the following quantities:

$$L = p(w_1, w_2, \dots, w_n) \quad \text{likelihood}$$

$$H = -\frac{1}{N} \log_2 L \quad (\text{per-word}) \text{ cross-entropy}$$

$$PP = 2^H \quad \text{perplexity}$$

Basic Language Modeling: the goals

- Given a **training** and **evaluation** dataset (consisting of texts w_1, w_2, \dots, w_n), we can measure the following quantities:

$$L = p(w_1, w_2, \dots, w_n) \quad \text{likelihood}$$

$$H = -\frac{1}{N} \log_2 L \quad (\text{per-word}) \text{ cross-entropy}$$

$$PP = 2^H \quad \text{perplexity}$$

Intuitively, taking N to be the vocabulary size, the perplexity of a uniform probability model, i.e.,

$$p(w_j \mid w_{<j}) = \frac{1}{N}$$

is equal to N , the *effective vocabulary size* under model.

Classical Language Modeling

How to estimate? **Count-based models**, collect counts from large collection of texts (maximum likelihood estimation), *discrete objects*

$$p(w_j \mid w_1, \dots, w_{j-1}) \propto \frac{\text{count}(w_1, \dots, w_{j-1}, w_j)}{\text{how often have I seen this } \underline{\text{discrete}} \text{ event in data?}}$$

Classical Language Modeling

How to estimate? **Count-based models**, collect counts from large collection of texts (maximum likelihood estimation), *discrete objects*

$$p(w_j \mid w_1, \dots, w_{j-1}) \propto \frac{\text{count}(w_1, \dots, w_{j-1}, w_j)}{\text{how often have I seen this } \underline{\text{discrete}} \text{ event in data?}}$$

Example with Sweden:

$$p(\text{Sweden} \mid \text{Norway is..west of}) = \frac{\text{count}(\text{Norway is...west of Sweden})}{\sum_w \text{count}(\text{Norway is...west of } w)}$$

Classical Language Modeling

How to estimate? **Count-based models**, collect counts from large collection of texts (maximum likelihood estimation), *discrete objects*

$$p(w_j \mid w_1, \dots, w_{j-1}) \propto \frac{\text{count}(w_1, \dots, w_{j-1}, w_j)}{\text{count}(w_1, \dots, w_{j-1})}$$

how often have I seen this discrete event in data?

Example with Sweden:

$$p(\text{Sweden} \mid \text{Norway is..west of}) = \frac{\text{count}(\text{Norway is...west of Sweden})}{\sum_w \text{count}(\text{Norway is...west of } w)}$$

Immediate Problems: Hard to obtain reliable counts for large contexts (*sparsity*), not feasible to store *all* possible contexts.

Classical Language Modeling

e.g., $t = \text{Norway}$ is to the west of Sweden

How to approximate? Markov assumption (order= N) (Shannon, 1948):

$$\begin{aligned} p(w_1, w_2, \dots, w_m) &= \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}}) \\ &\approx \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_{j-N+1}, \dots, w_{j-1}}_{\text{up to } N \text{ words before}}) \end{aligned}$$

Classical Language Modeling

e.g., $t = \text{Norway is to the west of Sweden}$

How to approximate? Markov assumption (order= N) (Shannon, 1948):

$$\begin{aligned} p(w_1, w_2, \dots, w_m) &= \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}}) \\ &\approx \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_{j-N+1}, \dots, w_{j-1}}_{\text{up to } N \text{ words before}}) \end{aligned}$$

$$\begin{aligned} p_{N=2}(t) &= p(\text{Norway} \mid \text{<BOS>}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ &\quad p(\text{the} \mid \text{is to}) \times p(\text{west} \mid \text{to the}) \\ &\quad p(\text{of} \mid \text{the west}) \times p(\text{Sweden} \mid \text{west of}) \times p(\text{<EOS>} \mid \text{west of Sweden}) \end{aligned}$$

Classical Language Modeling

e.g., $t = \text{Norway}$ is to the west of Sweden

How to approximate? Markov assumption (order= N) (Shannon, 1948):

$$\begin{aligned} p(w_1, w_2, \dots, w_m) &= \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}}) \\ &\approx \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_{j-N+1}, \dots, w_{j-1}}_{\text{up to } N \text{ words before}}) \end{aligned}$$

$$\begin{aligned} p_{N=2}(t) &= p(\text{Norway} \mid \text{<BOS>}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ &\quad p(\text{the} \mid \text{is to}) \times p(\text{west} \mid \text{to the}) \\ &\quad p(\text{of} \mid \text{the west}) \times p(\text{Sweden} \mid \text{west of}) \times p(\text{<EOS>} \mid \text{west of Sweden}) \end{aligned}$$

Observation: Still many parameters, e.g., for vocabulary of 100,000 words, 10^{15} (*quadrillion*) parameters (see Norvig (2012)).

Classical Language Modeling

e.g., $t = \text{Norway}$ is to the west of Sweden

How to approximate? Markov assumption (order= N) (Shannon, 1948):

$$\begin{aligned} p(w_1, w_2, \dots, w_m) &= \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_1, \dots, w_{j-1}}_{\text{what came before}}) \\ &\approx \prod_{j=1}^m p(\underbrace{w_j}_{\text{current word}} \mid \underbrace{w_{j-N+1}, \dots, w_{j-1}}_{\text{up to } N \text{ words before}}) \end{aligned}$$

$$\begin{aligned} p_{N=2}(t) &= p(\text{Norway} \mid \text{<BOS>}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ &\quad p(\text{the} \mid \text{is to}) \times p(\text{west} \mid \text{to the}) \\ &\quad p(\text{of} \mid \text{the west}) \times p(\text{Sweden} \mid \text{west of}) \end{aligned}$$

Early datasets: Google n-gram corpus. E.g., (Lin et al., 2012), 8 million books, 6% of all books published, around 500 billion tokens for English.

Classical Language Modeling: Why this doesn't work well

$$p_{N=2}(t) = p(\text{Norway} \mid \text{<BOS>}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ p(\text{the} \mid \text{is to}) \times p(\text{west} \mid \text{to the}) \\ p(\text{of} \mid \text{the west}) \times \underbrace{p(\text{Sweden} \mid \text{west of})}_{\text{what is to the west?}} \times \dots$$

Classical Language Modeling: Why this doesn't work well

$$p_{N=2}(t) = p(\text{Norway} \mid \text{<BOS>}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ p(\text{the} \mid \text{is to}) \times p(\text{west} \mid \text{to the}) \\ p(\text{of} \mid \text{the west}) \times \underbrace{p(\text{Sweden} \mid \text{west of})}_{\text{what is to the west?}} \times \dots$$

Syntactic Structures ([Chomsky, 1957](#))

Despite the undeniable interest and importance of semantic and statistical studies of language, they appear to have no direct relevance to the problem of determining and characterizing the set of grammatical utterances.... I think we are forced to conclude that... **probabilistic models give no particular insight into some of the basic problems of syntactic structure**

Classical Language Modeling: Why this doesn't work well

$$p_{N=2}(t) = p(\text{Norway} \mid \text{<BOS>}) \times p(\text{is} \mid \text{Norway}) \times p(\text{to} \mid \text{Norway is}) \times \\ p(\text{the} \mid \text{is to}) \times p(\text{west} \mid \text{to the}) \\ p(\text{of} \mid \text{the west}) \times \underbrace{p(\text{Sweden} \mid \text{west of})}_{\text{what is to the west?}} \times \dots$$

Syntactic Structures ([Chomsky, 1957](#))

Despite the undeniable interest and importance of semantic and statistical studies of language, they appear to have no direct relevance to the problem of determining and characterizing the set of grammatical utterances.... I think we are forced to conclude that... **probabilistic models give no particular insight into some of the basic problems of syntactic structure**

Technically: do not *faithfully* model complex joint probability distributions $p(w_1, \dots, w_m)$, independence assumptions, limited context.

The kinds of problems we started with

USER Write me a poem about the ESSLLI summer school

ASSISTANT In the heart of summer's golden glow,
Where curious minds wander and grow,
There lies a place of learning delight,
The haven that is ESSLLI, shining bright.

From distant lands they gather, young and old,
Seeking knowledge's treasure, untold.
Linguists, logicians, philosophers too,
Together they explore ideas anew.

Like a tapestry woven with care,
Each strand unique, yet all in flair.
Days filled with lectures and workshops vast,
Weaving knowledge, creating a future that lasts.

This would **never** work with classical models.

Language Models (LMs): Interim Summary

- ▶ **Language models:** Estimate the probability of word sequences, decomposed into next-word prediction, **generation**.

Language Models (LMs): Interim Summary

- ▶ **Language models:** Estimate the probability of word sequences, decomposed into next-word prediction, **generation**.
Classical Methods: words as discrete objects, independence assumptions (otherwise intractable), *do not faithfully model target distributions.*

Contextual Models

Continuous Representations of Linguistic Objects

- ▶ A big problem for classical methods: capturing linguistic **similarity** and other types of fuzziness encountered in language.

Continuous Representations of Linguistic Objects

- ▶ A big problem for classical methods: capturing linguistic **similarity** and other types of fuzziness encountered in language.
 - e.g., *Sweden* is closely related *Norway*, and related (to a lesser degree) to *Italy* and *Portugal*.

Continuous Representations of Linguistic Objects

- ▶ A big problem for classical methods: capturing linguistic **similarity** and other types of fuzziness encountered in language.
 - e.g., *Sweden* is closely related *Norway*, and related (to a lesser degree) to *Italy* and *Portugal*.

Old Solution: Represent linguistic objects as continuous vectors (see Widdows (2004)): $\vec{\text{sweden}} = [1., 3.]$, $\vec{\text{norway}} = [2, 2]$, $\vec{\text{italy}} = [6, .7.]$, ...

Continuous Representations of Linguistic Objects

- ▶ A big problem for classical methods: capturing linguistic **similarity** and other types of fuzziness encountered in language.
 - e.g., *Sweden* is closely related *Norway*, and related (to a lesser degree) to *Italy* and *Portugal*.

Old Solution: Represent linguistic objects as continuous vectors (see Widdows (2004)): $\vec{\text{sweden}} = [1., 3.]$, $\vec{\text{norway}} = [2, 2]$, $\vec{\text{italy}} = [6, .7.]$, ...

$$\text{SIMILARITY}(\vec{\text{sweden}}, \vec{\text{norway}}) = \sqrt{(2 - 1)^2 + (2 - 3)^2} \approx 1.41$$

$$\text{SIMILARITY}(\vec{\text{sweden}}, \vec{\text{italy}}) \approx 2.24$$

Word Vectors

```
1 import spacy ## to install: pip install spacy
2 ##download: python -m spacy download en_core_web_md
3 nlp = spacy.load("en_core_web_md")
4
5 norway = nlp("Norway")
6
7 ### norway word embedding
8 print(norway.vector)
9 #array([-7.2509e-01,  2.7890e-01,  2.6907e-01, ...,
10 #        4.7657e-02, -6.3465e-02,  6.3940e-01, ...,
11 #       -9.1360e-01, -5.9481e-01, -1.0941e-01, ...,
12 #      -3.0875e-01, -4.5818e-01,  1.1814e+00, ...,
13 #      4.3488e-01, -9.0963e-02,  5.9045e-01, ...,
14 #     -4.2597e-01, -3.6353e-01, -8.1080e-02, ...,
15 #      4.4343e-01,  1.1410e-01,  5.4636e-01, ...,
16 #      6.6868e-02, -2.0587e-01, -5.9080e-02, ...,
17 #      2.7198e-01, -2.7745e-02,  1.4556e-01, ...,
18 #     -3.6832e-01,  3.1339e-01, -2.3456e-01, ...,
19 #     -5.9530e-01,  1.1208e-01, -1.0553e-01, ...,
```

Word Vectors

```
1 import spacy ## to install: pip install spacy
2 ##download: python -m spacy download en_core_web_md
3 nlp = spacy.load("en_core_web_md")
4
5 norway = nlp("Norway")
6
7 ### norway word embedding
8 print(norway.vector)
9 array([-7.2509e-01,  2.7890e-01,  2.6907e-01, ...,
10 #          4.7657e-02, -6.3465e-02,  6.3940e-01, ...,
11 #         -9.1360e-01, -5.9481e-01, -1.0941e-01, ...,
12 #        -3.0875e-01, -4.5818e-01,  1.1814e+00, ...,
13 #         4.3488e-01, -9.0963e-02,  5.9045e-01, ...,
14 #        -4.2597e-01, -3.6353e-01, -8.1080e-02, ...,
15 #         4.4343e-01,  1.1410e-01,  5.4636e-01, ...,
16 #         6.6868e-02, -2.0587e-01, -5.9080e-02, ...,
17 #         2.7198e-01, -2.7745e-02,  1.4556e-01, ...,
18 #        -3.6832e-01,  3.1339e-01, -2.3456e-01, ...,
19 #        -5.9530e-01,  1.1208e-01, -1.0553e-01, ...,
```

`norway.vector`, general-purpose (non-contextual) representation of *Norway* learned from data (e.g., [Mikolov et al. \(2013\)](#)), useful for similarity.

Word Vectors

```
1 import spacy ## to install: pip install spacy
2 ##download: python -m spacy download en_core_web_md
3 nlp = spacy.load("en_core_web_md")
4
5 norway = nlp("Norway")
6
7 ### norway word embedding
8 print(norway.vector)
9 array([-7.2509e-01,  2.7890e-01,  2.6907e-01, ...,
10 #          4.7657e-02, -6.3465e-02,  6.3940e-01, ...,
11 #         -9.1360e-01, -5.9481e-01, -1.0941e-01, ...,
12 #        -3.0875e-01, -4.5818e-01,  1.1814e+00, ...,
13 #         4.3488e-01, -9.0963e-02,  5.9045e-01, ...,
14 #        -4.2597e-01, -3.6353e-01, -8.1080e-02, ...,
15 #         4.4343e-01,  1.1410e-01,  5.4636e-01, ...,
16 #         6.6868e-02, -2.0587e-01, -5.9080e-02, ...,
17 #         2.7198e-01, -2.7745e-02,  1.4556e-01, ...,
18 #        -3.6832e-01,  3.1339e-01, -2.3456e-01, ...,
19 #        -5.9530e-01,  1.1208e-01, -1.0553e-01, ...,
```

semantically: approximates the *intensional* meaning of words, the sense of *Norway*.

Beyond Static Word Vectors

```
1 ## pip install sentence-transformers
2 from sentence_transformers import SentenceTransformer
3 model = SentenceTransformer("all-MiniLM-L6-v2")
4
5 sentence=["Norway is to the west of Sweden"]
6 sentence_vector = model.encode(sentence)
7 print(sentence_vector)
8 #[array([[ 8.98966566e-02,  7.80573040e-02, ...,
9 #          -5.28105311e-02,  1.03836119e-01, ...,
10 #         -3.54157202e-02, -2.84450850e-03, ...,
11 #          3.65298055e-02,  1.44437077e-02, ...,
12 #         -1.59231629e-02, -2.63415072e-02, ...,
13 #          -4.15021786e-03, -3.24299373e-02, ...,
14 #          4.20359559e-02,  7.27154762e-02, ...,
15 #         -2.55496521e-02,  4.76241782e-02, ...,
16 #          1.28677487e-01,  7.31618553e-02, ...,
17 #         -5.24449795e-02,  1.07175205e-02, ...,
18 #          -4.30012196e-02, -5.51163033e-02, ...,
19 #          6.70064837e-02, -6.46066740e-02, ...]
```

Recent NLP: learning embedding representations for more complex linguistic objects.

Consequences of moving from discrete to continuous...

- ▶ Models can make certain *unforgivable* mistakes.

Consequences of moving from discrete to continuous...

- Models can make certain *unforgivable* mistakes.

Text Normalization (Sproat and Jaitly, 2016)

Input	Correct	Prediction
60	sixty	six
82.55 mm	eighty two point five five millimeters	eighty two one five five meters
2 mA	two milliamperes	two units
£900 million	nine hundred million pounds	nine hundred million euros
16 см	шестнадцати сантиметров	sil сантиметров
16 cm	sixteen centimeters	sil centimeters
-11	минус одиннадцать	минус один Similar representations

Early Neural Translation (Arthur et al., 2016)

Input:	I come from Tunisia.
Reference:	チュニジアの出身です。 <u>Chunisia</u> no shusshindesu. (I'm from Tunisia.)
System:	ノルウェーの出身です。 <u>Norue-</u> no shusshindesu. (I'm from Norway.)

Consequences of moving from discrete to continuous...

- Models can make certain *unforgivable* mistakes.

Text Normalization (Sproat and Jaitly, 2016)

Input	Correct	Prediction
60	sixty	six
82.55 mm	eighty two point five five millimeters	eighty two one five five meters
2 mA	two milliamperes	two units
£900 million	nine hundred million pounds	nine hundred million euros
16 см	шестнадцати сантиметров	sil сантиметров
16 cm	sixteen centimeters	sil centimeters
-11	минус одиннадцать	минус один Similar representations

Early Neural Translation (Arthur et al., 2016)

Input:	I come from Tunisia.
Reference:	チュニジアの出身です。 <u>Chunisia</u> no shusshindesu. (I'm from Tunisia.)
System:	ノルウェーの出身です。 <u>Noruee-</u> no shusshindesu. (I'm from Norway.)

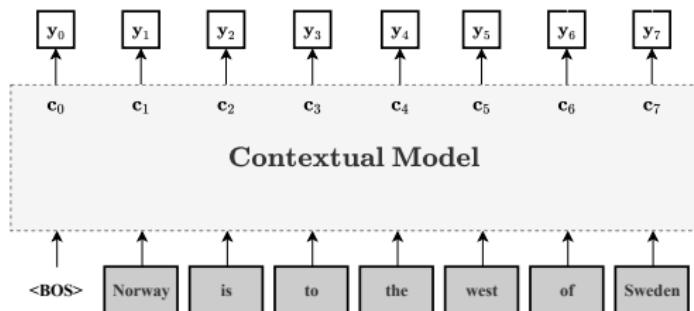
The only way this can work: learning good representations from data.

Contextual Models

- ▶ **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.

Contextual Models

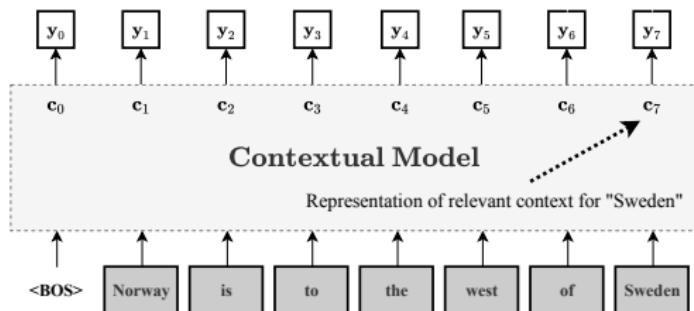
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Operationally: neural network models, often large and opaque.

Contextual Models

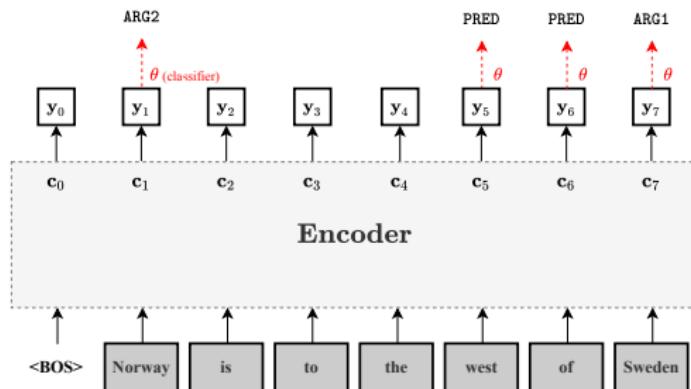
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Semantics: \mathbf{y}_7 captures the meaning of *Sweden* grounded in this particular sentential context, output of a compositional procedure.

Contextual Models

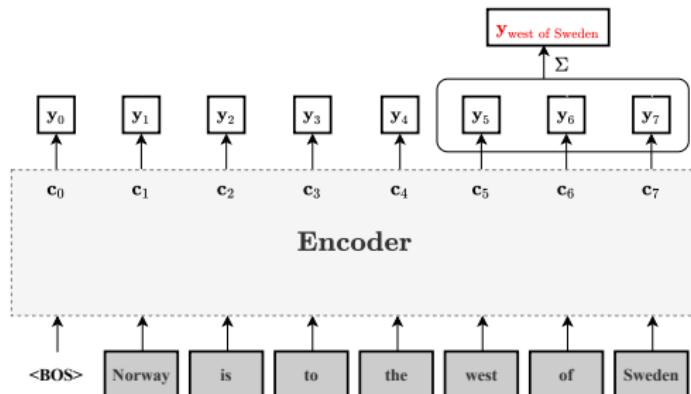
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Why? Build representations that allow you make predictions, **success:** learned representations that effectively solve problems.

Contextual Models

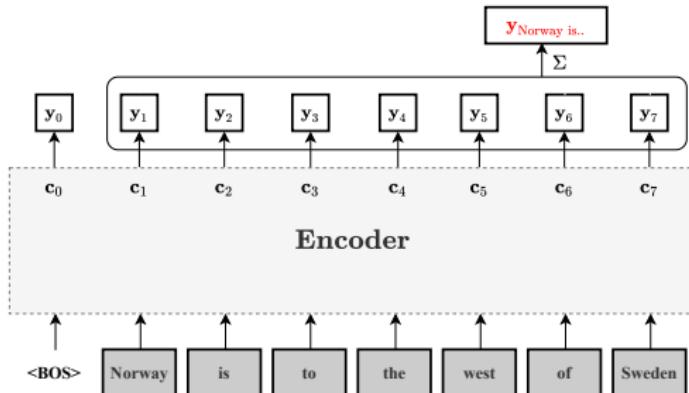
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Why? Build representations that allow you make predictions, **success:** learned representations that effectively solve problems.

Contextual Models

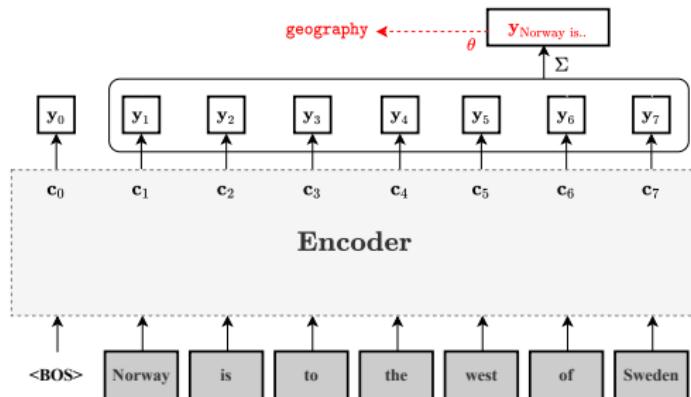
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Why? Build representations that allow you make predictions, **success:** learned representations that effectively solve problems.

Contextual Models

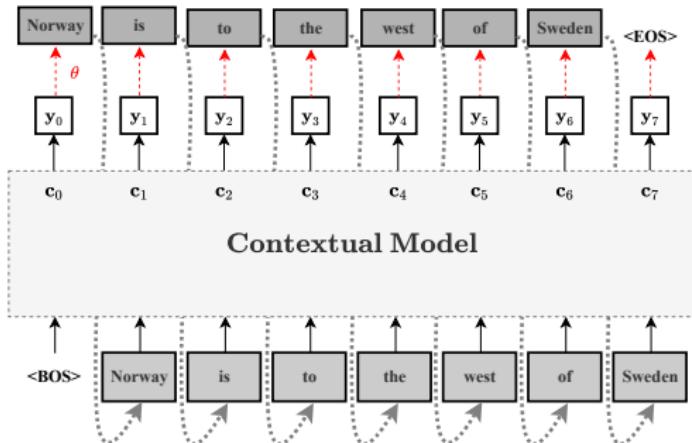
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Why? Build representations that allow you make predictions, **success:** learned representations that effectively solve problems.

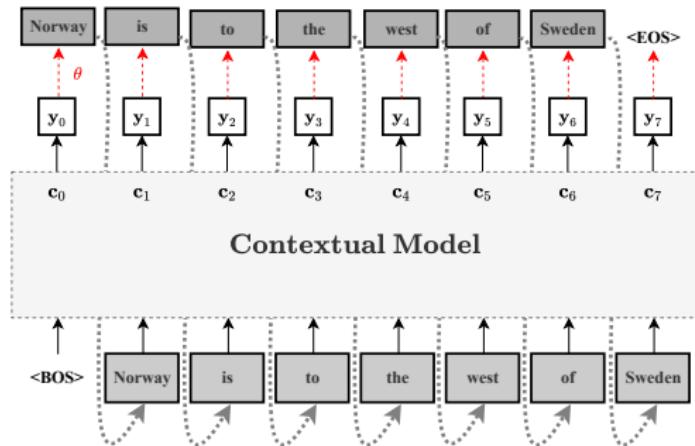
Contextual Models

- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Contextual Models

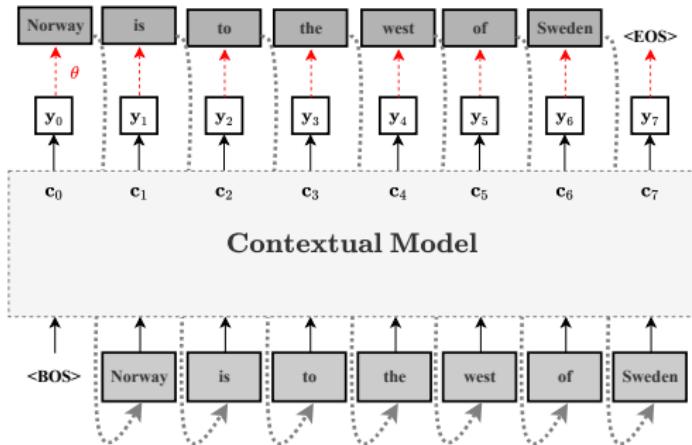
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



as LMs: $p(w_j | w_1, \dots, w_{j-1}) = p(w_j | \mathbf{c}_{j-1})$, Important: Can condition on full contexts, faithfully model complex joint probability distributions

Contextual Models

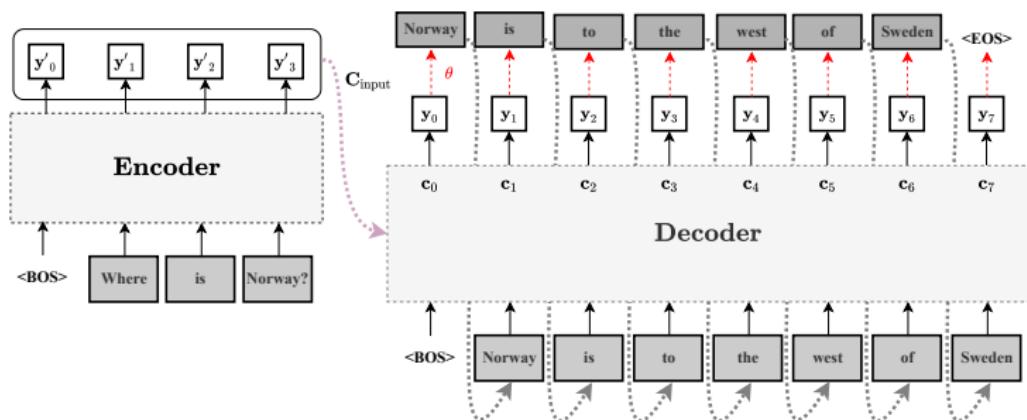
- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Model Architectures: how information is processed, internal representations are constructed. **Common:** RNNs, Transformers.

Contextual Models

- **Role:** assign continuous vectors representations $\mathbf{y}_j \in \mathbb{R}^d$ to elements in a sequence that capture their meaning of those elements in context.



Text2Text models: estimate $p(y^{\text{output}} | x^{\text{input}})$, natural way to express many language understanding problems.

Contextual Models: Interim Summary

- ▶ **Contextual Model:** builds continuous representations for sequential input, neural networks, faithfully models full context

Contextual Models: Interim Summary

- ▶ **Contextual Model:** builds continuous representations for sequential input, neural networks, faithfully models full context

Used to build *contextual language models* and variants: *text2text models* (**encoders** vs. **decoders**).

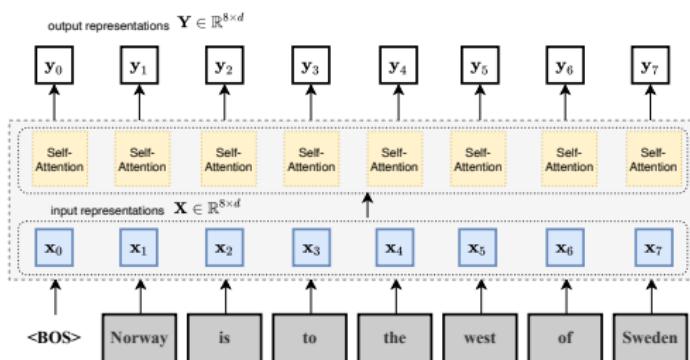
Contextual Models: Interim Summary

- ▶ **Contextual Model:** builds continuous representations for sequential input, neural networks, faithfully models full context
 - Used to build *contextual language models* and variants: *text2text models* (**encoders** vs. **decoders**).
 - essential questions (architecture):** how does information flow? how are representations built?

Attention-based Models

Self-Attention: Simple Encoder Example

Attention: Mechanism for building contextual representations (see [Vaswani et al. \(2017\)](#)).

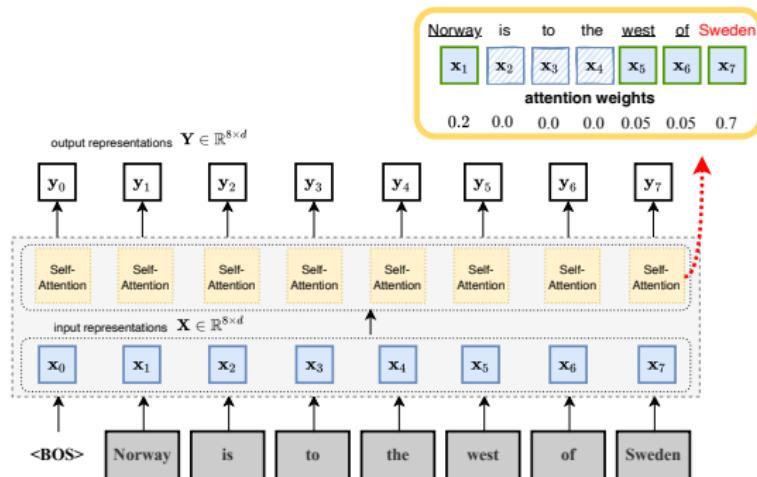


Self-Attention: Embeddings Representations

```
1 import torch ## to install: pip install pytorch
2
3 ### word embedding parameters and matrix E
4 E = torch.nn.Embedding(
5     embedding_dim=768, ##<--- dimensionality
6     num_embeddings=3000, ##<--- # words
7 )
8
9 ##e.g., Representation of our Input
10 X = E(torch.tensor([
11     0, # <BOS>
12     1, # Norway
13     2, # is
14     3, # to
15     4, # the
16     5, # west
17     6, # of
18     7, # Sweden
19 ])) ### => matrix 7 * 768
20
21 x1 = X[1] ##-> initial representation of Norway
22 print(x1)
23 ##### [-1.1344e+00, -3.0359e-01, 7.3585e-02, ....]
```

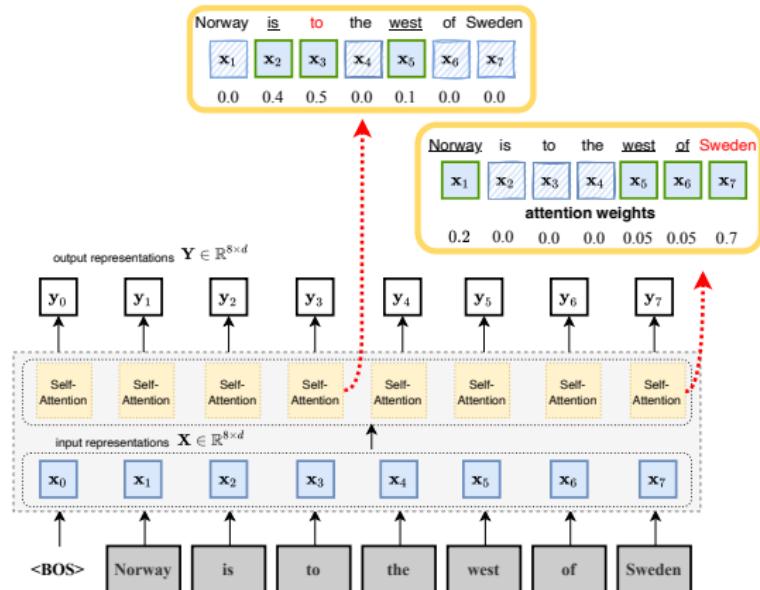
Self-Attention: Simple Encoder Example

Attention: Mechanism for building contextual representations (see [Vaswani et al. \(2017\)](#)).



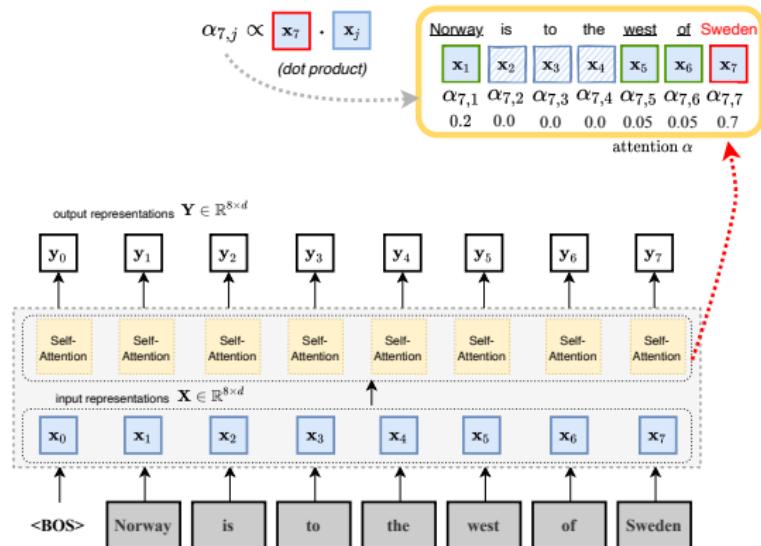
Self-Attention: Simple Encoder Example

Attention: Mechanism for building contextual representations (see [Vaswani et al. \(2017\)](#)).



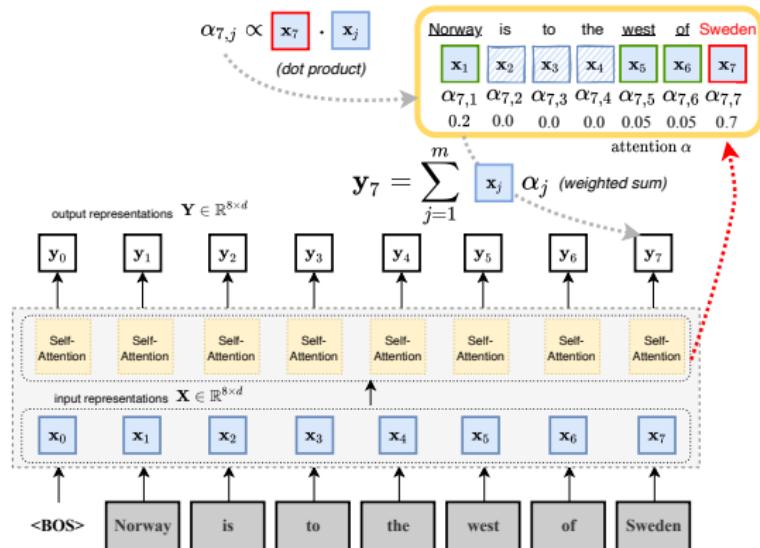
Self-Attention: Simple Encoder Example

Attention: Mechanism for building contextual representations (see [Vaswani et al. \(2017\)](#)).



Self-Attention: Simple Encoder Example

Attention: Mechanism for building contextual representations (see [Vaswani et al. \(2017\)](#)).



Self-Attention: Computing Final Representations

The full computation

$$\alpha'_{i,j} = \mathbf{x}_i \cdot \mathbf{x}_j \quad \text{dot product}$$

$$\alpha_{i,j} = \frac{e^{\alpha'_{i,j}}}{\sum_j e^{\alpha'_{i,j}}} \quad \text{softmax}$$

$$\mathbf{y}_i = \sum_j \alpha_{i,j} \mathbf{x}_j \quad \text{weighted sum}$$

Self-Attention: Computing Final Representations

The full computation

$$\alpha'_{i,j} = \mathbf{x}_i \cdot \mathbf{x}_j \quad \text{dot product}$$

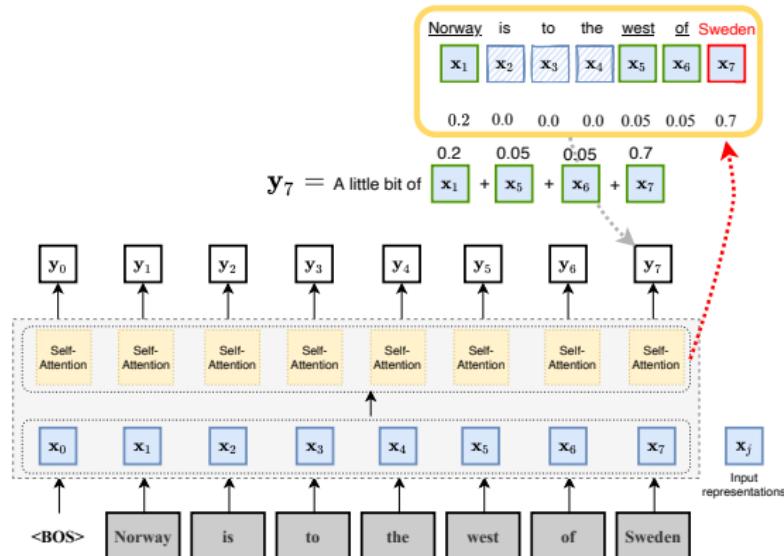
$$\alpha_{i,j} = \frac{e^{\alpha'_{i,j}}}{\sum_j e^{\alpha'_{i,j}}} \quad \text{softmax}$$

$$\mathbf{y}_i = \sum_j \alpha_{i,j} \mathbf{x}_j \quad \text{weighted sum}$$

```
1 ## Input representations (again),
2 X = torch.tensor([0,1,2,3,4,5,6,7])
3
4 ### raw weights (dot product / matrix multiplication)
5 raw_Alpha = torch.matmul(X,X.transpose(0,1))
6 ### normalized via softmax, probability distribution
7 alpha = raw_Alpha.softmax(dim=-1)
8 ### Final self attention representations
9 Y = torch.matmul(alpha,X)
10 ### self attention representation of 'Norway'
11 y1 = Y[1]
```

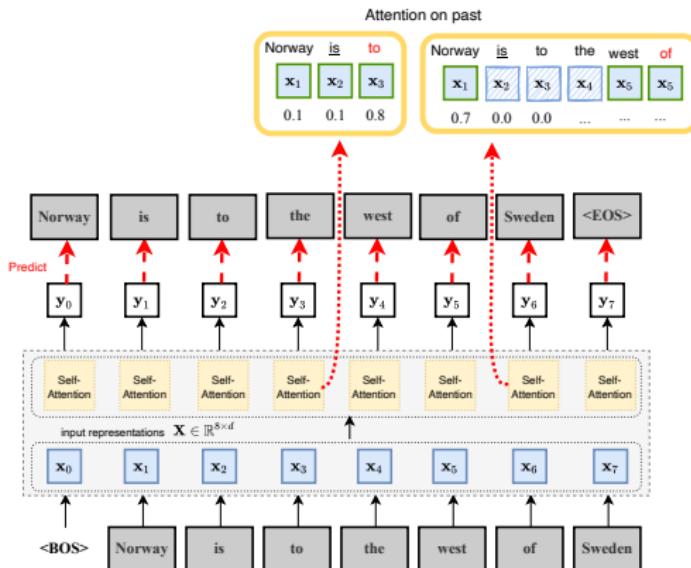
Self-Attention: Intuition

Attention: A kind of brute-force looking around and aggregation of contextual information.



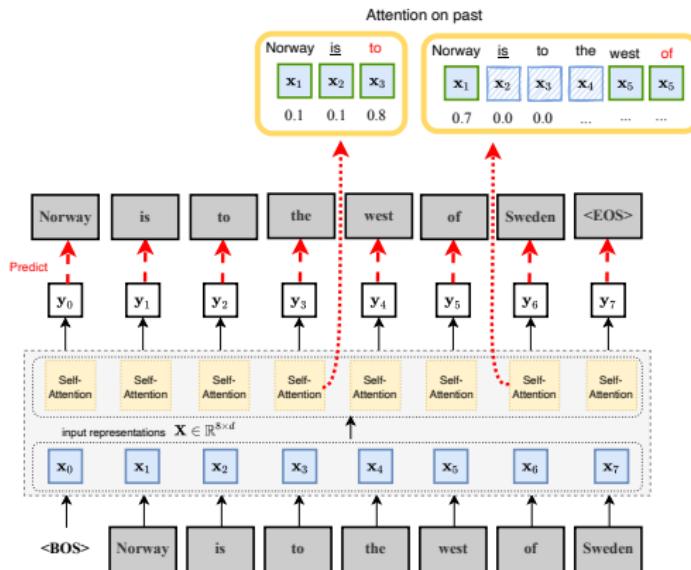
Attention in Other Contexts

Decoders: Attention limited to past context, allows for generation (step-by-step prediction of next word).



Attention in Other Contexts

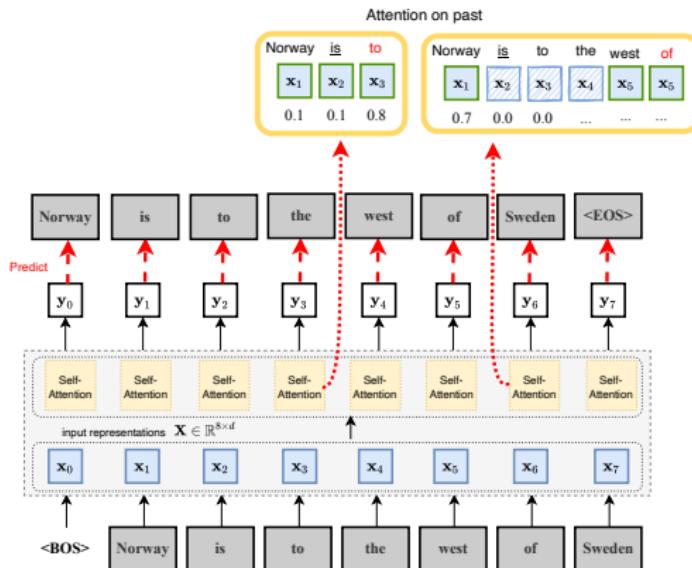
Decoders: Attention limited to past context, allows for generation (step-by-step prediction of next word).



Note: No independence assumptions, full context.

Attention in Other Contexts

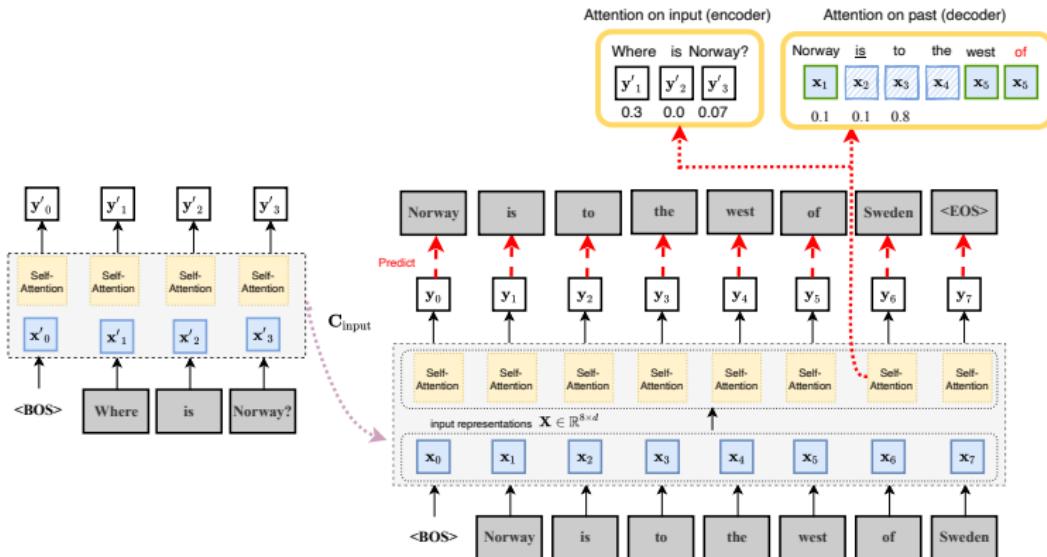
Decoders: Attention limited to past context, allows for generation (step-by-step prediction of next word).



Additional component: Classifier for predicting words from y_j .

Attention in Other Contexts

Text2Text: Additional attention on an input (*cross attention*)



Causal Attention

- ▶ Implementing causal attention, involves adding a **mask**.

```
1 import torch
2 ## Input representations (again),
3 X = E(torch.tensor([0,1,2,3,4,5,6,7]))
4
5 ### raw weights (dot product / matrix multiplication)
6 raw_Alpha = torch.matmul(X,X.transpose(0,1))
7
8 ### causal mask
9 mask = torch.triu_indices(8, 8, offset=1)
10 raw_Alpha[:, mask[0], mask[1]] = float('-inf')
11
12 ### normalized via softmax, probability distribution
13 alpha = raw_Alpha.softmax(dim=-1)
14 ### Final self attention representations
15 Y = torch.matmul(alpha,X)
16 ### self attention representation of 'Norway'
17 y1 = Y[1]
```

Why this might work

USER

Write me a poem about the ESSLLI summer school

ASSISTANT

In the heart of summer's golden glow,
Where curious minds wander and grow,
There lies a place of learning delight,
The haven that is ESSLLI, shining bright.

From distant lands they gather, young and old,
Seeking knowledge's treasure, untold.
Linguists, logicians, philosophers too,
Together they explore ideas anew.

Like a tapestry woven with care,
Each strand unique, yet all in flair.
Days filled with lectures and workshops vast,
Weaving knowledge, creating a future that lasts.

Why this might work

USER	Write me a poem about the ESSLLI summer school
ASSISTANT	<p>In the heart of summer's golden glow, Where curious minds wander and grow, There lies a place of learning delight, The haven that is ESSLLI, shining bright.</p> <p>From distant lands they gather, young and old, Seeking knowledge's treasure, untold. Linguists, logicians, philosophers too, Together they explore ideas anew.</p> <p>Like a tapestry woven with care, Each strand unique, yet all in flair. Days filled with lectures and workshops vast, Weaving knowledge, creating a future that lasts.</p>

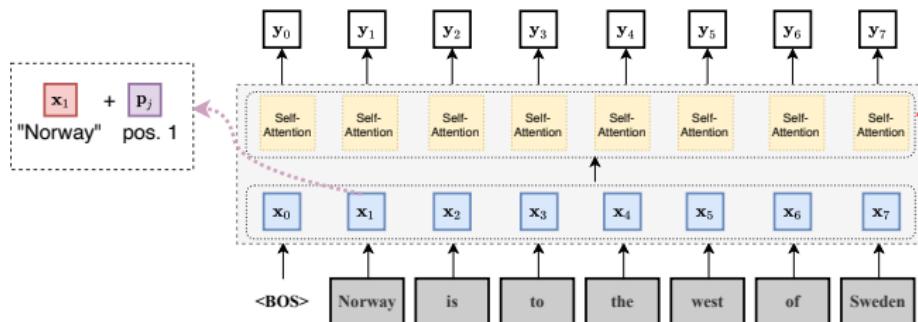
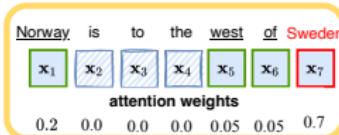
With attention, we can more faithfully model complex joint probability distributions, full context can be taken into account.

Current Transformers

An important detail: positional information

- Word embeddings so far do not encode position information.

$$\begin{aligned} \mathbf{x}_j & \quad \mathbf{E} \in \mathbb{R}^{|words| \times d} \text{ (word embeddings)} \\ \mathbf{p}_j & \quad \mathbf{P} \in \mathbb{R}^{p \times d} \quad \text{(position embeddings)} \\ \mathbf{x}_j & = \mathbf{x}_{id} + \mathbf{p}_j \end{aligned}$$



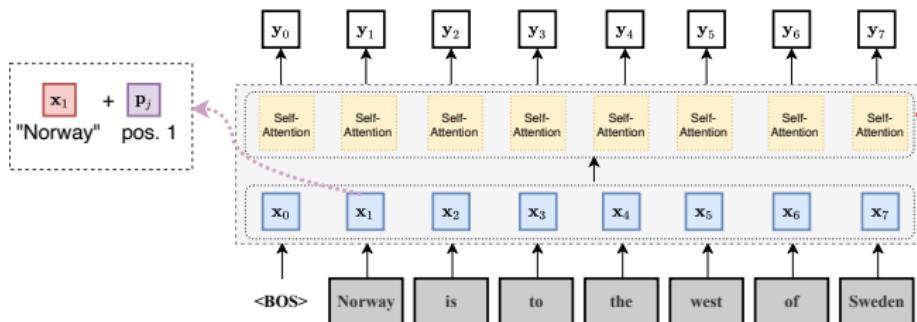
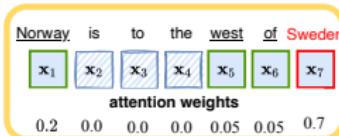
An important detail: positional information

- Word embeddings so far do not encode position information.

$$\mathbf{x}_j \in \mathbb{R}^{|words| \times d} \text{ (word embeddings)}$$

$$\mathbf{p}_j \in \mathbb{R}^{p \times d} \text{ (position embeddings)}$$

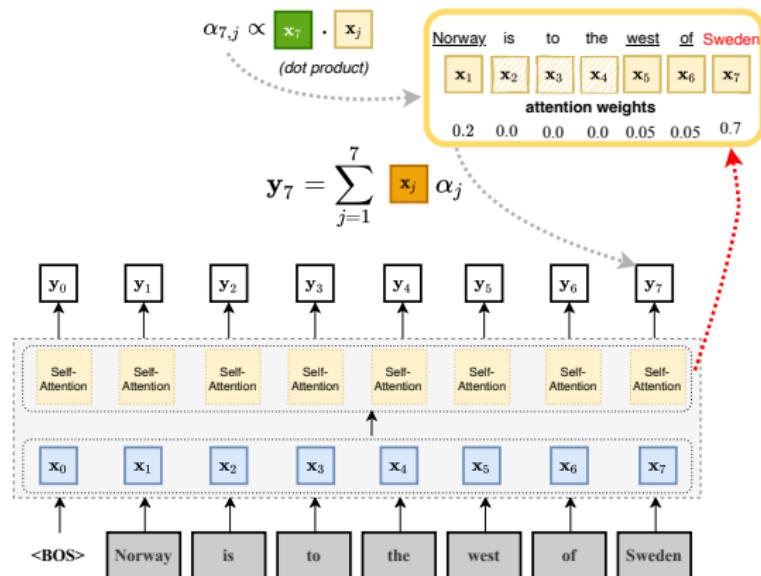
$$\mathbf{x}_j = \mathbf{x}_{id} + \mathbf{p}_j$$



Note: we can add any additional information we want, many variations.

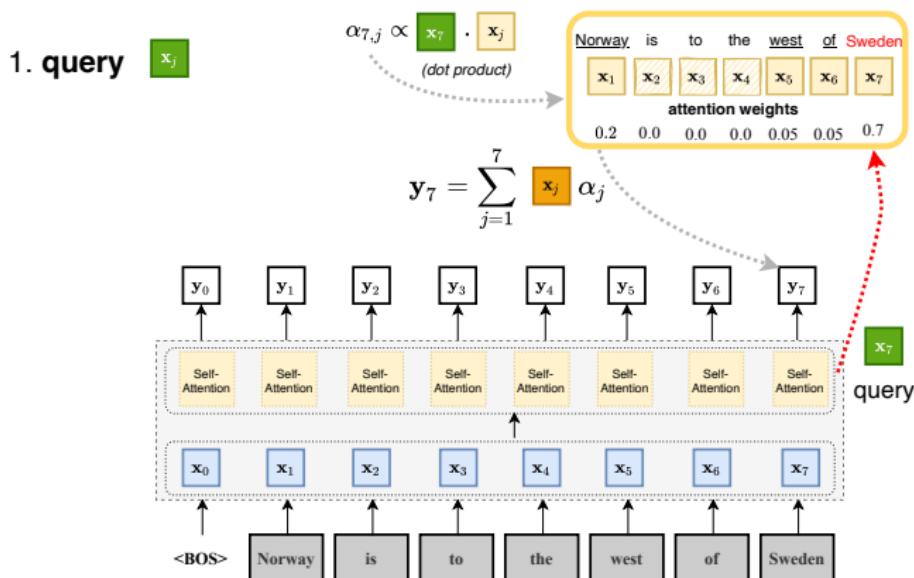
More Parameters: Keys, Queries and Values

- ▶ Three components in our computation.



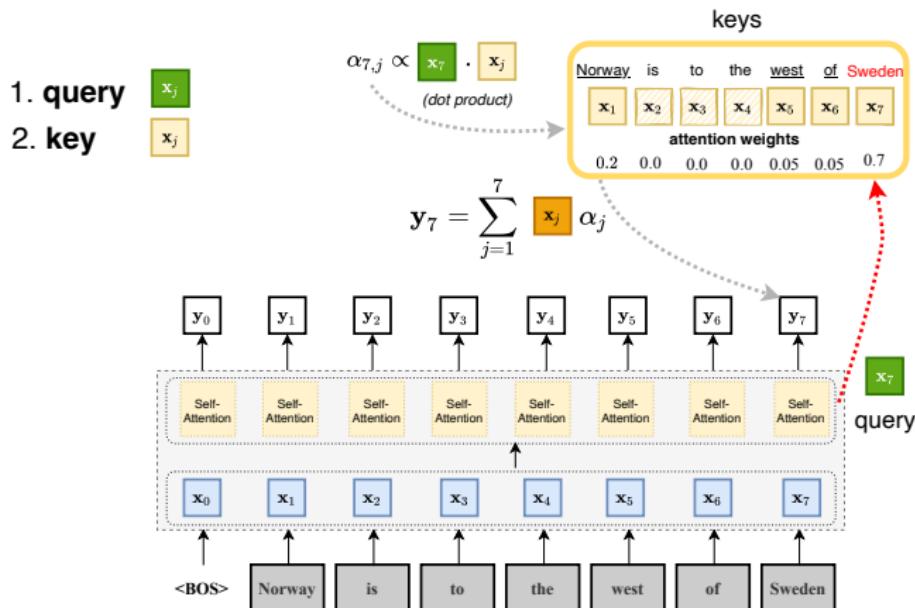
More Parameters: Keys, Queries and Values

- ▶ Three components in our computation.



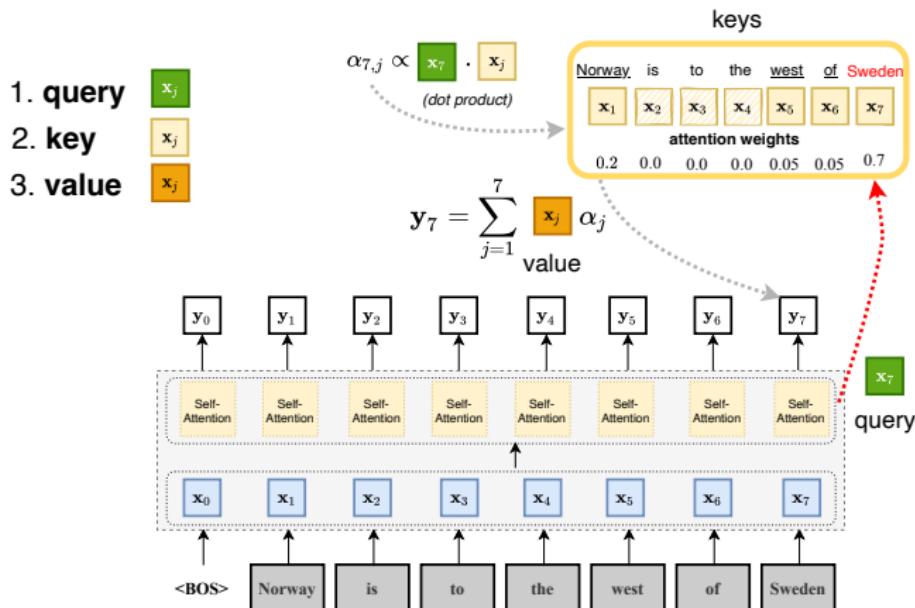
More Parameters: Keys, Queries and Values

- ▶ Three components in our computation.



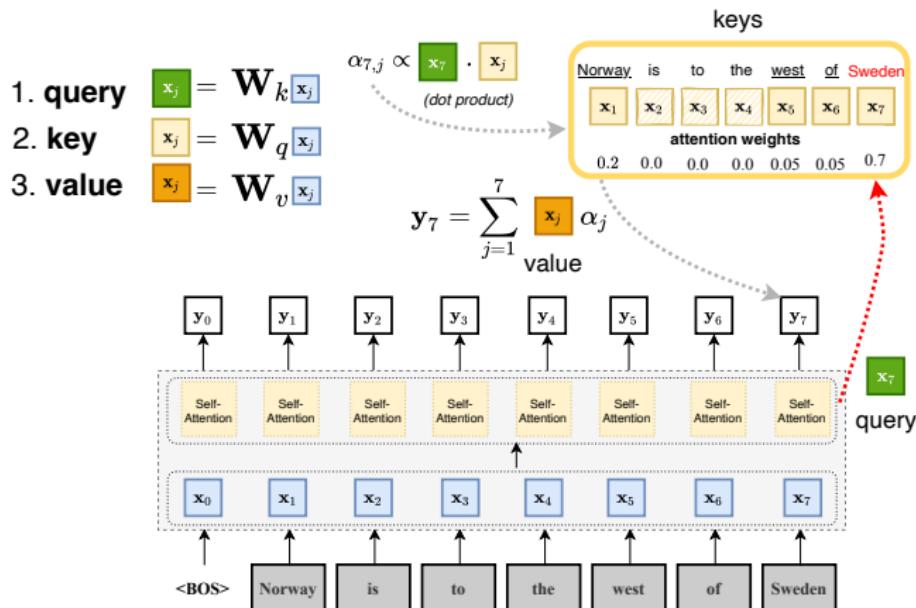
More Parameters: Keys, Queries and Values

- ▶ Three components in our computation.



More Parameters: Keys, Queries and Values

- ▶ Three components in our computation.



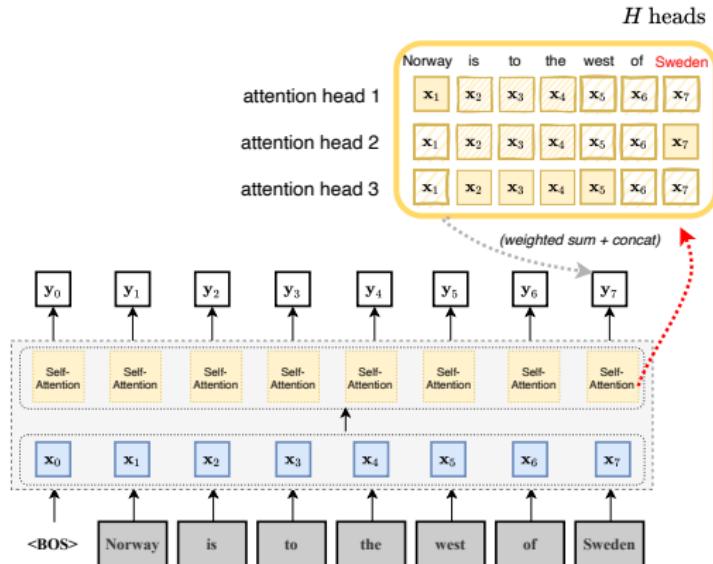
More Parameters: Keys, Queries and Values

Another few additional lines of PyTorch

```
1 ## Input representations (again),
2 X = E(torch.tensor([0,1,2,3,4,5,6,7]))
3 D = 768
4
5 ### key, value, query parameters (linear layer)
6 W_k    = torch.nn.Linear(D, D, bias=False)
7 W_q    = torch.nn.Linear(D, D, bias=False)
8 W_v    = torch.nn.Linear(D, D, bias=False)
9
10 key_rep   = W_k(X) #<-- rep. of X as keys
11 query_rep = W_q(X) #<-- rep. of X as queries
12 value_rep = W_v(X) #<-- rep of X as values
13
14 ### sample computation with parameters applied over 'X'
15 alpha = torch.matmul(
16     query_rep,
17     key_rep.transpose(0,1)
18 ).softmax(dim=-1)
19
20 ### same as before
21 Y = torch.matmul(alpha,value_rep)
```

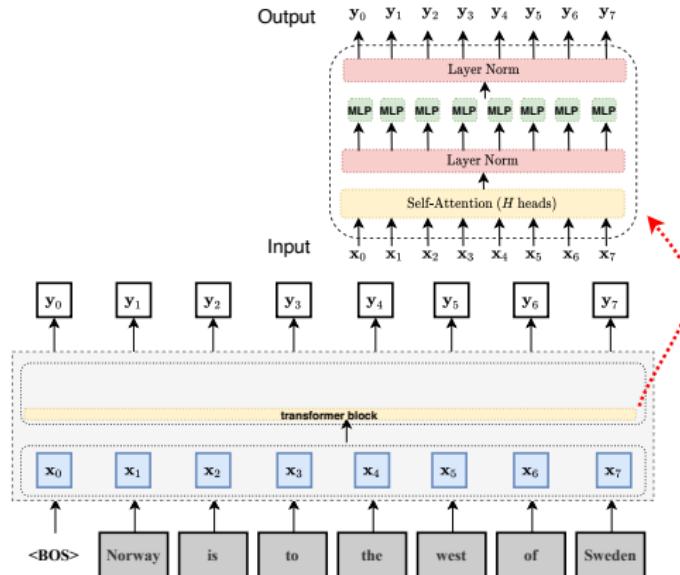
Multi-headed Attention

- ▶ Allows the model to simultaneously focus on multiple parts of input.



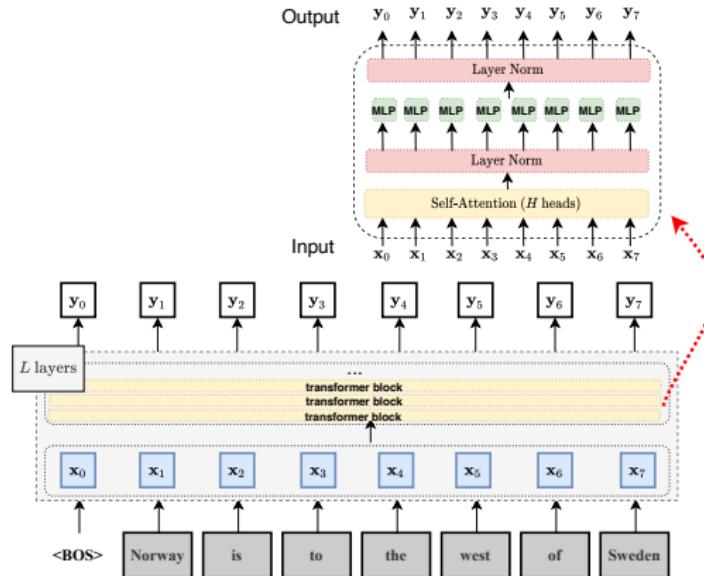
Transformer Blocks and Multiple Layers

- ▶ Current models are a bit more complex and multi-layered.



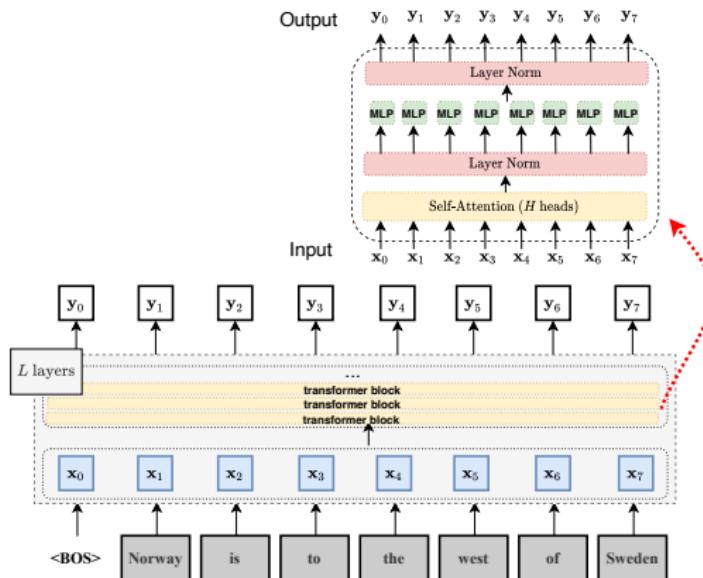
Transformer Blocks and Multiple Layers

- ▶ Current models are a bit more complex and multi-layered.



Transformer Blocks and Multiple Layers

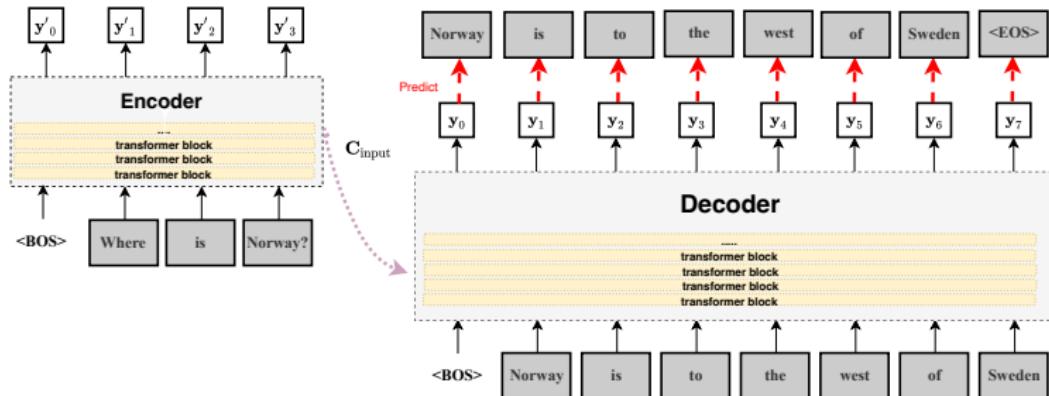
- ▶ Current models are a bit more complex and multi-layered.



BERT (Devlin et al., 2018): $L = 24$ layers each with $H = 16$ heads, $340M$ parameters, embedding dimension=1024

Transformer Blocks and Multiple Layers

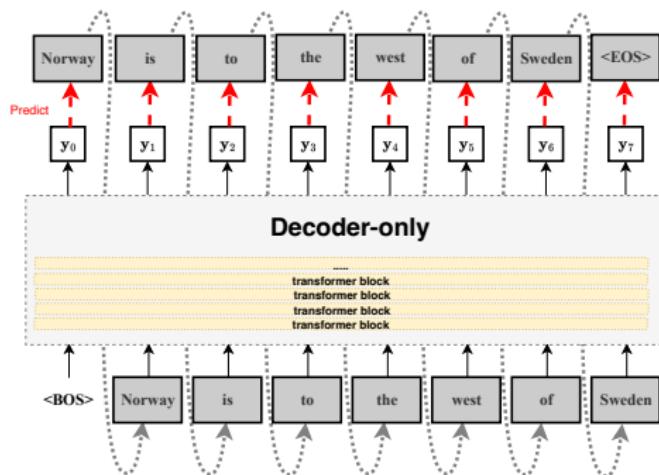
- ▶ Current models are a bit more complex and multi-layered.



T5 model ([Raffel et al., 2020](#)), text2text architecture: **encoder** and **decoder**, 24 layers, 128 heads, cross-attention (11B parameters).

Transformer Blocks and Multiple Layers

- ▶ Current models are a bit more complex and multi-layered.



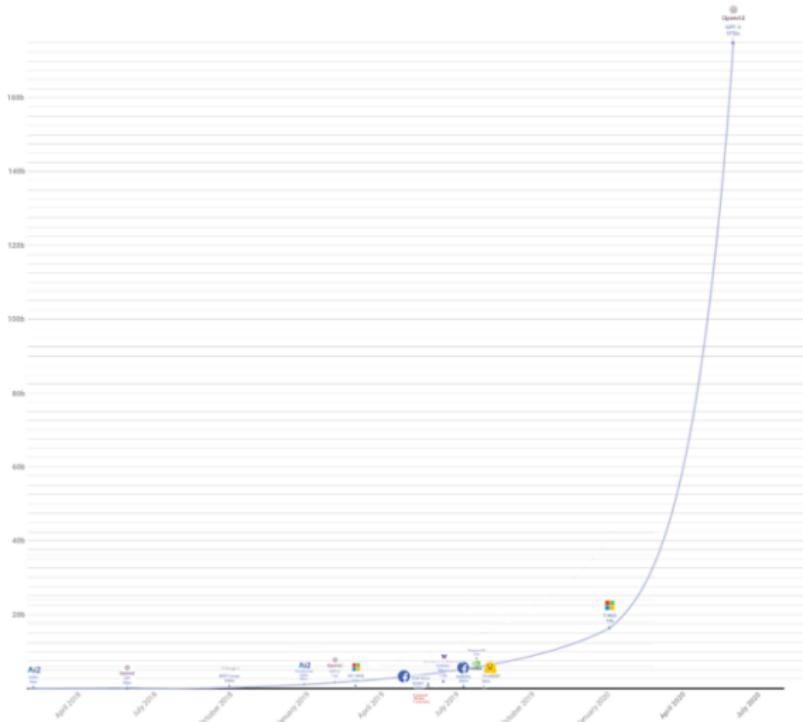
GPT3 (Brown et al., 2020): $L = 96$ layers each with $H = 96$ heads ($175B$ parameters), embedding dimension=12288

The Race for Larger Models (Sanh et al., 2019)

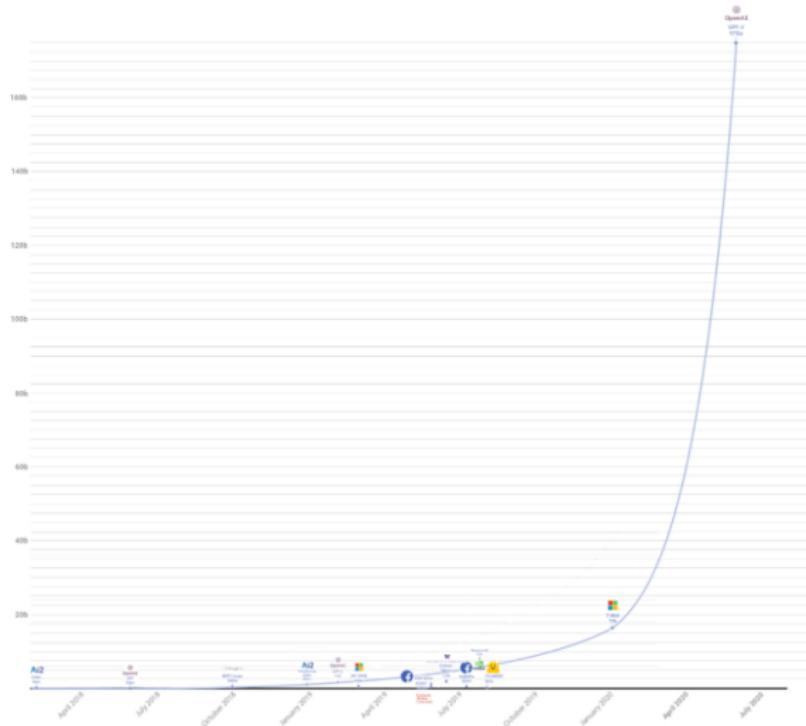


Not (yet) a quadrillion parameters, but models are quickly becoming much larger.

A more updated picture



A more updated picture



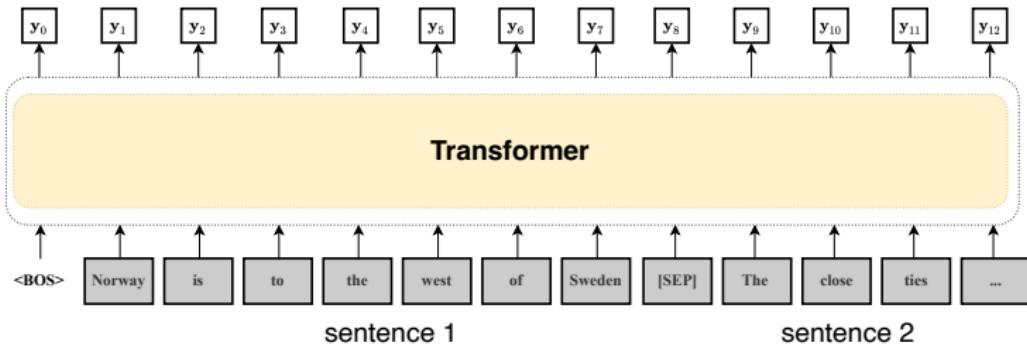
- ▶ How big are models like ChatGPT, GPT4? *We don't know.*

Attention and Transformers: Interim Summary

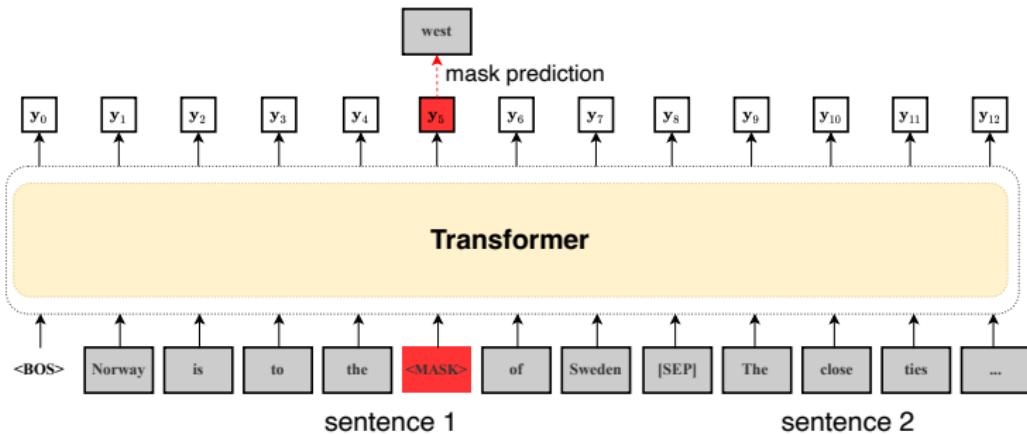
- ▶ **Attention:** mechanism for building contextual representations via brute-force looking around, *variants*: self, causal, cross.
- Transformer:** architecture based on attention (*plus other components, transformer block*). **Existing models:** multi-layered, many parameters.

Training and Using Models

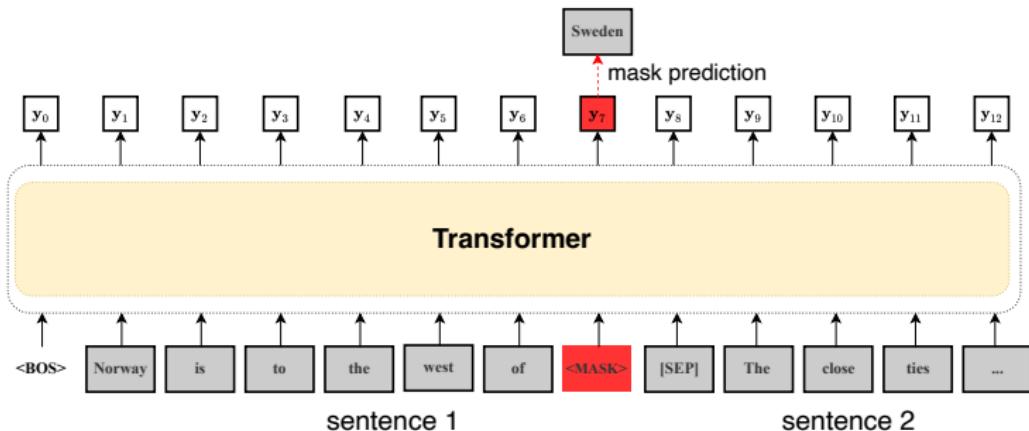
Pre-training Encoders (Devlin et al. (2018))



Pre-training Encoders (Devlin et al. (2018))

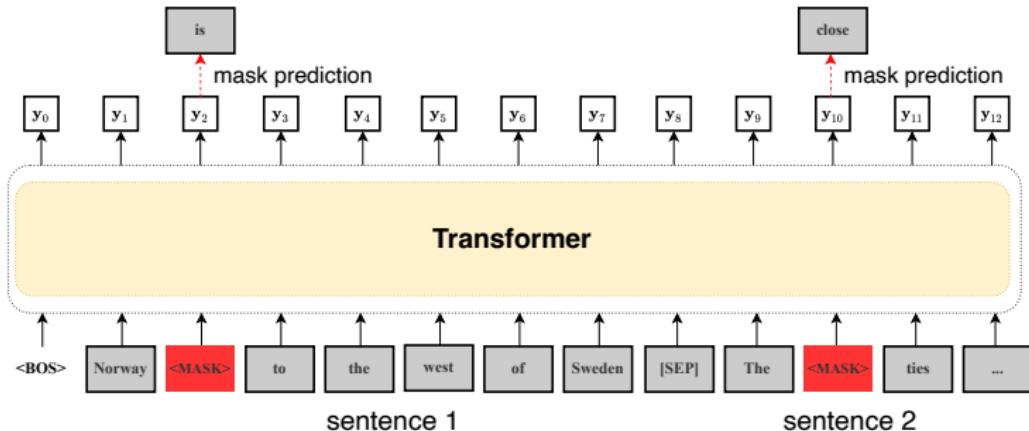


Pre-training Encoders (Devlin et al. (2018))



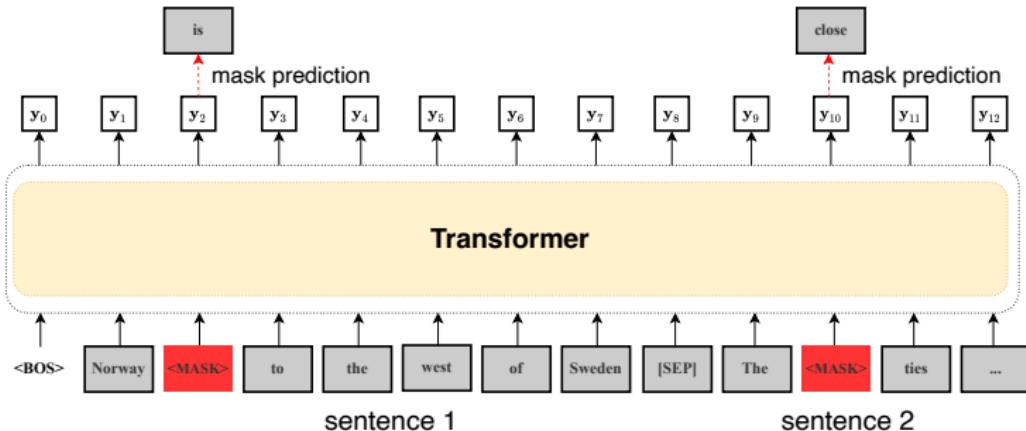
Pre-training Encoders

(Devlin et al. (2018))



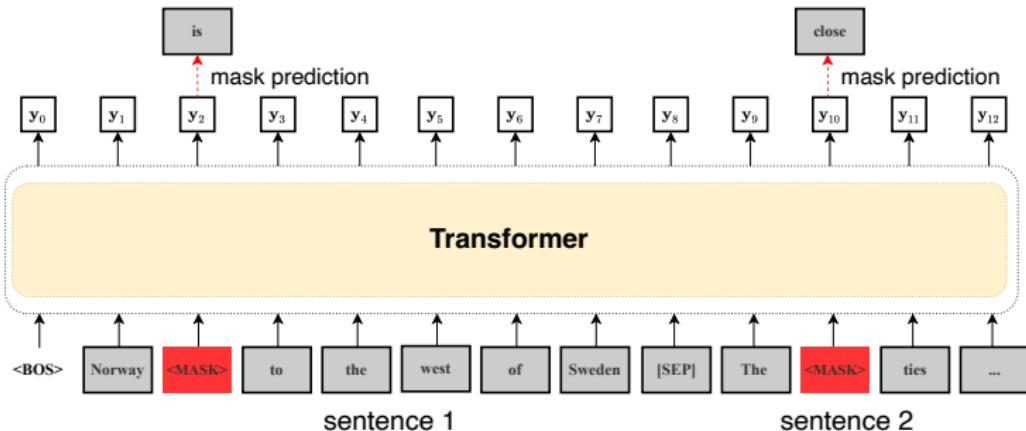
Pre-training Encoders

(Devlin et al. (2018))



Does this make sense to do? Has long been used as a teaching and assessment technique for humans (**cloze test**) (Taylor (1953)).

Pre-training Encoders (Devlin et al. (2018))



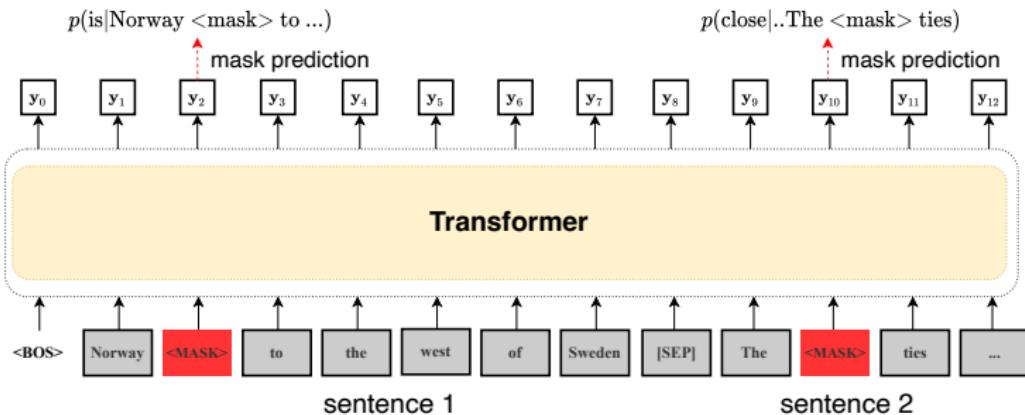
Self-supervised learning: no need for data annotation, can be scaled to large amounts of unlabeled data!

Mask prediction in action

- ▶ An example using **RoBERTa** (Liu et al., 2019) and Huggingface.

```
1 ## install: pip install transformers
2 from transformers import pipeline
3
4 mask_predictor = pipeline(
5     "fill-mask", model="roberta-base"
6 )
7
8 mask_predictor("This lecture is really <mask> .")
9 #[{'score': 0.392790824174881,
10 #   'token_str': ' good',}
11 # {'score': 0.09650349617004395,
12 #   'token_str': ' interesting',}
13 # {'score': 0.0915081575512886,
14 #   'token_str': ' great',}
15 # {'score': 0.03177962079644203,
16 #   'token_str': ' important',}
17 # {'score': 0.031510066241025925,
18 #   'token_str': ' long',} ...]
```

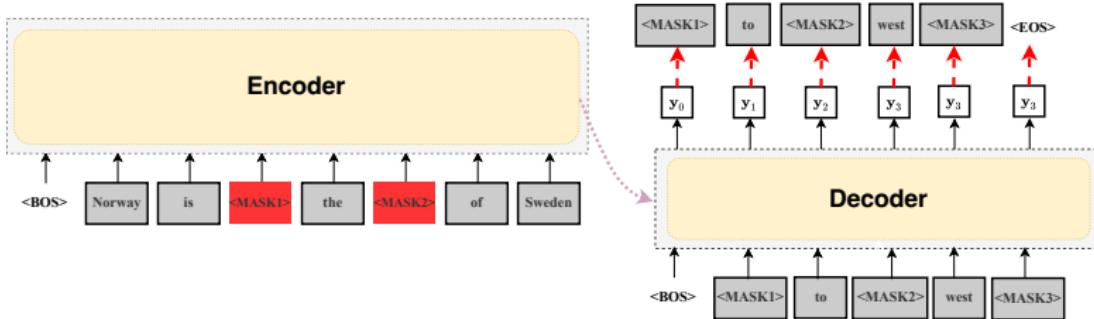
How this works technically



Maximum likelihood training (**mask language objective**), maximize the *likelihood* of correct token prediction.

Word Prediction Pre-training: Text2Text Models

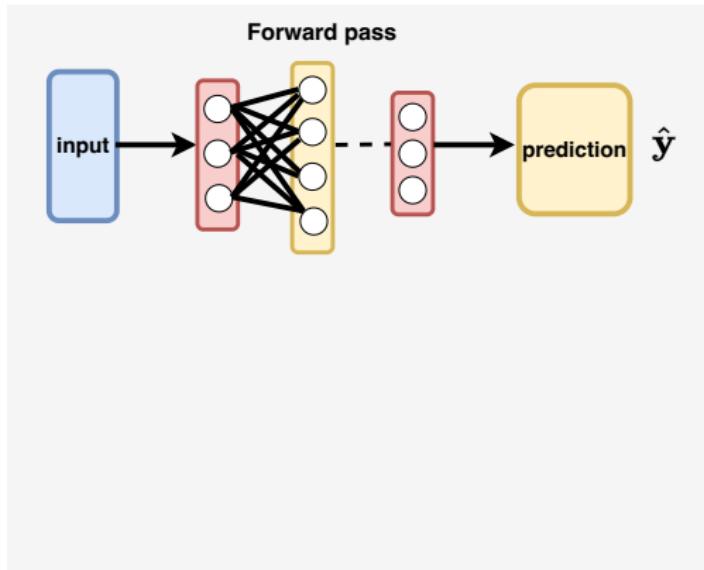
Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <ID> <ID> me to your party apple week .	(original text)
Deshuffling	party me for your to . last fun you inviting week Thank	(original text)
MASS-style Song et al. (2019)	Thank you <ID> <ID> me to your party <ID> week .	(original text)
L.i.d. noise, replace spans	Thank you <ID> me to your party <Y> week .	<ID> for inviting <Y> last <Z>
L.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <ID> to <Y> week .	<ID> for inviting me <Y> your party last <Z>



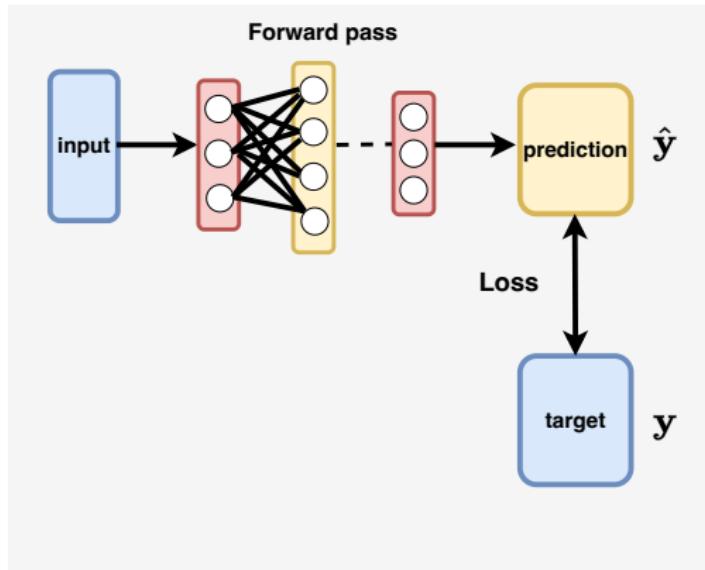
(Raffel et al., 2020)

- ▶ From perturbed input, reconstruct target sentence via generation; related to *generative* pre-training for decoders-only models (Radford et al., 2018).

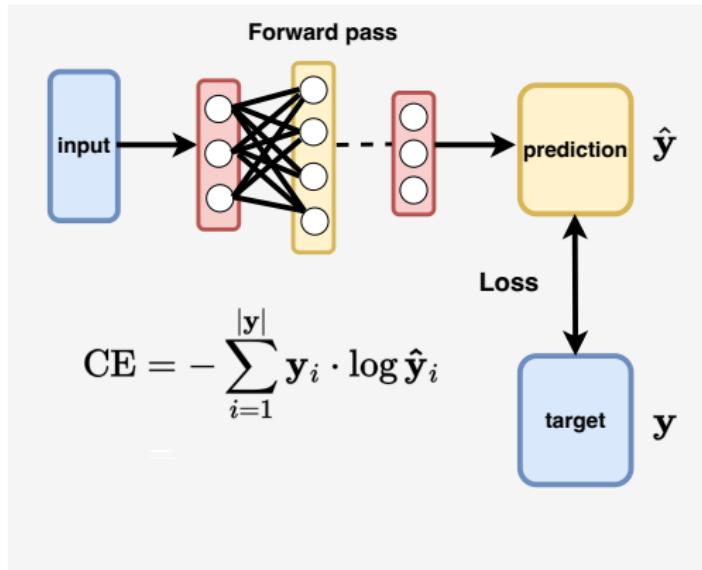
Model training in a nutshell



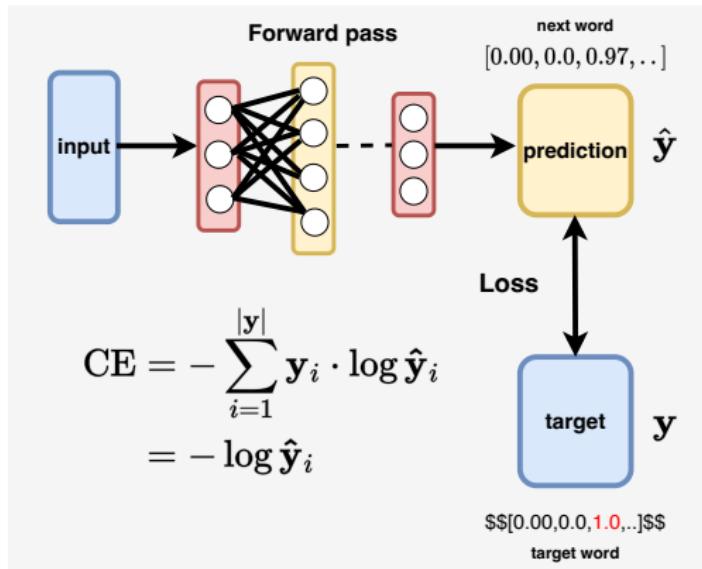
Model training in a nutshell



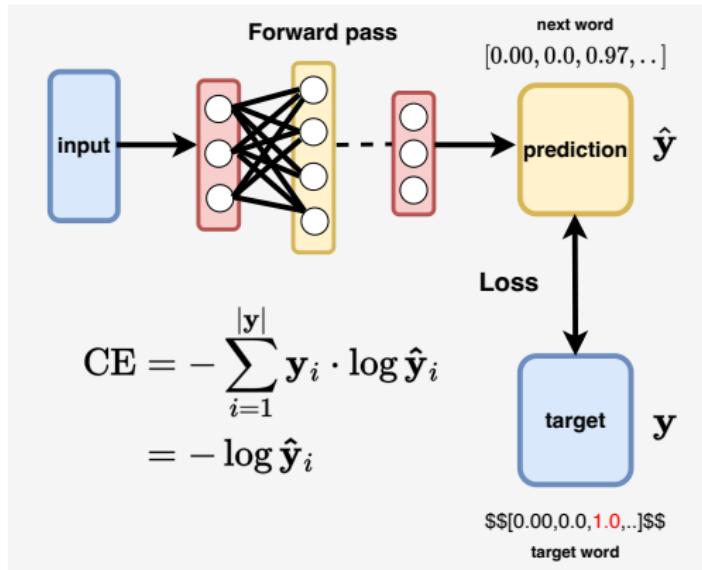
Model training in a nutshell



Model training in a nutshell



Model training in a nutshell



- ▶ Involved gradient-based optimization (variants of SGD), efficient computation of gradients via **backpropagation**.

Pytorch Training Example

```
1 import torch
2 from transformers import GPT2Tokenizer, \
3     GPT2LMHeadModel
4
5     ### GPT model and tokenizer
6 tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
7 model = GPT2LMHeadModel.from_pretrained("gpt2",
8         return_dict=True)
9 model.train()
10     ### optimizer
11 optimizer = torch.optim.SGD(model.parameters())
12     ### data
13 data = tokenizer(["this is a boring class"],
14                 return_tensors="pt"
15 )
16     ### run model
17 output = model(**data, labels=data.input_ids)
18 loss = output.loss ## CE loss
19 loss.backward()      ## computes gradients/backprop
20 optimizer.step()    ## update
```

see more details about training [here](#)

Large-scale pre-training

- ▶ **Pre-training:** training models (e.g., using masking objective, generative training) on a large corpus of text, learn general-purpose representations.

Large-scale pre-training

Web-scale resources for training e.g., C4 ([Raffel et al. \(2020\)](#); [Dodge et al. \(2021\)](#)),
English (raw): 1.4Tr tokens, 1.1B documents (2.3 TB raw text).



<http://satsat.info/english-forums/5660-uk-section.html>

European Championship Match (Uefa Cup) Kayserispor vs Paris S.G. Do you know if there is any no-mpeg4 channel on express am-1 that i can receive in belgium except uzbekistan tv? pichi po russki essli kochitii, ya ponimaiou nu nye pishu horoshenka kak vi vidite. Read the name of the subject, please!

<http://www.macs.hw.ac.uk/esslli05/give-page.php%3F11.html>

Short Description: The 10th conference on Formal Grammar and The 9th Meeting on Mathematics of Language will be held in Edinburgh from August 5-th to August 7-th 2005. FG-MOL provides a forum for the presentation of new and original research on formal grammar, mathematical linguistics and the application of formal and mathematical methods to the study of natural language. ESSLI 2005 will be hosted in the Riccarton campus in beautiful Edinburgh during the impressive festive month, 5 major festivals are taking place in August.

<http://www.macs.hw.ac.uk/esslli05/>

ESSLLI is the annual summer school of FoLLI, the Association for Logic, Language and Information. ESSLI 2005 will be hosted in the Riccarton campus in beautiful Edinburgh during the impressive festive month. 5 major festivals take place in August. ESSLI05 photos are now online. Please send us your pictures to be added to ESSLI05 webpage. 03.10.05 - Photos of ESSLI05 online. Send us your pictures



<http://publin.ie/2018/its-like-a-confession-box-watch-a-few-older-r...>

This is one of the more simple things we'll post on Publin, but it's the kind of entertainment that any lover of the Dublin pub will enjoy. Here's a short conversation between a few regulars and publican Liam Aherne in The Palace Bar on Fleet Street. The discussions are outtakes from the excellent 2013 documentary 'The Irish Pub'. They discuss the importance of the pub to rural and urban areas while having a jar and a laugh at the same time. Tommy Wright - THE PALACE BAR - Extract from Fegan Films / Atom Films on Vimeo.

<https://o.canada.com/travel/international-travel/best-pubs-in-dublin..>

"A good puzzle would be to cross Dublin without passing a pub," wrote the famous Irish novelist James Joyce in Ulysses. The mirrors and wooden niches of the Palace Bar – considered by many to be the perfect example of an old Dublin pub. Enjoying a Guinness stout in the Temple Bar pub, Colm Quilligan with a group of Literary Pub Crawlers inside O'Neill's bar, The Guinness Storehouse and Gravity Bar. No visit to the Irish capital would be complete without a pilgrimage to the Guinness Storehouse at St. James's Gate, where you'll learn plenty about this world famous stout including the brewing process and the Arthur Guinness story. After you have seen how it's made, it's time to taste the famous product. Hovering above the roof of the Storehouse is the Gravity Bar, and with a pint of Guinness in hand and incredible 360-degree views over the streets of Dublin, it's the perfect

Large-scale pre-training

What's in this corpus: we don't really know, a little of everything (see Dodge et al. (2021)), blackbox (*similar to our models*).

AI2 Allen Institute for AI

C4 Search <https://c4-search.apps.allenai.org>

ESSLI

Search

Found 14 results in 0.18 seconds

<http://satsat.info/english-forums/5660-uk-section.html>

European Championship Match (Uefa Cup) Kayserispor vs Paris S.G. Do you know if there is any no-mpeg4 channel on express am-1 that i can receive in belgium except uzbekistan tv? pichi po russki essli kochitii, ya ponimaiou nu nye pishu horoshenka kak vi vidite. Read the name of the subject, please!

<http://www.macs.hw.ac.uk/esslli05/give-page.php%3F11.html>

Short Description: The 10th conference on Formal Grammar and The 9th Meeting on Mathematics of Language will be held in Edinburgh from August 5-th to August 7-th 2005. FG-MOL provides a forum for the presentation of new and original research on formal grammar, mathematical linguistics and the application of formal and mathematical methods to the study of natural language. ESSLI 2005 will be hosted in the Riccarton campus in beautiful Edinburgh during the impressive festive month. 5 major festivals are taking place in August.

<http://www.macs.hw.ac.uk/esslli05/>

ESSLLI is the annual summer school of FoLLI, the Association for Logic, Language and Information. ESSLI 2005 will be hosted in the Riccarton campus in beautiful Edinburgh during the impressive festive month. 5 major festivals take place in August. ESSLI05 photos are now online. Please send us your pictures to be added to ESSLI05 webpage. 03.10.05 - Photos of ESSLI05 online. Send us your pictures

AI2 Allen Institute for AI

C4 Search

21 Palace pub Dublin

Search

Found more than 10,000 results in 0.12 seconds

<http://publin.ie/2018/its-like-a-confession-box-watch-a-few-older-r...>

This is one of the more simple things we'll post on Publin, but it's the kind of entertainment that any lover of the Dublin pub will enjoy. Here's a short conversation between a few regulars and publican Liam Aherne in The Palace Bar on Fleet Street. The discussions are outtakes from the excellent 2013 documentary 'The Irish Pub'. They discuss the importance of the pub to rural and urban areas while having a jar and a laugh at the same time. Tommy Wright - THE PALACE BAR - Extract from Fegan Films / Atom Films on Vimeo.

<https://o.canada.com/travel/international-travel/best-pubs-in-dublin...>

"A good puzzle would be to cross Dublin without passing a pub," wrote the famous Irish novelist James Joyce in Ulysses. The mirrors and wooden niches of the Palace Bar â€“ considered by many to be the perfect example of an old Dublin pub.

Enjoying a Guinness stout in the Temple Bar pub. Colm Quilligan with a group of Literary Pub Crawlers inside O'Neill's bar. The Guinness Storehouse and Gravity Bar. No visit to the Irish capital would be complete without a pilgrimage to the Guinness Storehouse at St. James's Gate, where you'll learn plenty about this world famous stout including the brewing process and the Arthur Guinness story. After you have seen how it's made, it's time to taste the famous product. Hovering above the roof of the Storehouse is the Gravity Bar, and with a pint of Guinness in hand and incredible 360-degree views over the streets of Dublin, it's the perfect

Large-scale pre-training

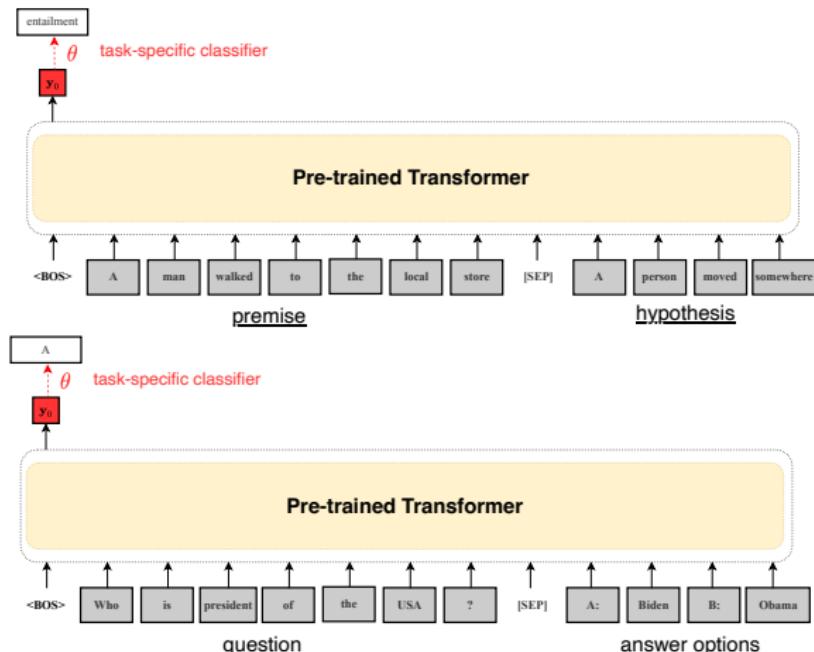
RedPajama-Data: An Open Source Recipe to Reproduce LLaMA training dataset



This repo contains a reproducible data recipe for the RedPajama data, with the following token counts:

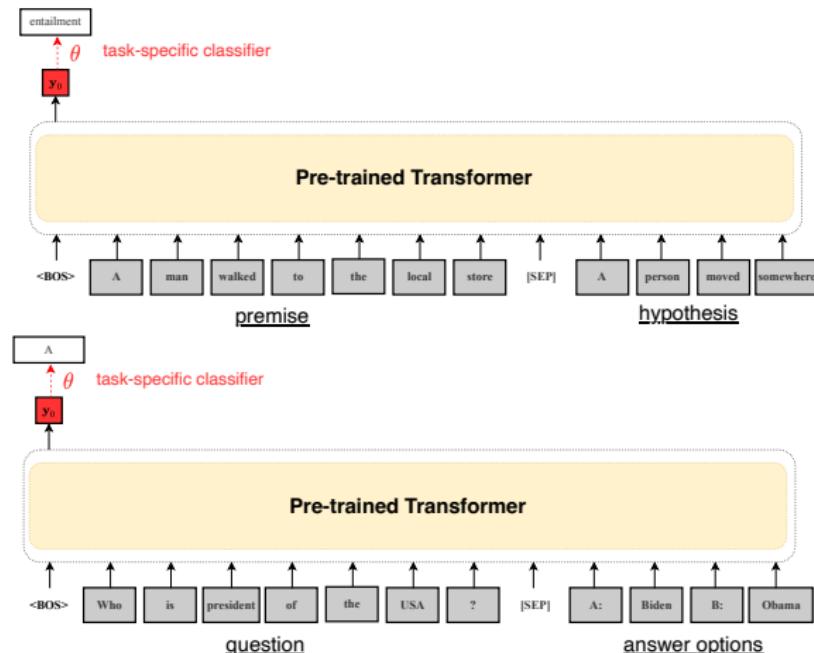
Dataset	Token Count
Commoncrawl	878 Billion
C4	175 Billion
GitHub	59 Billion
Books	26 Billion
ArXiv	28 Billion
Wikipedia	24 Billion
StackExchange	20 Billion
Total	1.2 Trillion

Model Fine-tuning



Fine-tuning: customizing models to target tasks using additional parameters; **idea:** bootstrap off of pre-training knowledge.

Typical experiment



Supervised Learning: fine-tune using labeled training set, measure and report prediction accuracy on unseen set.

Recent Developments: Prompting and In-context Learning

USER	Write me a poem about the ESSLLI summer school
ASSISTANT	<p>In the heart of summer's golden glow, Where curious minds wander and grow, There lies a place of learning delight, The haven that is ESSLLI, shining bright.</p> <p>From distant lands they gather, young and old, Seeking knowledge's treasure, untold. Linguists, logicians, philosophers too, Together they explore ideas anew.</p> <p>Like a tapestry woven with care, Each strand unique, yet all in flair. Days filled with lectures and workshops vast, Weaving knowledge, creating a future that lasts.</p>

Large language models (LLMs) are used without specialized training, frozen, after tuning on *instructions* (Ouyang et al., 2022).

Recent Developments: Prompting and In-context Learning

```
1 import openai
2
3 IN_CONTEXT=f"""
4 # This task requires you to negate sentences
5
6 sentence: I am swimming but sailing.
7 negation: I am not sailing but swimming instead.
8
9 sentence: She went over to the house but not inside
10 negation: She went to the house and entered the house.
11
12 sentence: {sentence}
13 """
14 new_problem = "Dan went to pick apples"
15 prompt = IN_CONTEXT.format(sentence=new_problem)
16
17 ##### call to GPT3
18 response = openai.Completion.create(
19     model="text-davinci-003",
20     prompt=prompt,
21 )
```

Conclusion

- ▶ **Language models:** assign probabilities to sequences, generation.
- ▶ **Recent Advances:** novel neural architectures (*transformers, attention*), large-scale pre-training of neural representations, enormous datasets.

State-of-the-art results on many NLP problems, many exciting areas for future research and new applications.

Credits and Additional Resources

- ▶ **Transformers:** Many examples taken from *Peter Bloem's* excellent survey **Transformers from Scratch** See also: **Annotated Transformer** (*Sasha Rush*), **Illustrated Transformer** (*Jay Alammar*)
- Classic NLP resources:** (*Jurafsky, 2000*) ([website](#)), (*Manning and Schutze, 1999*), **NLP notes** (*Michael Collins*)
- Topics glossed over:** Decoding and generation, see **this survey**, General machine learning: (*Daumé, 2017*) ([website](#), Ch. 7,10), (*Murphy, 2012*).
- Pytorch tutorials, math background.**

References I

- Arthur, P., Neubig, G., and Nakamura, S. (2016). Incorporating discrete translation lexicons into neural machine translation. *arXiv preprint arXiv:1606.02006*.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Chomsky, N. (1957). Syntactic structures. In *Syntactic Structures*. De Gruyter Mouton.
- Daumé, H. (2017). *A course in machine learning*. Hal Daumé III.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Dodge, J., Sap, M., Marasovic, A., Agnew, W., Ilharco, G., Groeneveld, D., and Gardner, M. (2021). Documenting the english colossal clean crawled corpus.
- Jurafsky, D. (2000). *Speech & language processing*. Pearson Education.
- Lin, Y., Michel, J.-B., Lieberman, E. A., Orwant, J., Brockman, W., and Petrov, S. (2012). Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

References II

- Manning, C. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.
- Norvig, P. (2012). Colorless green ideas learn furiously: Chomsky and the two cultures of statistical learning. *Significance*, 9(4):30–33.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.
- Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

References III

- Sproat, R. and Jaitly, N. (2016). Rnn approaches to text normalization: A challenge. *arXiv preprint arXiv:1611.00068*.
- Taylor, W. L. (1953). “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Widdows, D. (2004). *Geometry and meaning*. CSLI.