NLP

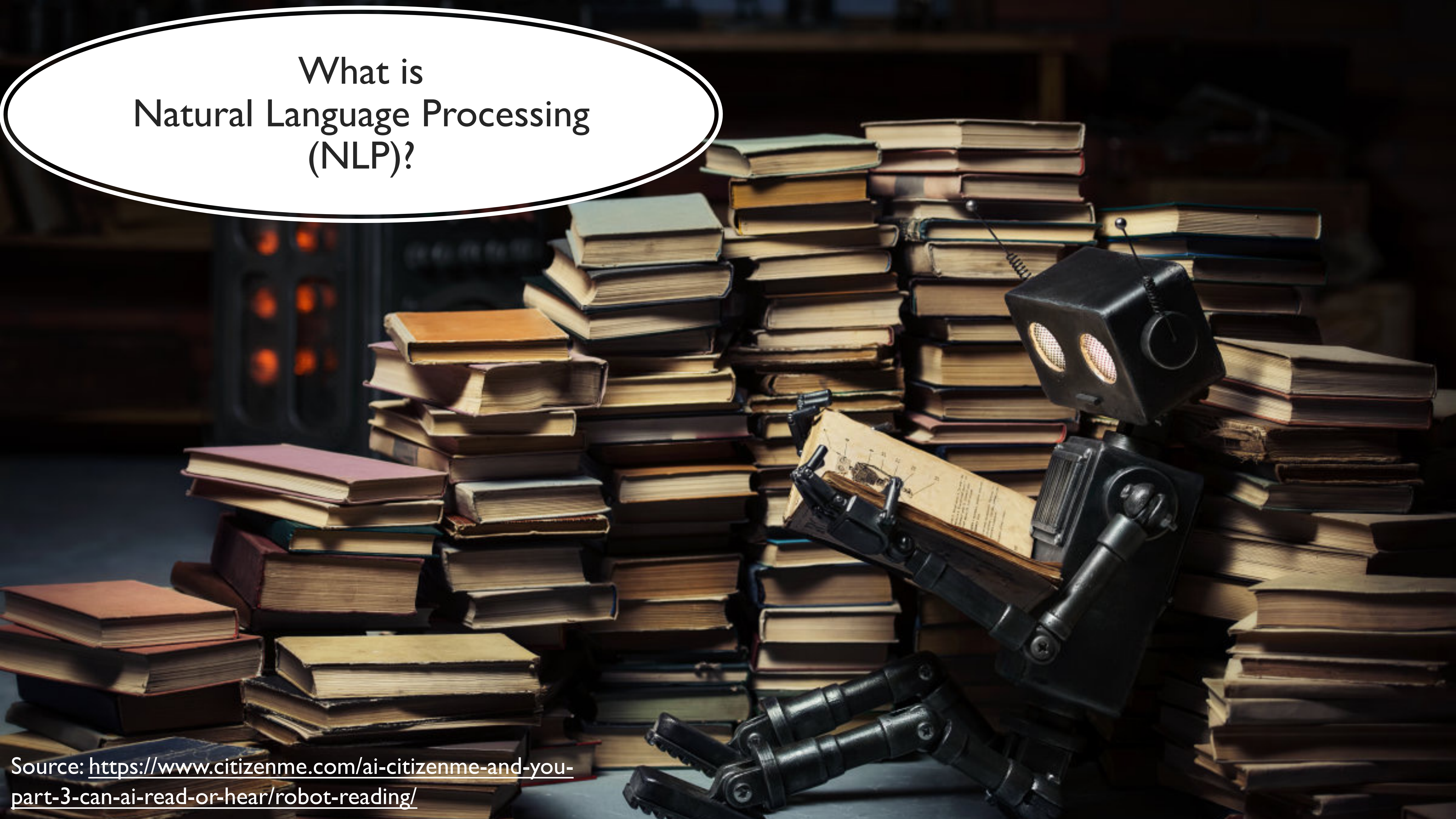
Deep Learning Applications in Natural Language Processing

Huan Sun, Zhen Wang
Computer Science and Engineering
TDAI Foundations of Data Science & Artificial Intelligence
Deep Learning Summer School

The Ohio State University
Translational Data Analytics Institute

Acknowledgement

- Stanford CS224n (Winter 2022) by Chris Manning, Anna Goldie
- UT Austin NLP courses by Greg Durrett
- Ohio State NLP CSE5525
- Textbook: Jurafsky and Martin, Speech and Language Processing
- References on the slides


What is
Natural Language Processing
(NLP)?
Source: https://www.citizenme.com/ai-citizenme-and-you-part-3-can-ai-read-or-hear/robot-reading/

A bit history…

- 1950 – 1969
- 1969 – 1992
- 1993 – 2012
- 2013 – present

Christopher Manning. "Human Language Understanding & Reasoning" in Daedalus, Spring 2022

A bit history…

- 1950 – 1969
  - Machine translation (word-level lookups, rule-based mechanisms)

- 1969 – 1992
  - Rule-based NLP demonstration systems
  - Start to model the complexity of human language understanding

- 1993 – 2012
  - Constructing annotated linguistic resources
  - Supervised machine learning

- 2013 – present
  - Deep learning

Christopher Manning. "Human Language Understanding & Reasoning" in Daedalus, Spring 2022

A bit history…

- 1950 – 1969
  - Machine translation (word-level lookups, rule-based mechanisms)

- 1969 – 1992
  - Rule-based NLP demonstration systems
  - Start to model the complexity of human language understanding

- 1993 – 2012
  - Constructing annotated linguistic sources
  - Supervised machine learning

- 2013 – present
  - Deep learning

2013 – 2017

2018 – present

2018:
Pre-trained
self-supervised
models

Christopher Manning. "Human Language Understanding & Reasoning" in Daedalus, Spring 2022

"In hindsight, the development of large-scale self-supervised learning approaches may well be viewed as the fundamental change, and the third era might be extended until 2017. The impact of pretrained self-supervised approaches has been revolutionary: it is now possible to train models on huge amounts of unlabeled human language material in such a way as to produce one large pretrained model that can be very easily adapted, via fine-tuning or prompting, to give strong results on all sorts of natural language understanding and generation tasks. As a result, progress and interest in NLP have exploded. There is a sense of optimism that we are starting to see the emergence of knowledge-imbued systems that have a degree of general intelligence."

--Christopher Manning. "Human Language Understanding & Reasoning,"
Daedalus, Spring 2022

Why do we care in TDAI?

- Text data is everywhere
  - Scientific articles
  - Clinical texts
  - Social media posts
  - Financial news

- NLP: A key component in interdisciplinary collaboration

Tutorial Structure

Part I (~75 mins):

- Tasks

- Deep Learning Models

Break (~15mins)

Part II: (~45 mins):

- Large Language Models

- Demo

QA (~15 mins)

Tutorial Structure

Part I (~75 mins):

- Tasks

- Deep Learning Models

Break (~15mins)

Part II: (~45 mins):

- Large Language Models

- Demo

QA (~15 mins)

Popular Tasks

- Classification (language understanding)
  - Sentiment analysis
- Sequence labeling (language understanding)
  - Part of Speech (POS) tagging
  - Named entity recognition (NER)
- Sequence-to-sequence problem (language generation)
  - Language modeling
  - Machine translation
  - Text summarization
  - Dialogue response generation

Bioinformatics

Political Science

Cheminformatics

Business Intelligence

…

Sentiment Analysis

Source: https://twitter.com/friends_quotes1/status/649997787199873024

Sentiment Analysis

Given a piece of text:	Predict label:
this movie was great! would watch again	+
the movie was gross and overwrought, but I liked it	+
this movie was not really very enjoyable	—

Classification: binary or multiclass

Named Entity Recognition (NER)

Ousted WeWork founder Adam Neumann lists his Manhattan penthouse for $37.5 million

WeWork	Adam Neumann	Manhattan	$37.5 million
[organization]	[person]	[location]	[monetary value]

Example Source: https://monkeylearn.com/blog/named-entity-recognition/

Named Entity Recognition (NER)

Ousted WeWork founder Adam Neumann lists his Manhattan penthouse for $37.5 million

Ousted	WeWork	founder	Adam Neumann	lists his	Manhattan	penthouse for	$37.5 million
	[organization]		[person]		[location]		[monetary value]

Sequence labeling: BIO tagging scheme

O	B-ORG	O	B-PER	I-PER	O	O	B-LOC	O	O	B-MV	I-MV
Ousted	WeWork	founder	Adam	Neumann	lists	his	Manhattan	penthouse	for	$37.5	million

Named Entity Recognition (NER)

Ousted WeWork founder Adam Neumann lists his Manhattan penthouse for $37.5 million

[organization] [person] [location] [monetary value]

Sequence labeling: BIO tagging scheme

Beginning of a PERSON entity

Inside of a PERSON entity

O	B-ORG	O	B-PER	I-PER	O	O	B-LOC	O	O	B-MV	I-MV
Ousted	WeWork	founder	Adam	Neumann	lists	his	Manhattan	penthouse	for	$37.5	million

Popular Tasks

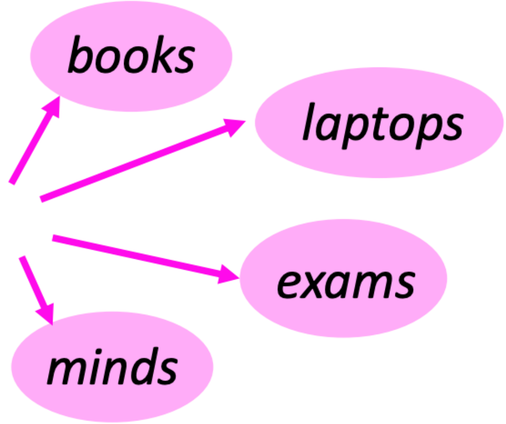
- Classification (language understanding)
  - Sentiment analysis
- Sequence labeling (language understanding)
  - Part of Speech (POS) tagging
  - Named entity recognition (NER)
- Sequence-to-sequence problem (language generation)
  - Language modeling
  - Machine translation
  - Text summarization
  - Dialogue response generation

- Bioinformatics
- Political Science
- Cheminformatics
- Business Intelligence
- …

Language Modeling

- **Language Modeling** is the task of predicting what word comes next

the students opened their ______



- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

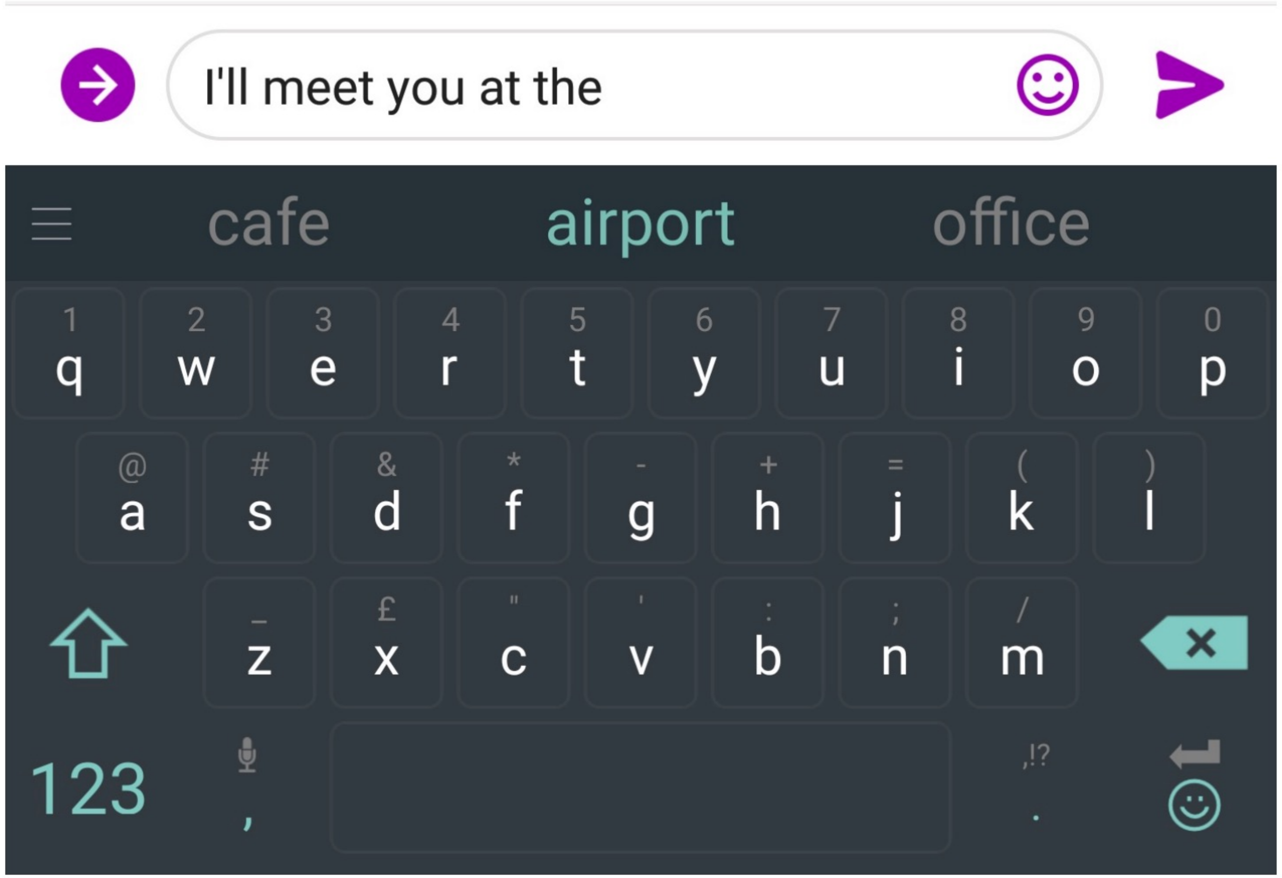
where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$

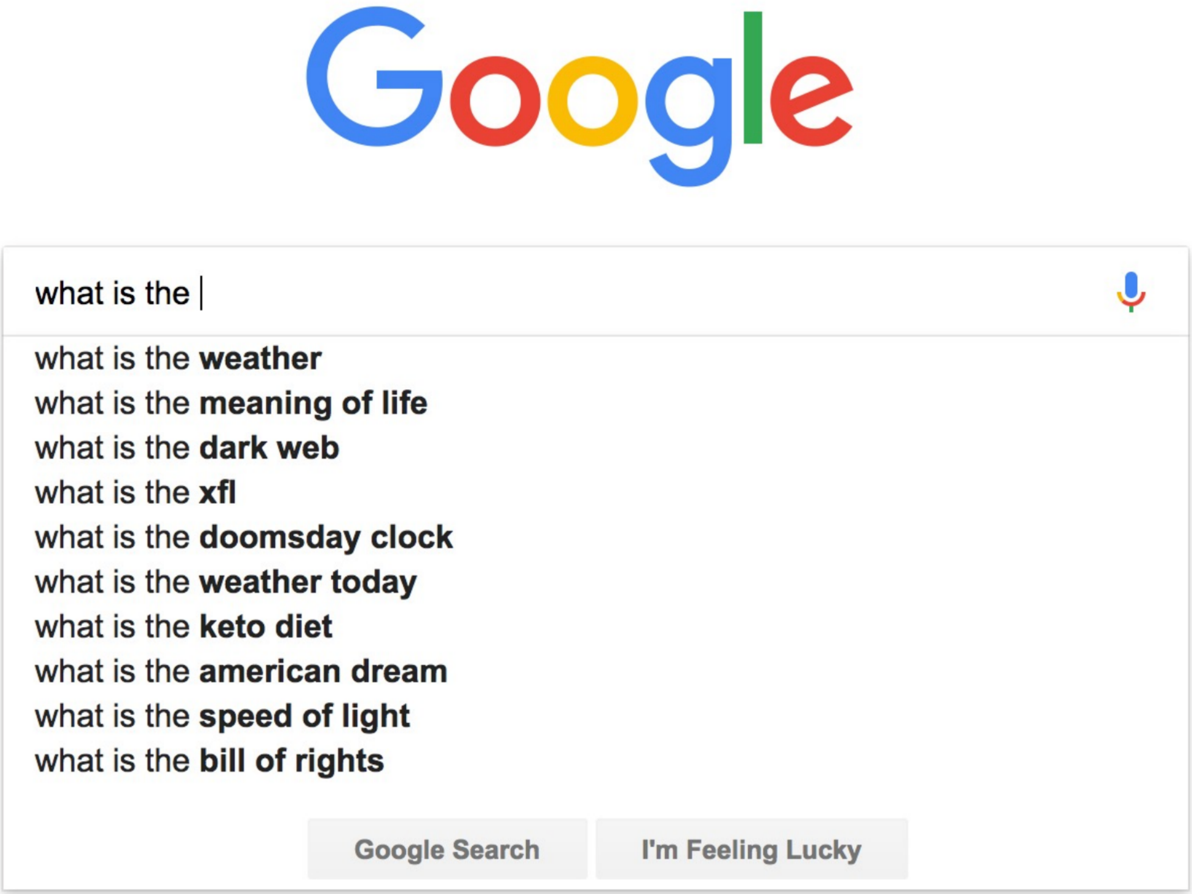
- A system that does this is called a **Language Model**

Credit: Stanford CS224n, Winter 2022

Language Modeling

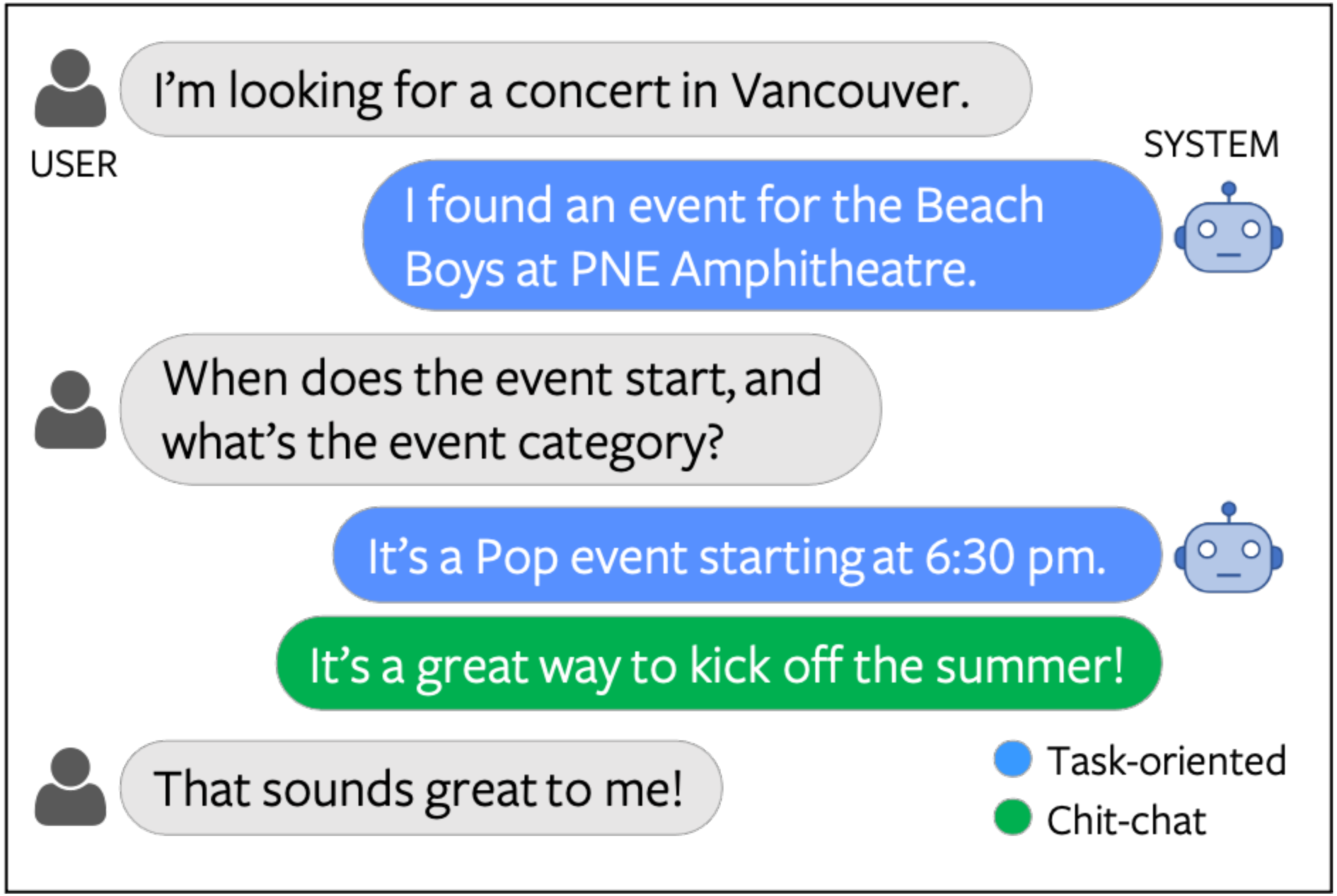
- We use language models every day!





Credit: Stanford CS224n, Winter 2022

Dialogue Response Generation



[Sun et al., NAACL'21]

Tutorial Structure

Part I (~75 mins):

- Tasks
- Deep Learning Models

Break (~15mins)

Part II: (~45 mins):

- Large Language Models
- Demo

QA (~15 mins)

Deep Learning Models for NLP

- How to model a word?
- How to model a sequence of words?
- What is a "pre-trained" model?

How to Model a Word?

- Distributional semantics: A word's meaning is given by the words that frequently appear close-by
  - "You shall know a word by the company it keeps" (J. R. Firth 1957: 11)
- When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window)
- We use the many contexts of w to build up a representation of w

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

Credit: Stanford CS224n, Winter 2022

How to Model a Word?

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts, measuring similarity as the vector dot (scalar) product

$$banking = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix} \qquad monetary = \begin{pmatrix} 0.413 \\ 0.582 \\ -0.007 \\ 0.247 \\ 0.216 \\ -0.718 \\ 0.147 \\ 0.051 \end{pmatrix}$$

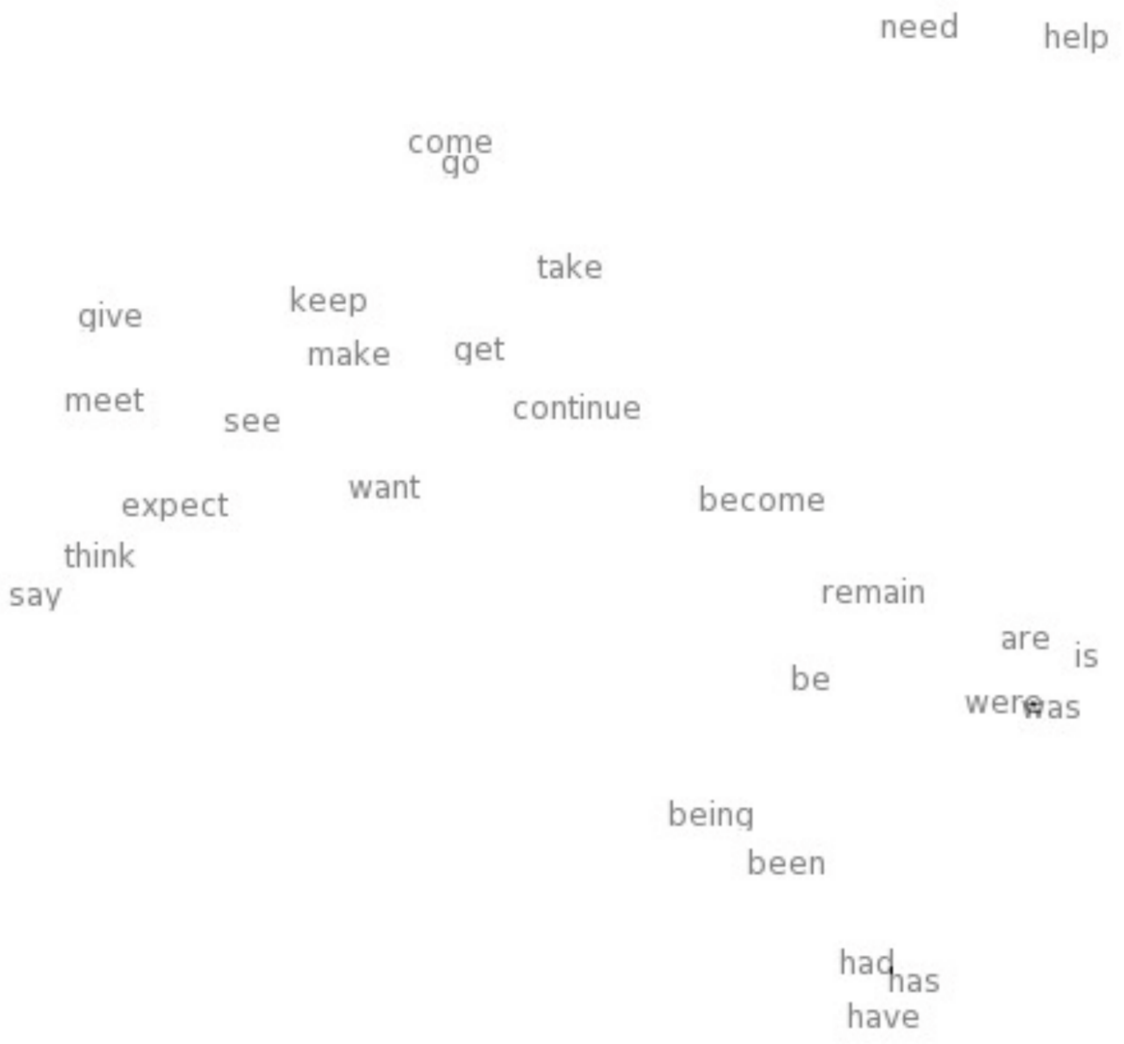
Note: word vectors are also called (word) embeddings or (neural) word representations

They are a distributed representation

Credit: Stanford CS224n, Winter 2022

How to Model a Word?

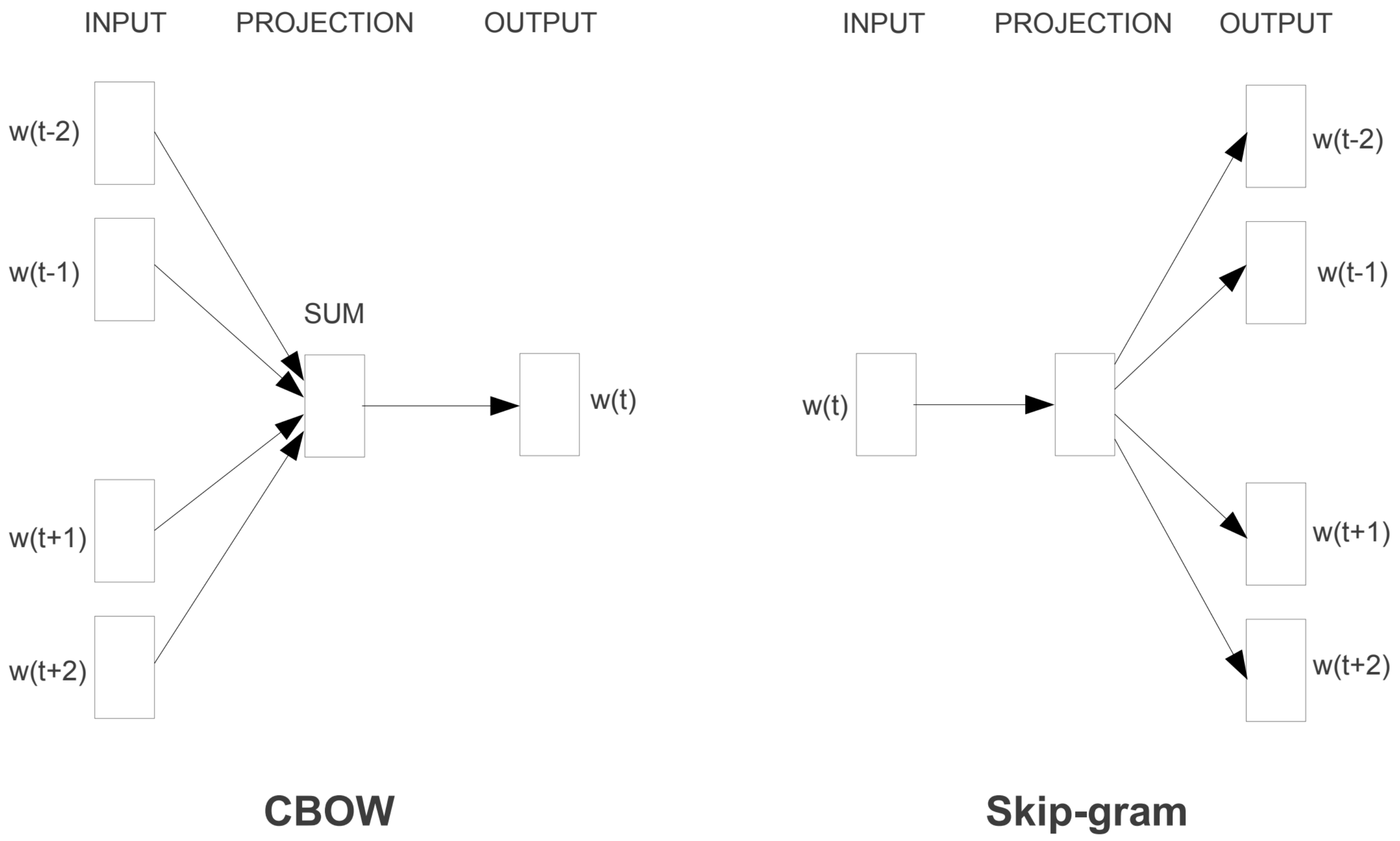
$$\textit{expect} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



Credit: Stanford CS224n, Winter 2022

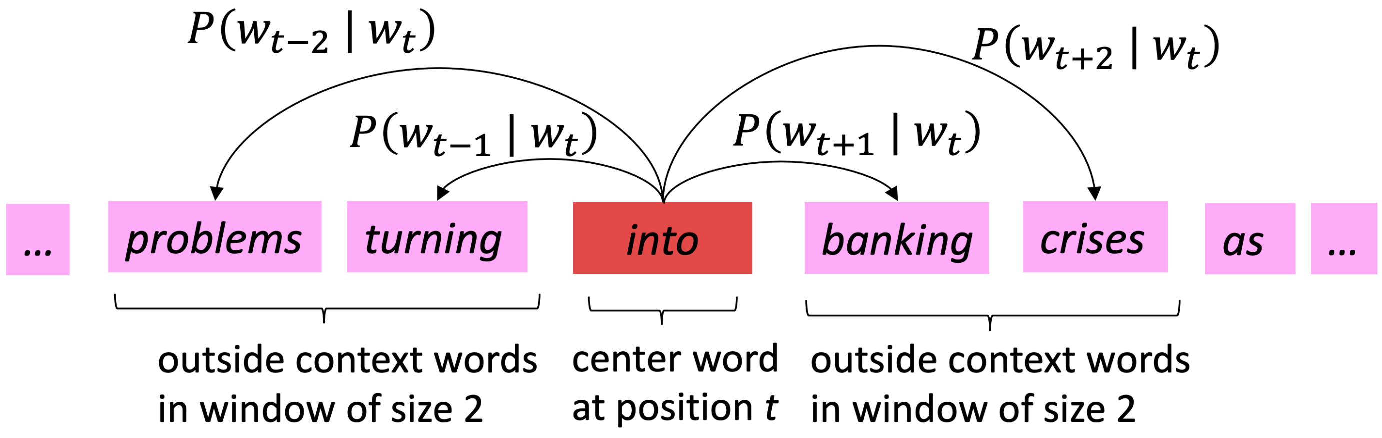
How to Model a Word?

- Word2Vec [Mikolov et al., 2013]



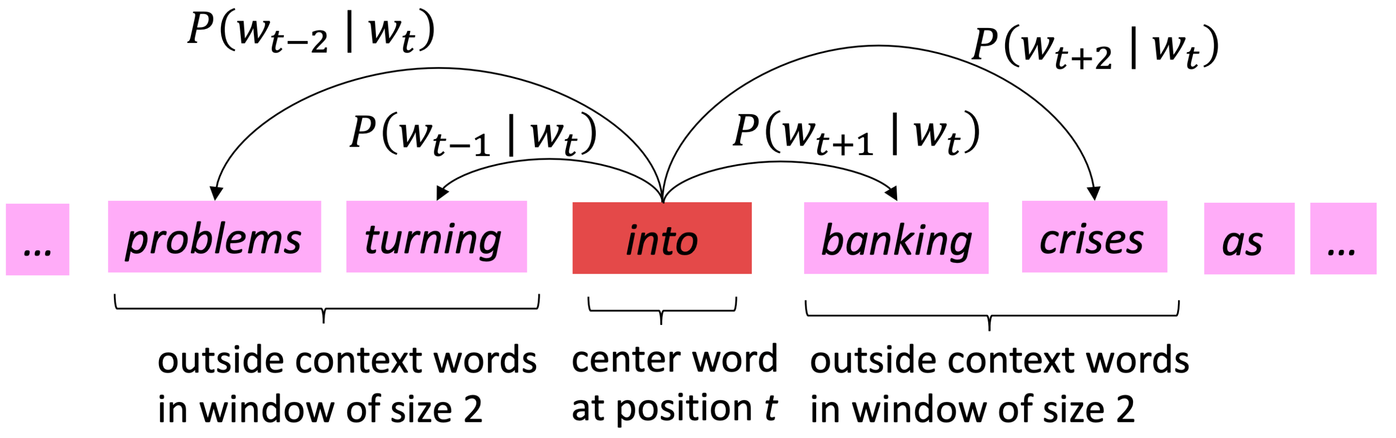
How to Model a Word?

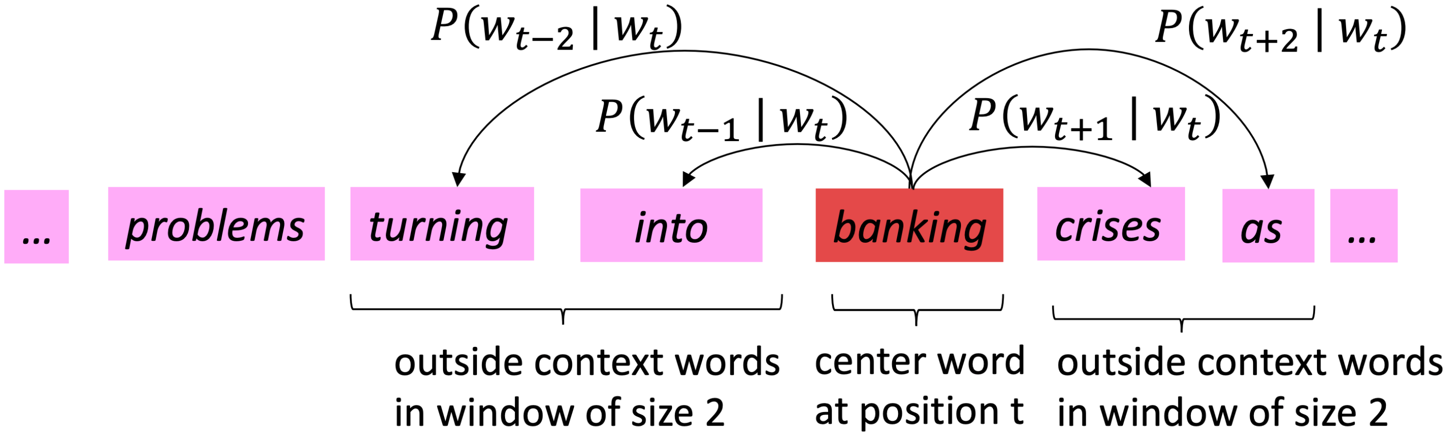
- Skip-gram [Mikolov et al., 2013]



How to Model a Word?

- Skip-gram [Mikolov et al., 2013]

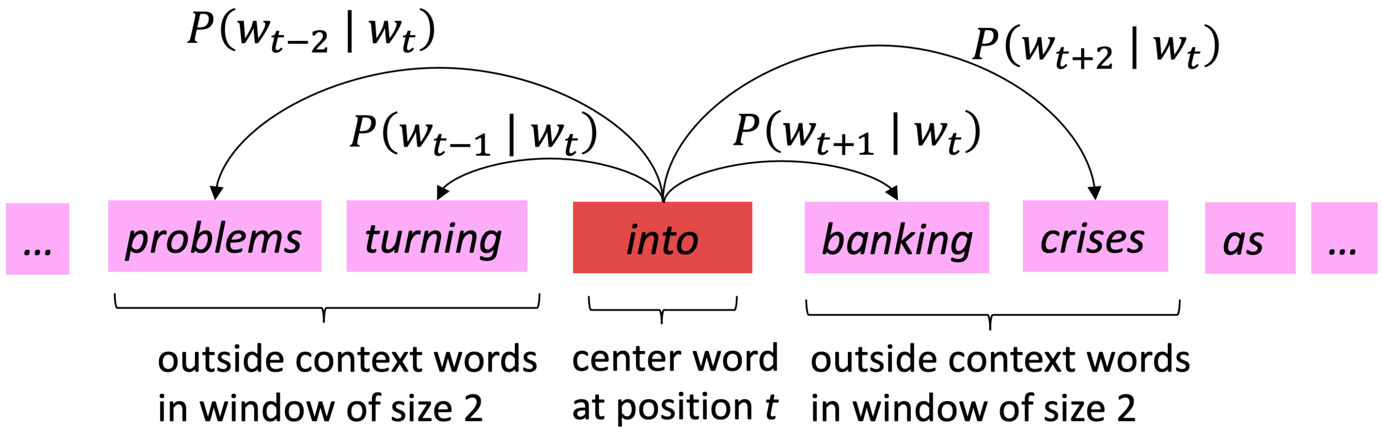




Credit: Stanford CS224n, Winter 2022

How to Model a Word?

- Skip-gram [Mikolov et al., 2013]



For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_t. Data likelihood:

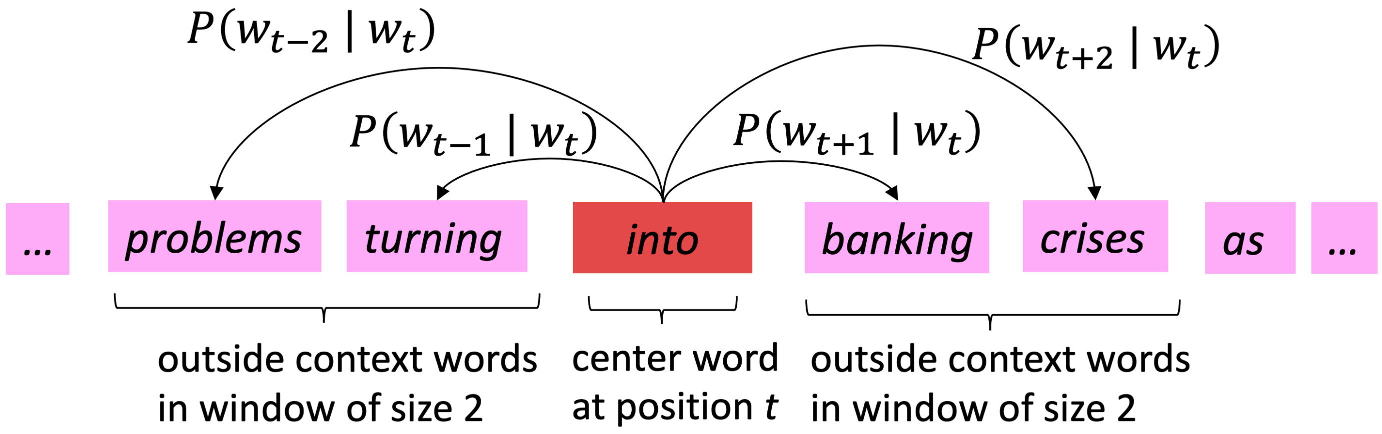
$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

θ is all variables to be optimized

Credit: Stanford CS224n, Winter 2022

How to Model a Word?

- Skip-gram [Mikolov et al., 2013]



sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

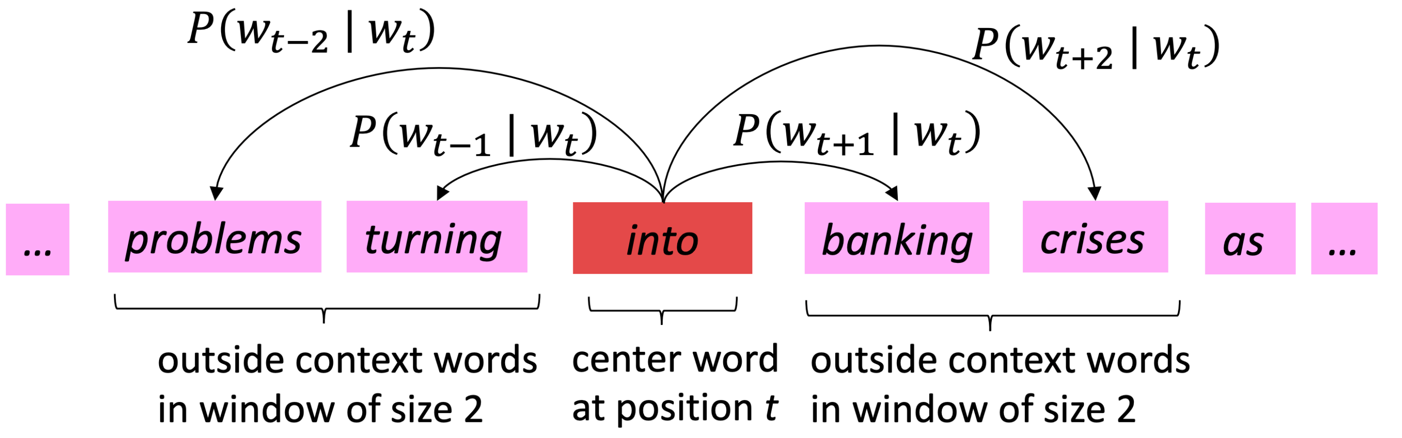
$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Minimizing objective function $\Leftrightarrow$ Maximizing predictive accuracy

Credit: Stanford CS224n, Winter 2022

How to Model a Word?

- Skip-gram [Mikolov et al., 2013]



sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\ne 0}} \boxed{\log P\left(w_{t+j} \mid w_t; \theta\right)}$$

?

Minimizing objective function ⟺ Maximizing predictive accuracy

Credit: Stanford CS224n, Winter 2022

How to Model a Word?

- Skip-gram [Mikolov et al., 2013]

Question: How to calculate $P\left(w_{t+j} \mid w_t; \theta\right)$?

Answer: We will *use two* vectors per word *w*:

- v_w when *w* is a center word
- u_w when *w* is a context word

Then for a center word *c* and a context word *o*:

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

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \ne 0}} \boxed{\log P\left(w_{t+j} \mid w_t; \theta\right)}$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Credit: Stanford CS224n, Winter 2022

How to Model a Sequence of Words?

- (Simple/Vanilla/Elman) Recurrent Neural Network (RNN) [Elman, 1990]

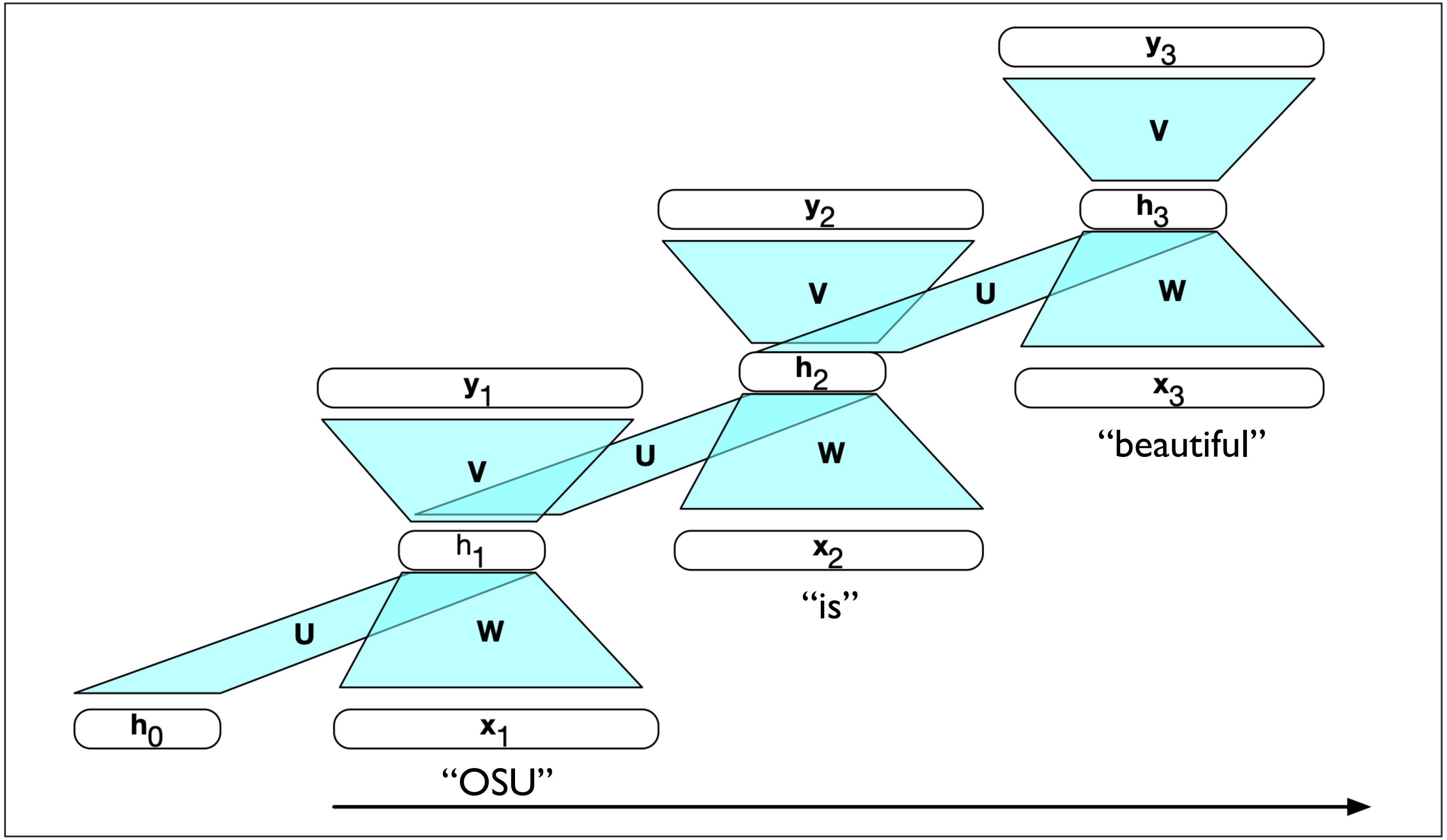


Image source: Jurafsky & Martin

How to Model a Sequence of Words?

- Recurrent Neural Network (RNN) [Elman, 1990]

A feedforward network

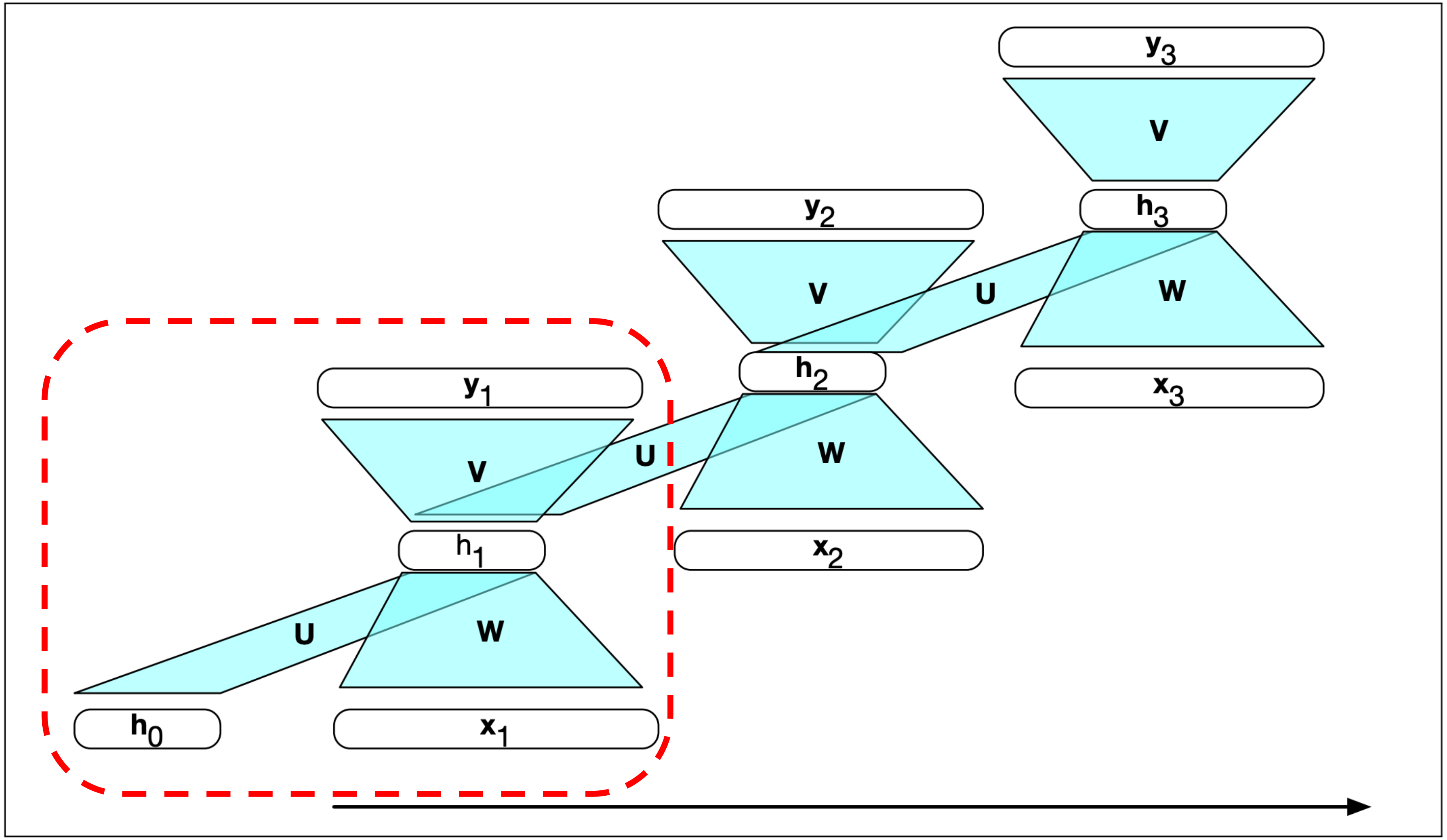
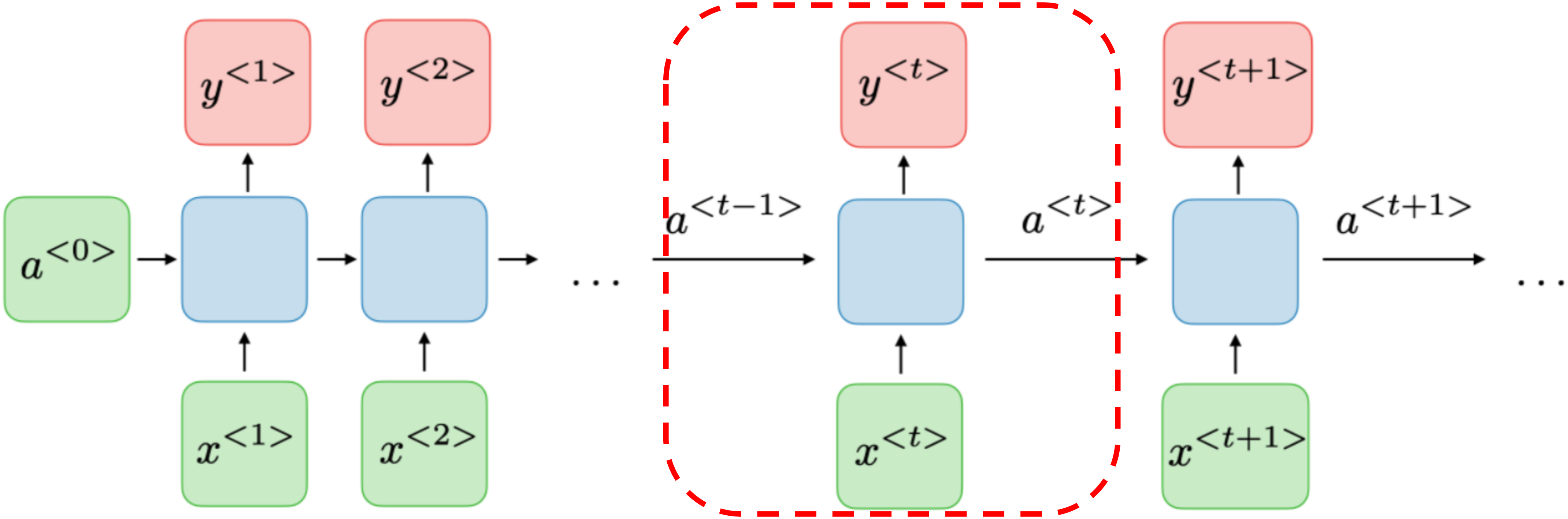


Image source: Jurafsky & Martin

How to Model a Sequence of Words?

- Recurrent Neural Network (RNN) [Elman, 1990]



For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

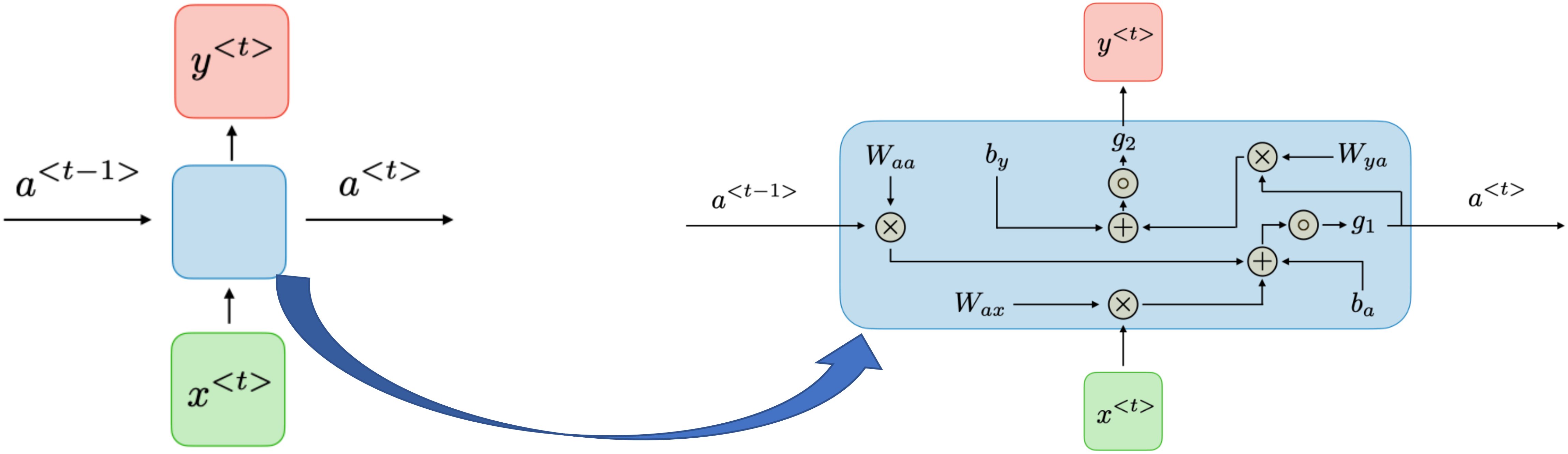
$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.

Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture

How to Model a Sequence of Words?

- Recurrent Neural Network (RNN) [Elman, 1990]



For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

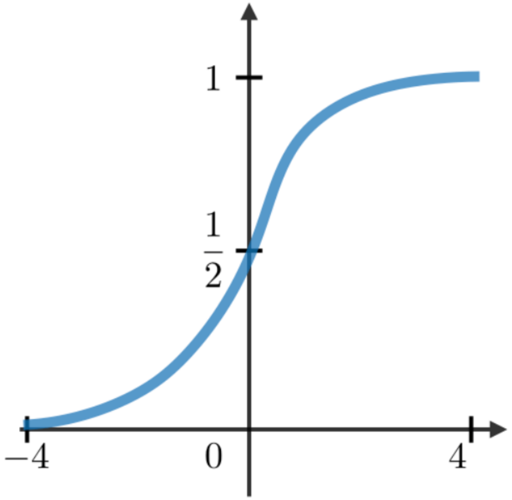
$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

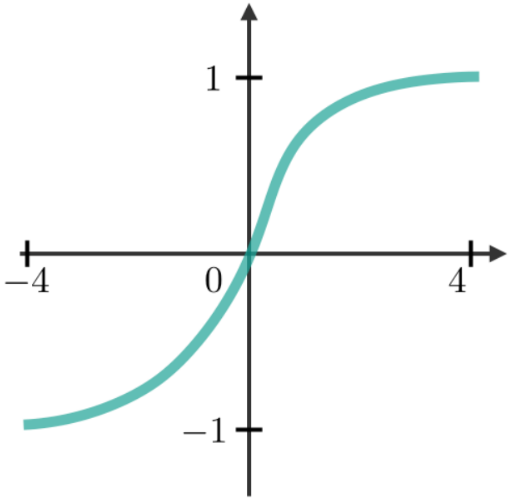
where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.

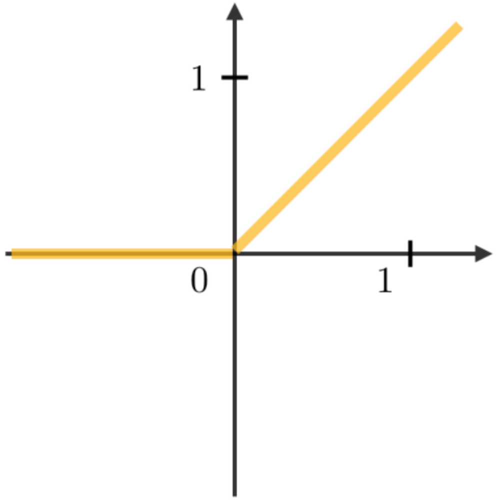
How to Model a Sequence of Words?

- Recurrent Neural Network (RNN) [Elman, 1990]
  - What are the commonly used activation functions?

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	$g(z) = \max(0, z)$

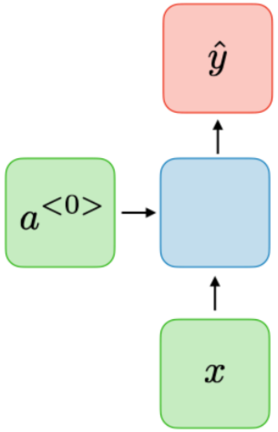
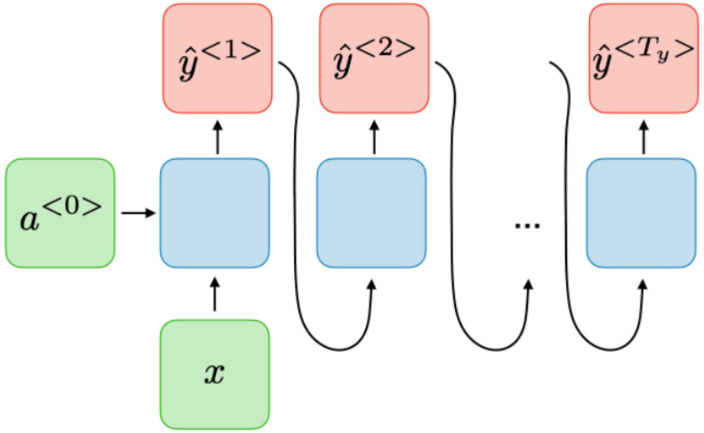






Applications of RNNs

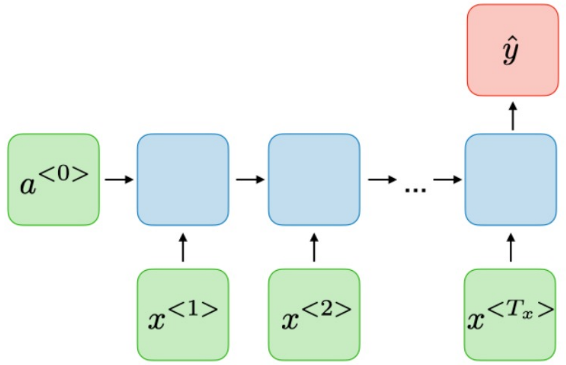
T_x (T_y): Number of timesteps on the input (output) side.

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation

Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture

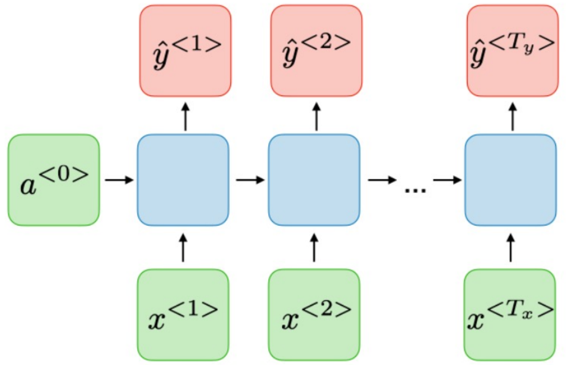
Applications of RNNs

Many-to-one
$T_x > 1, T_y = 1$



Sentiment classification

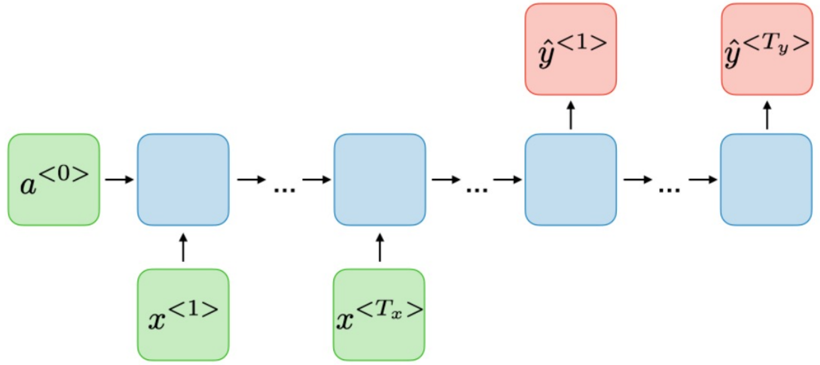
Many-to-many
$T_x = T_y$



Name entity recognition

(sequence-to-sequence)

Many-to-many
$T_x \neq T_y$

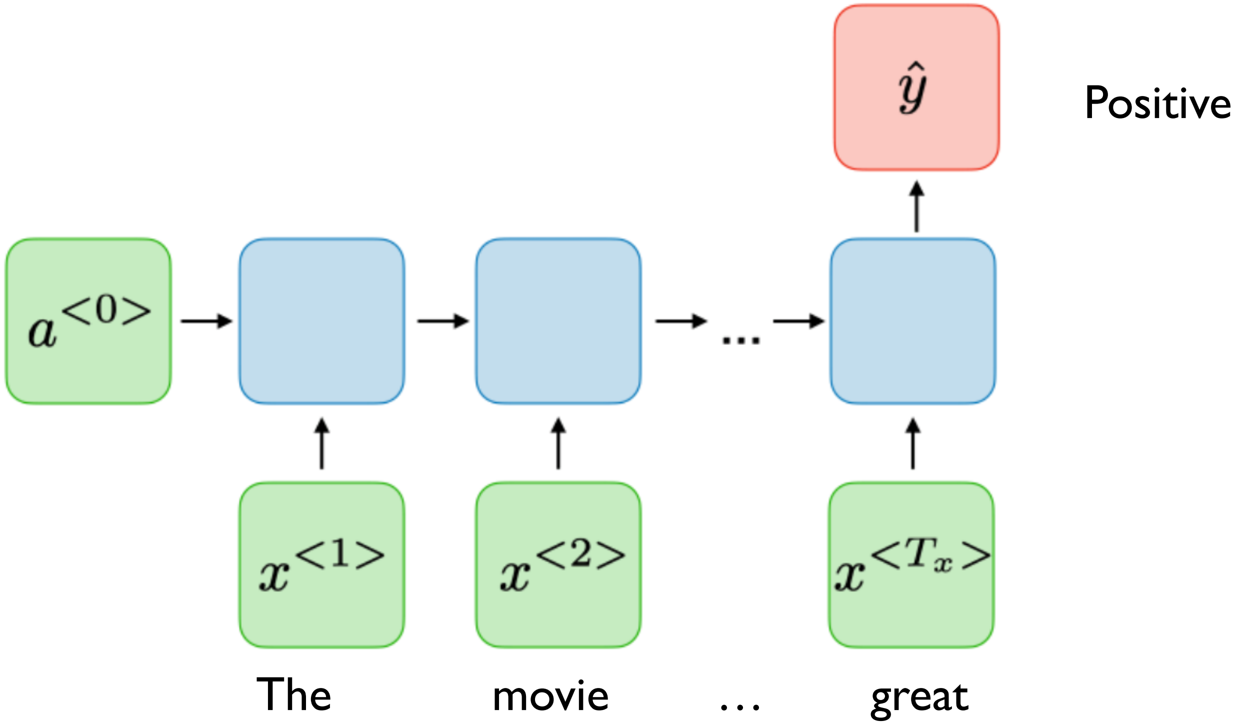


Machine translation

Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture

Loss function of RNNs

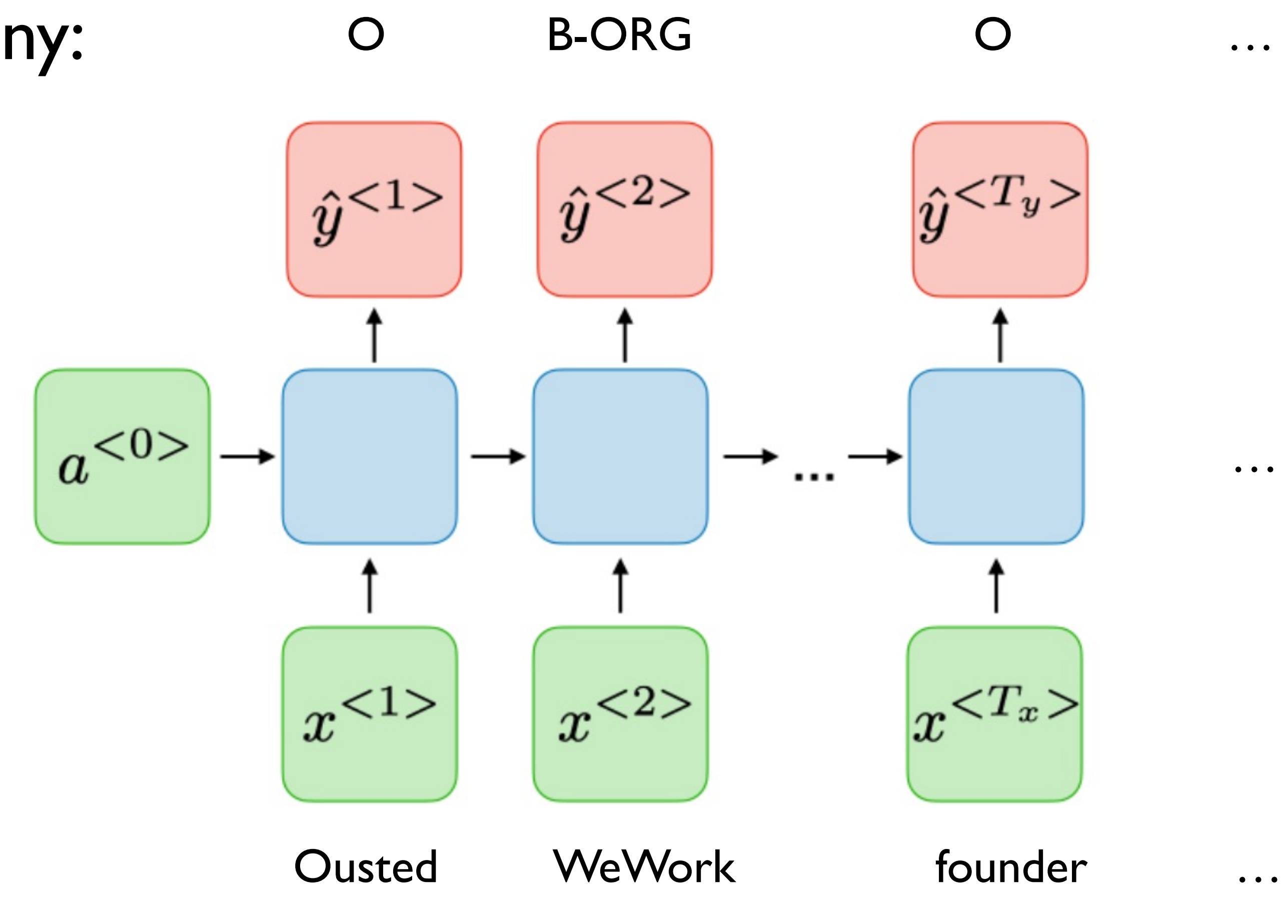
Type Many-to-one:



Example: Sentiment Analysis
Loss: Negative log likelihood of gold label

Loss function of RNNs

Type Many-to-Many:



Example: Named Entity Recognition

Loss: Negative log likelihood of gold labels, summed over all time steps

Optimization of RNNs

“Backpropagation through time”

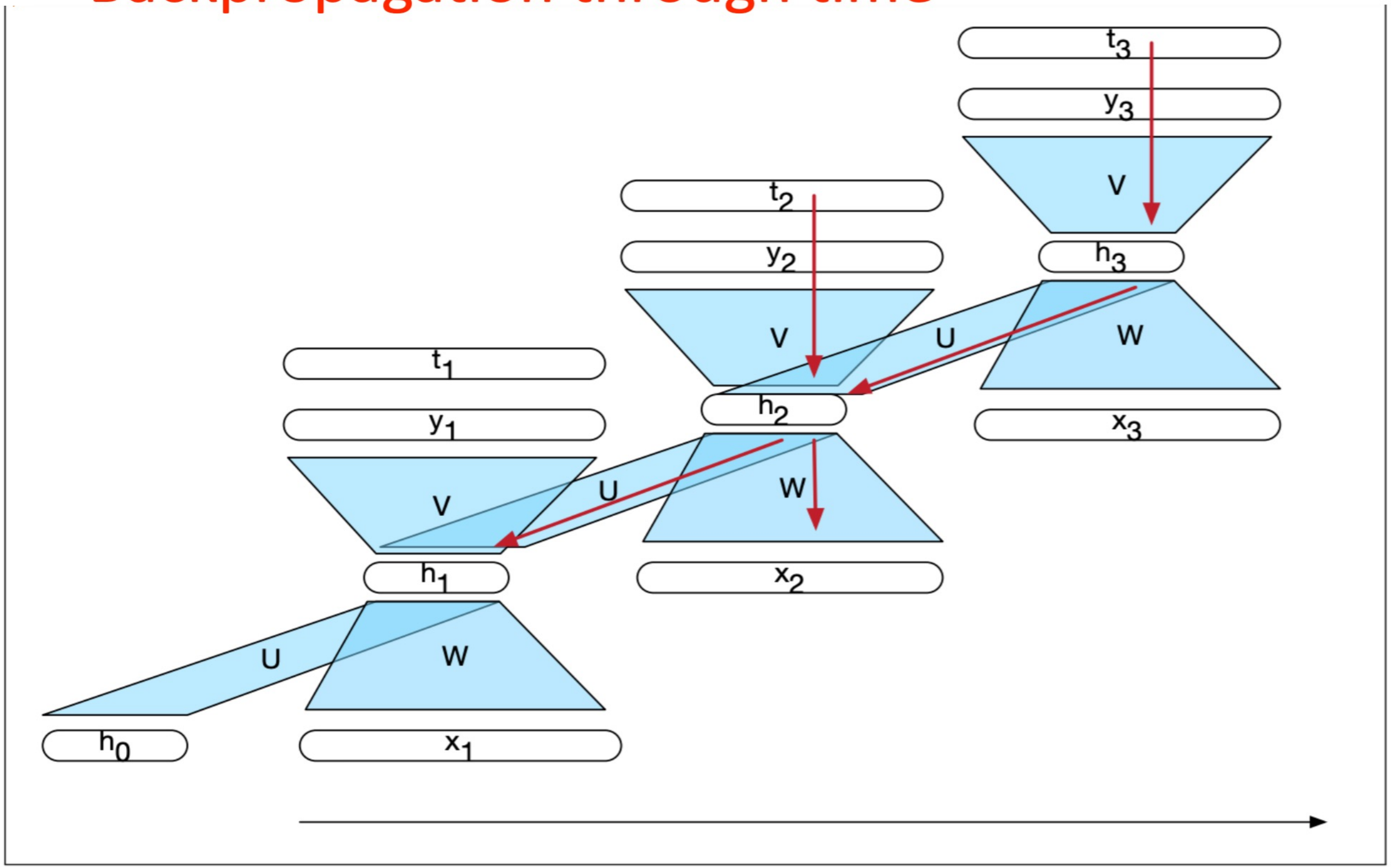
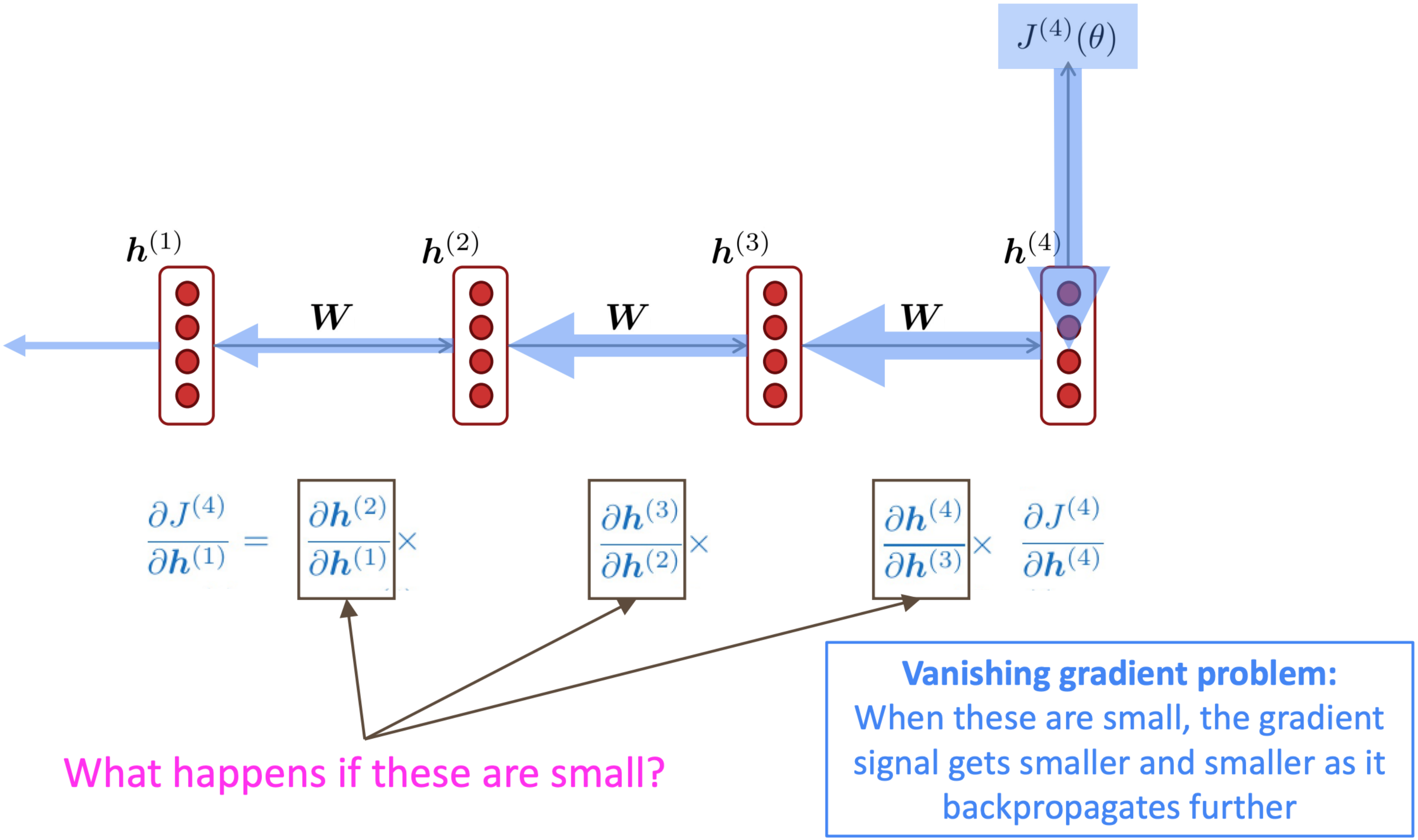


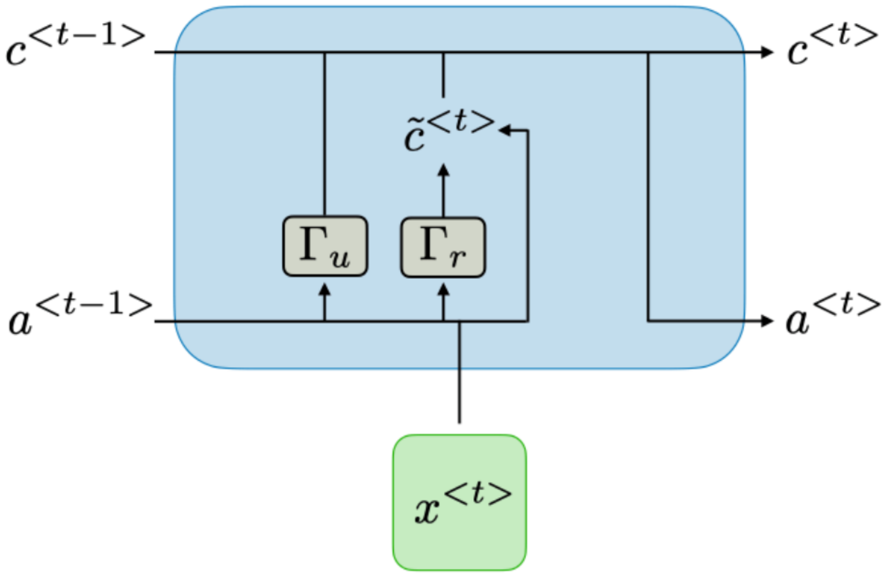
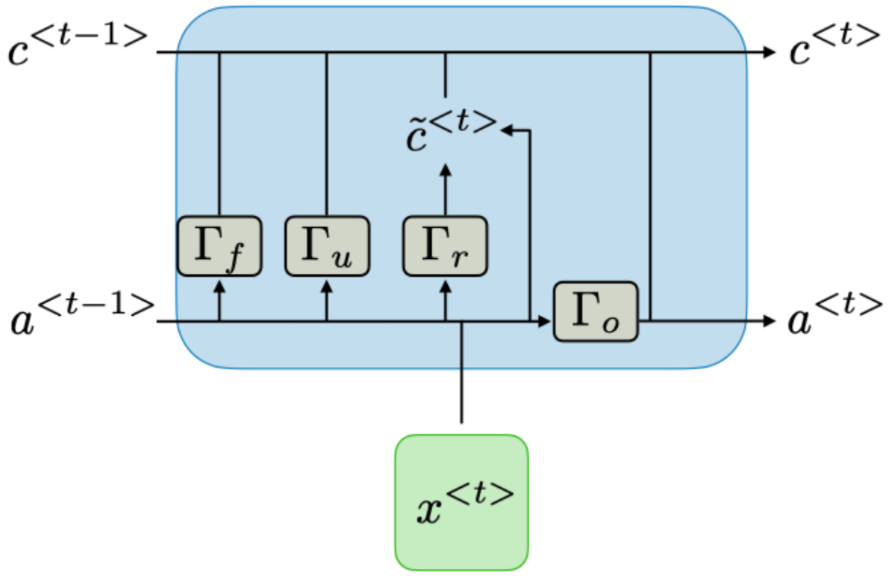
Image source: Jurafsky & Martin

Optimization of RNNs: Vanishing/Exploding Gradient



Source: https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture06-fancy-rnn.pdf

Other Variants of RNNs

Characterization	**Gated Recurrent Unit** (GRU)	**Long Short-Term Memory** (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Dependencies		

Remark: the sign $\star$ denotes the element-wise multiplication between two vectors.

Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture

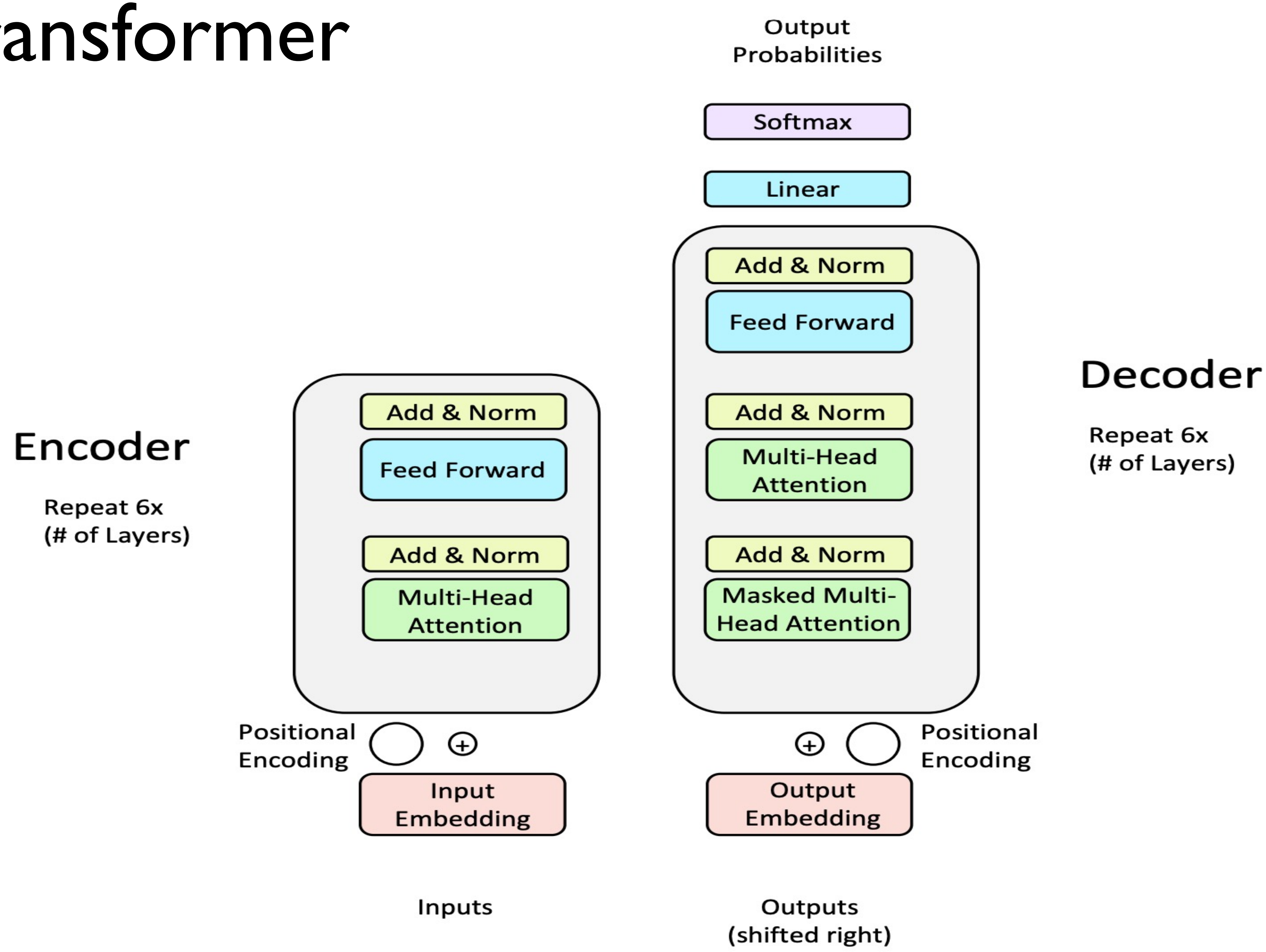
Transformer

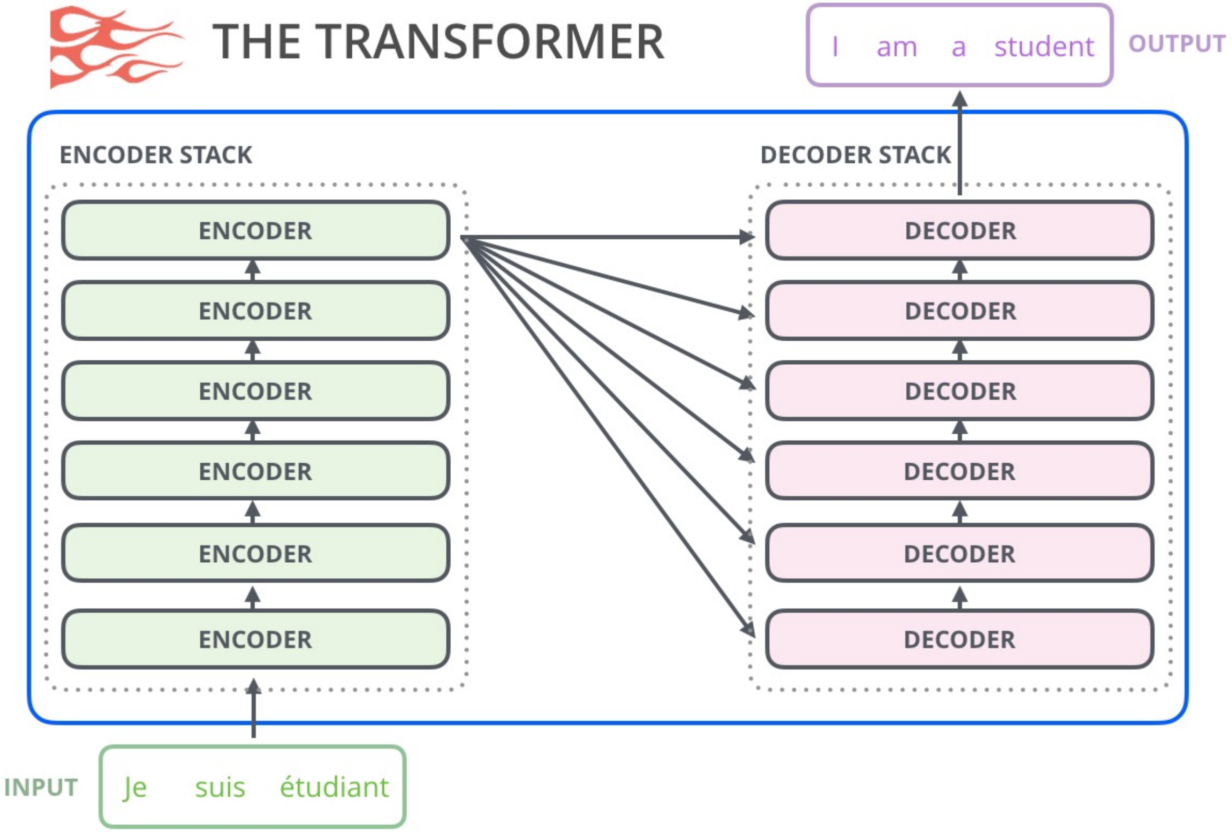
Vaswani et al., "Attention is all you need," 2017.

Used in (almost) every state-of-the-art NLP method!

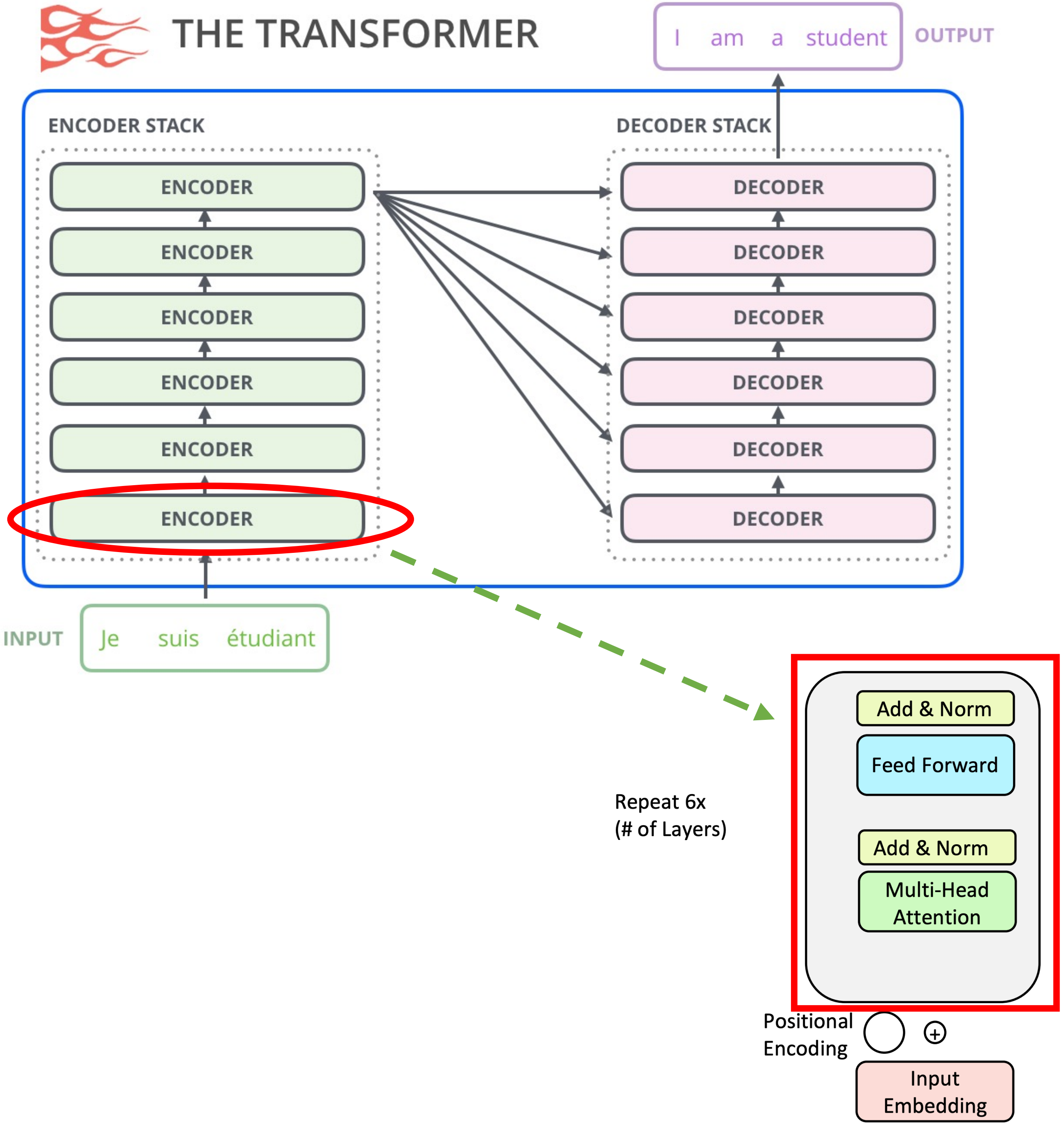
Source: https://movieweb.com/transformers-projects-annoucement-paramount/

Transformer

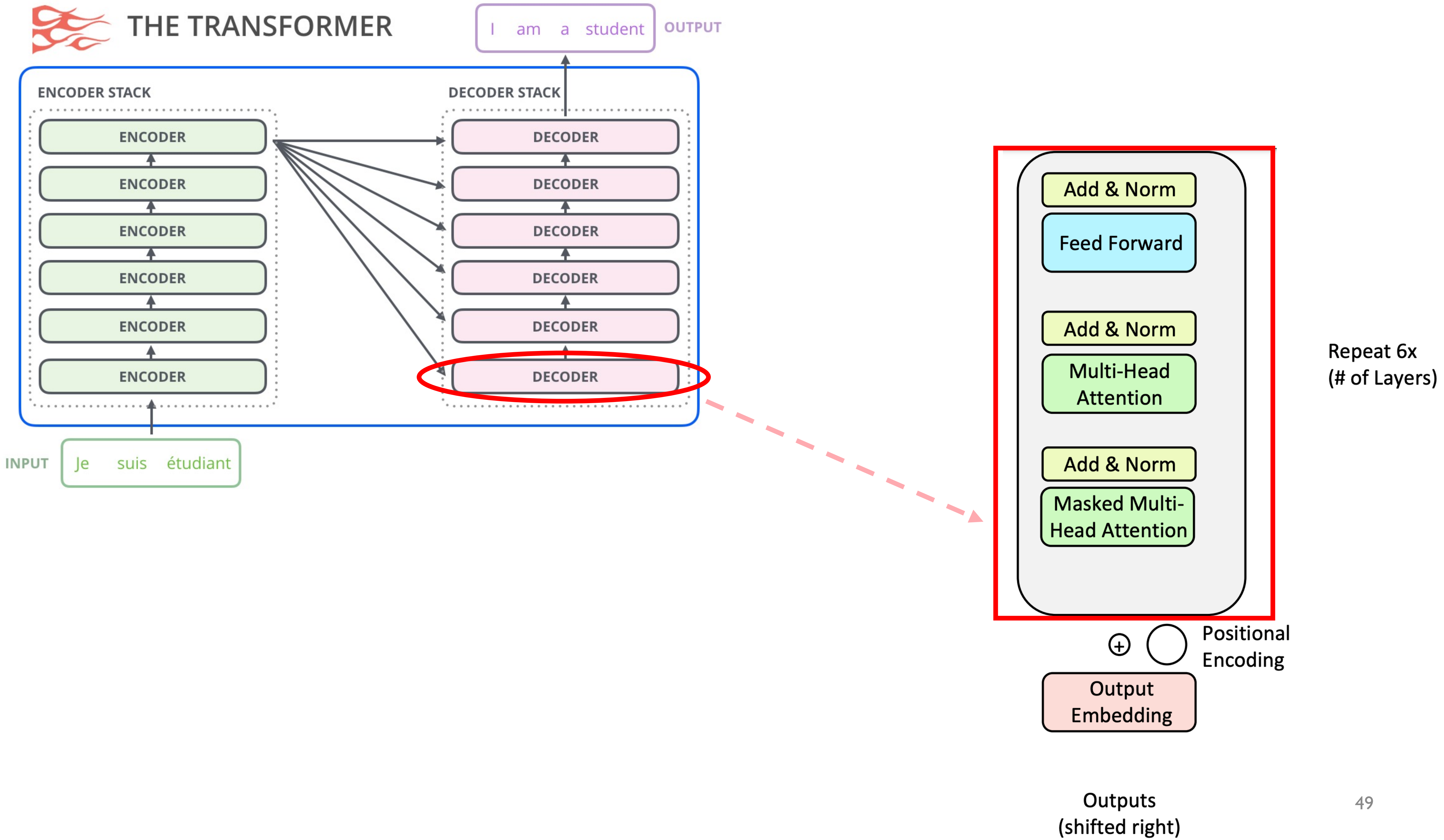




Source: https://jalammar.github.io/illustrated-gpt2/

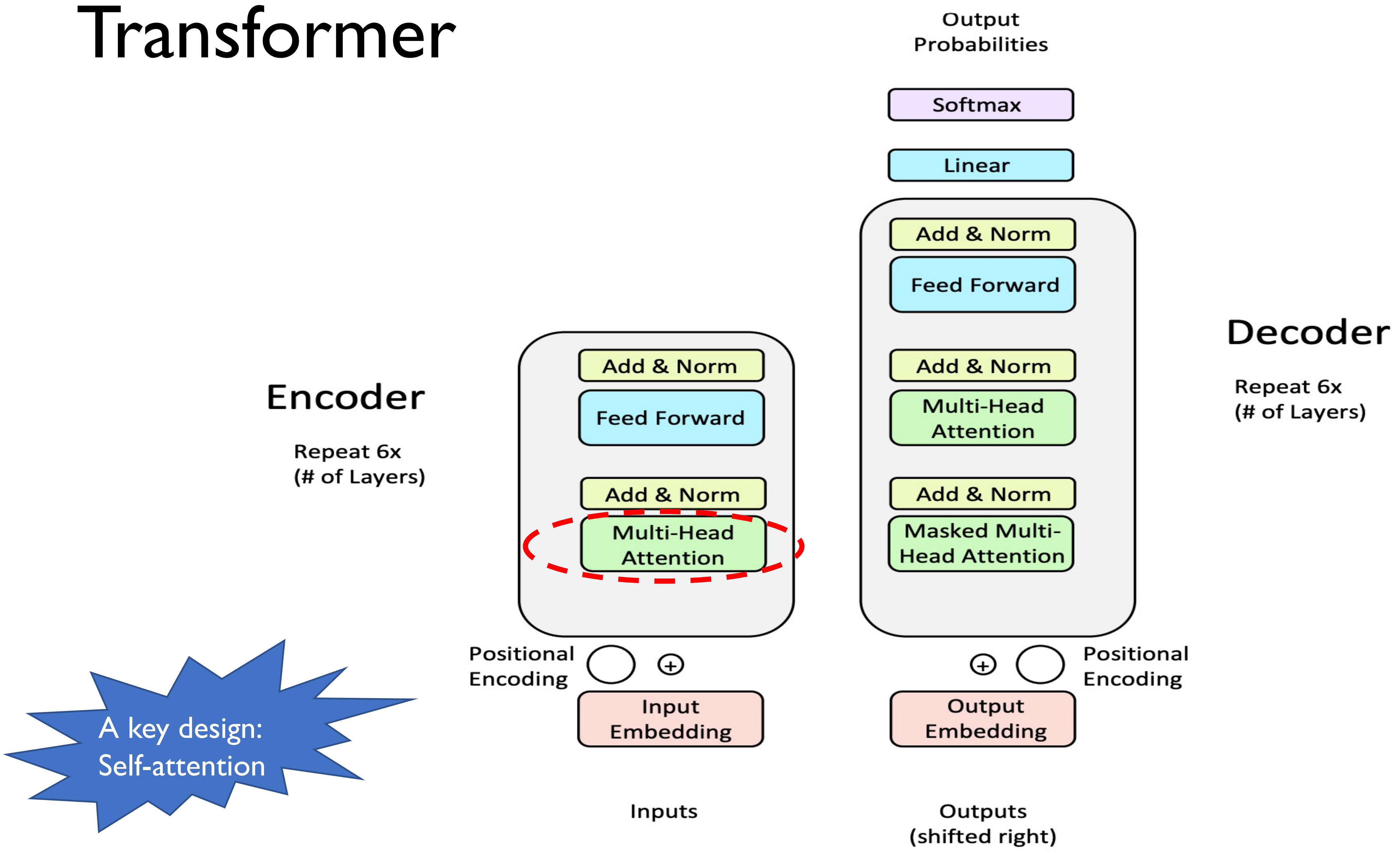


THE TRANSFORMER
I am a student
OUTPUT
ENCODER STACK
DECODER STACK
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant
Repeat 6x
(# of Layers)
Add & Norm
Feed Forward
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding



THE TRANSFORMER
I am a student
OUTPUT
ENCODER STACK
DECODER STACK
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Repeat 6x
(# of Layers)
Positional Encoding
Output Embedding
Outputs
(shifted right)

Transformer



Recipe for Self-Attention in the Transformer Encoder

- Step 1: For each word x_i, calculate its **query**, **key**, and **value**.

$$q_i = W^Q x_i \quad k_i = W^K x_i \quad v_i = W^V x_i$$

- Step 2: Calculate attention score between **query and keys.**

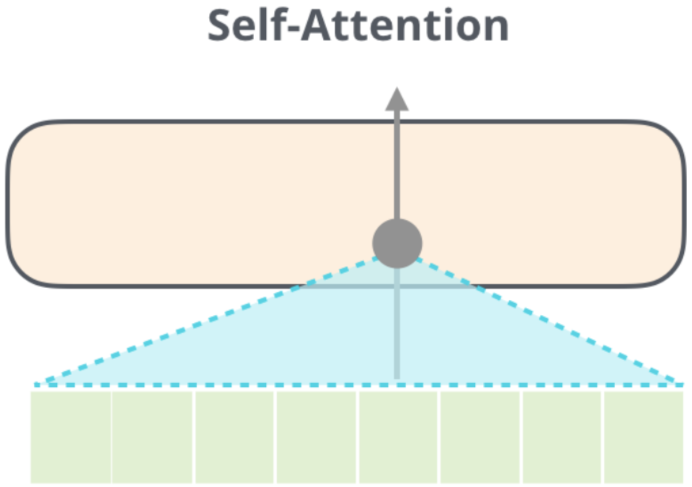
$$e_{ij} = q_i \cdot k_j$$

- Step 3: Take the softmax to normalize attention scores.

$$\alpha_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

- Step 4: Take a weighted sum of **values**.

$$Output_i = \sum_j \alpha_{ij} v_j$$



Credit: Stanford CS224n, Winter 2022, https://jalammar.github.io/illustrated-gpt2/

Recipe for (Vectorized) Self-Attention in the Transformer Encoder

- Step 1: With embeddings stacked in X, calculate **queries**, **keys**, and **values**.

$$Q = XW^Q \qquad K = XW^K \qquad V = XW^V$$

- Step 2: Calculate attention scores between **query and keys.**

$$E = QK^T$$

- Step 3: Take the softmax to normalize attention scores.

$$A = softmax(E)$$

- Step 4: Take a weighted sum of **values**.

$$Output = AV$$

$$\boxed{Output = softmax\left(QK^T\right)V}$$

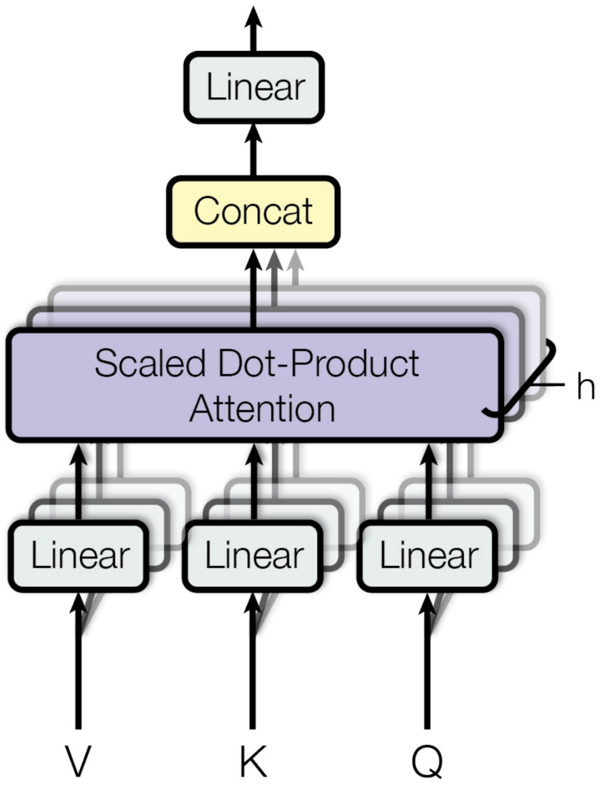
Credit: Stanford CS224n, Winter 2022

Multi-head attention?

- High-level idea:

Perform self-attention multiple (i.e., h) times in parallel and combine the results

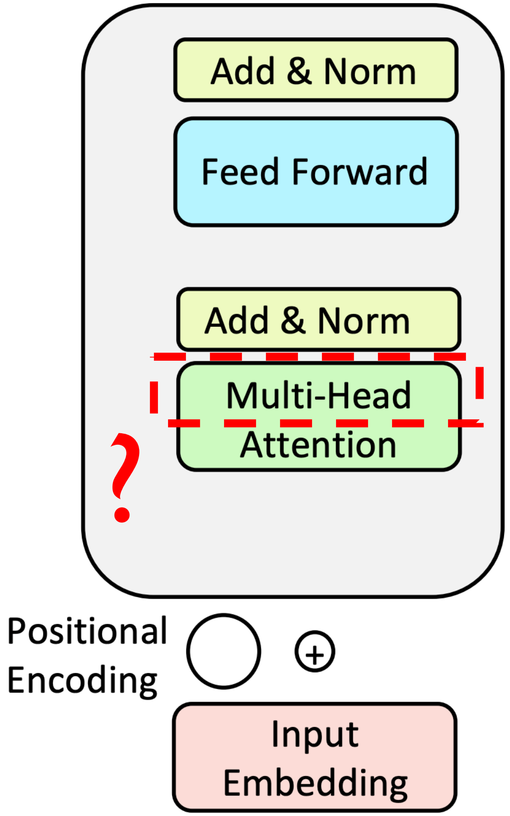
Multi-Head Attention



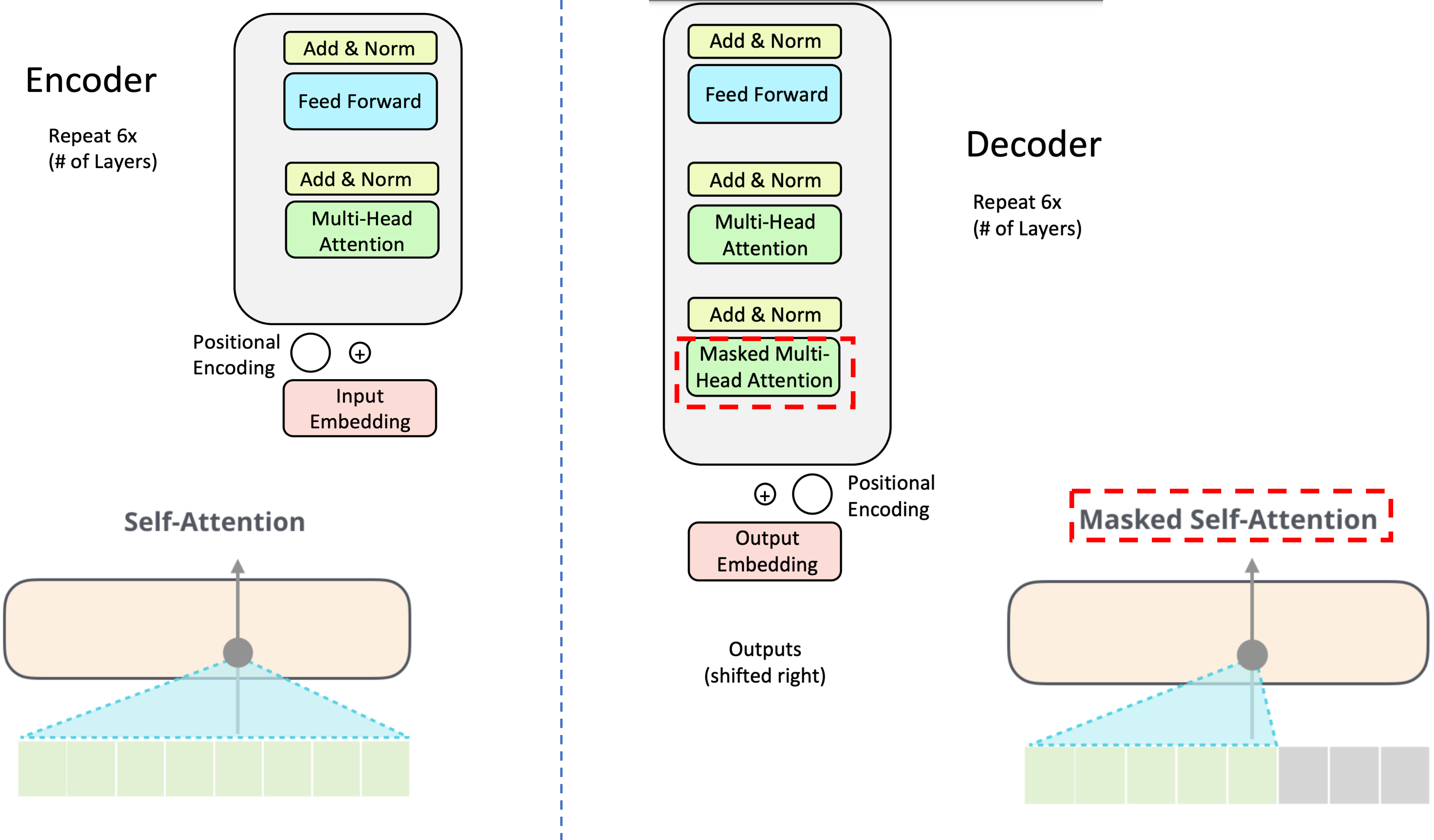
Wizards of the Coast, Artist: Todd Lockwood

Encoder

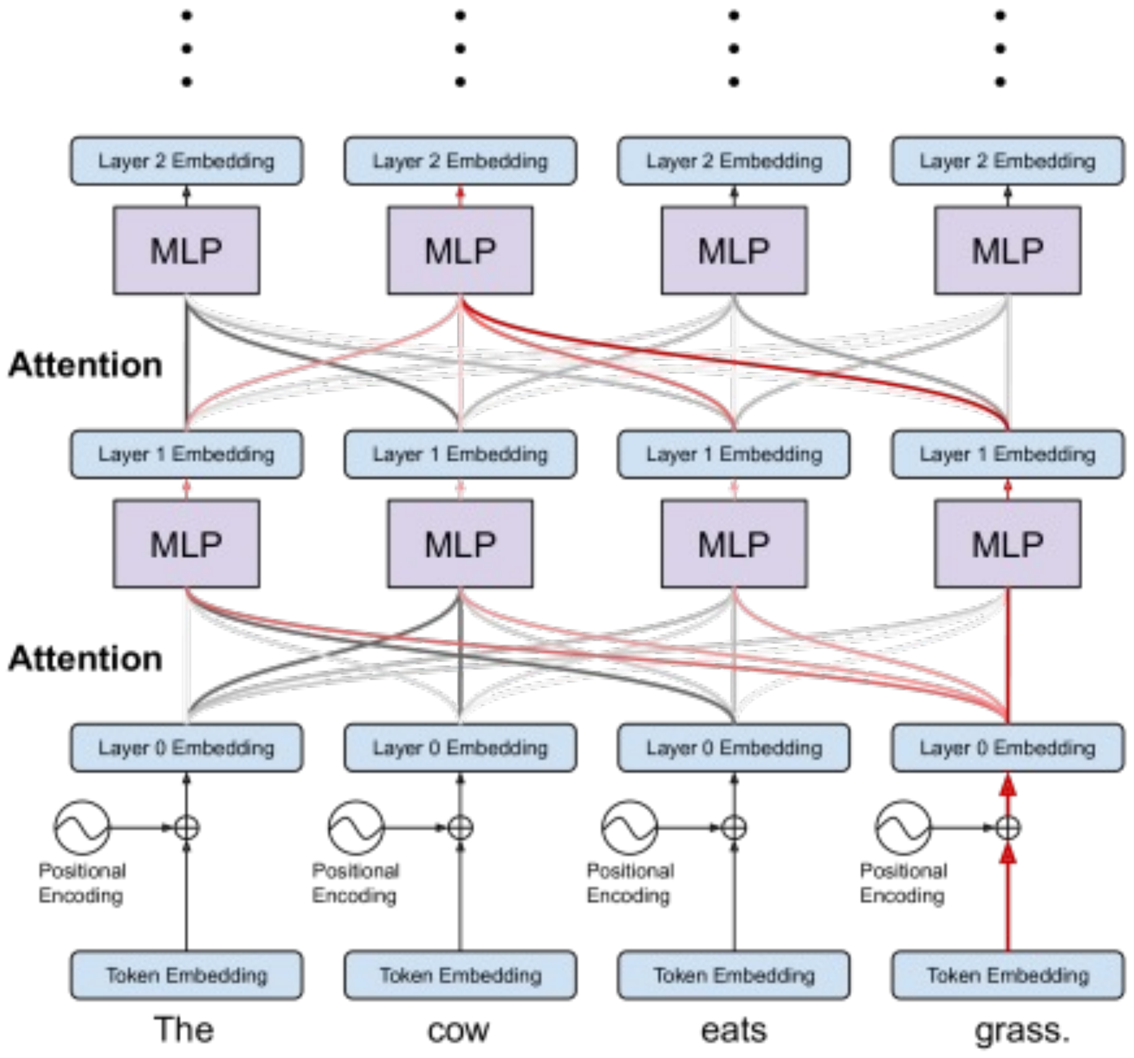
Repeat 6x
(# of Layers)



Credit: Stanford CS224n, Winter 2022


Encoder
Repeat 6x
(# of Layers)
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Self-Attention
Decoder
Repeat 6x
(# of Layers)
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
(shifted right)
Masked Self-Attention

A high-level view of transformer encoder

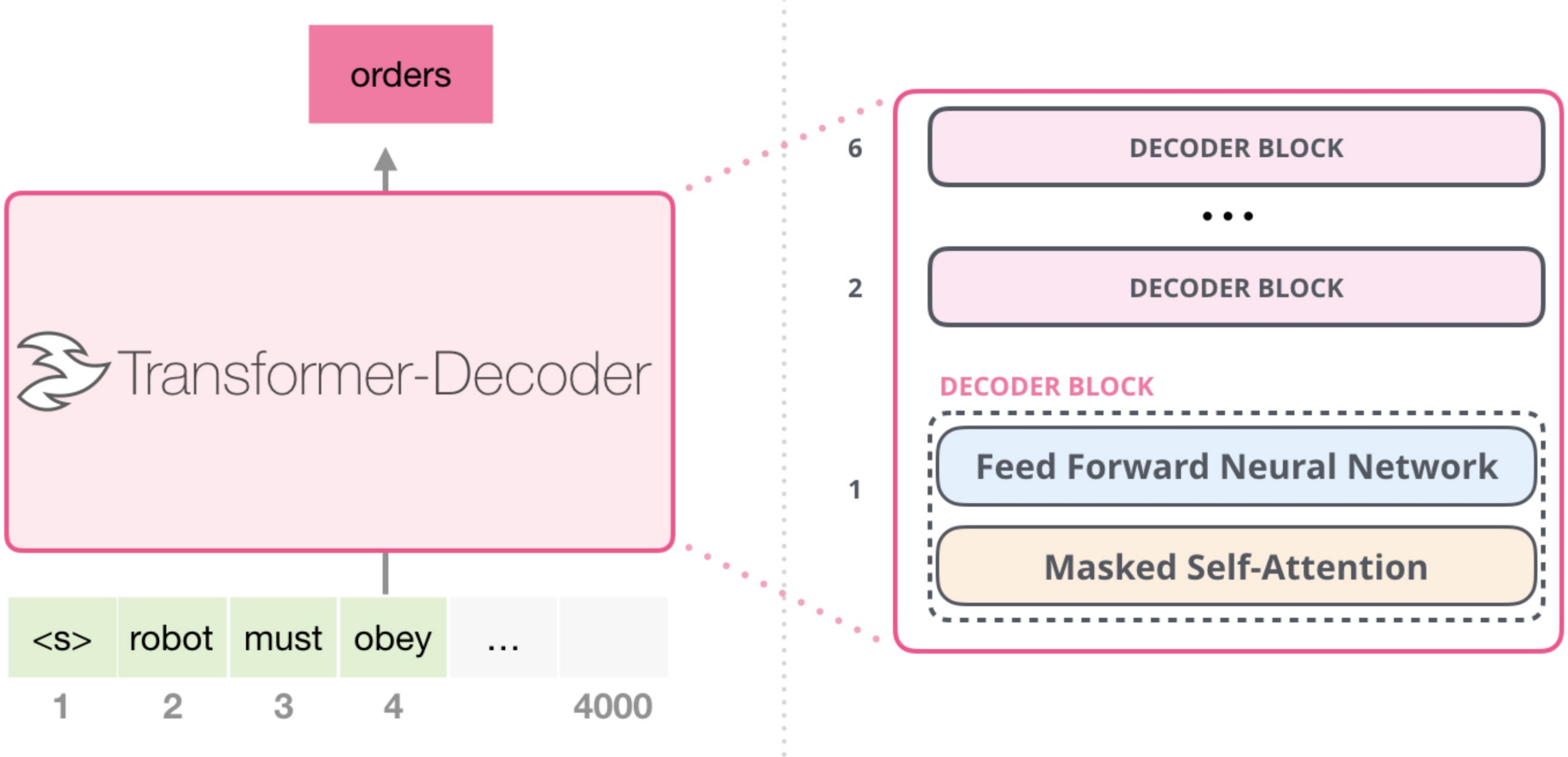


Input: a sequence of word vectors

Output: a sequence of "contextualized" word vectors

Figure credit: https://www.arxiv-vanity.com/papers/1908.04211

A high-level view of transformer decoder



Input: A sequence of words
Output: Probability distribution over the next word

Source: https://jalammar.github.io/illustrated-gpt2/

By now, we should know

- How to model a word

- How to model a sequence of words

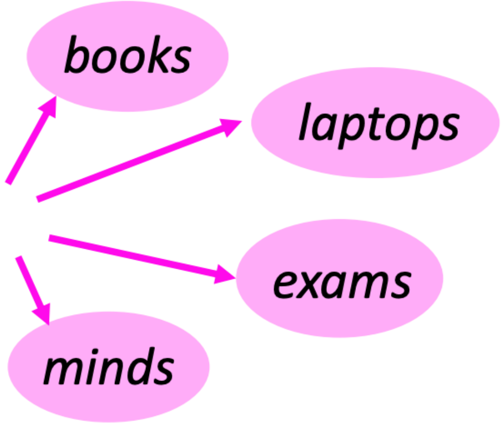
Next

- How to model a word?
- How to model a sequence of words?
- What is a “pre-trained” model?

Recall: What is a language model (LM)?

- **Language Modeling** is the task of predicting what word comes next

the students opened their ______



- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

- A system that does this is called a **Language Model**

A Simple RNN Language Model

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

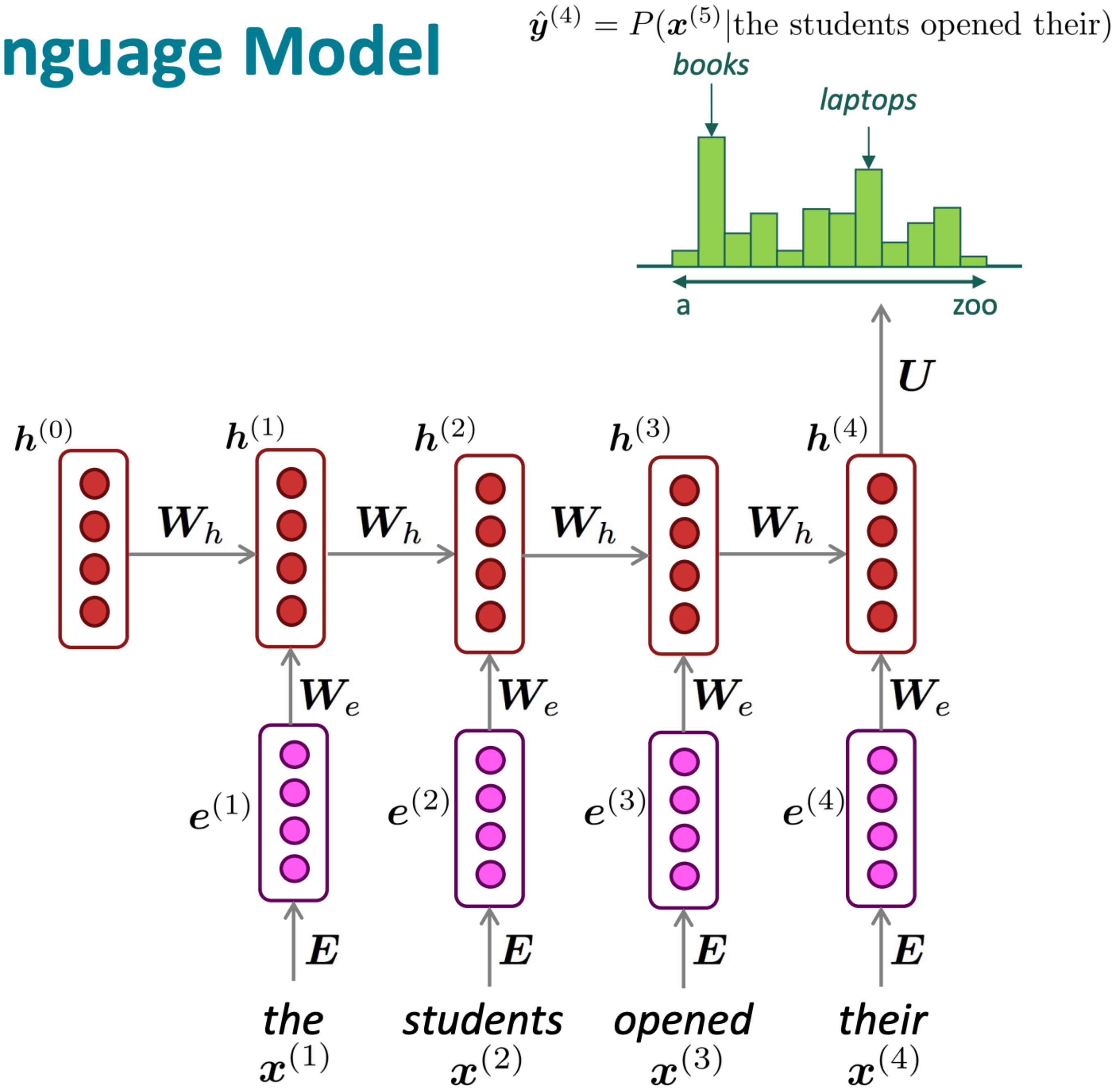
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

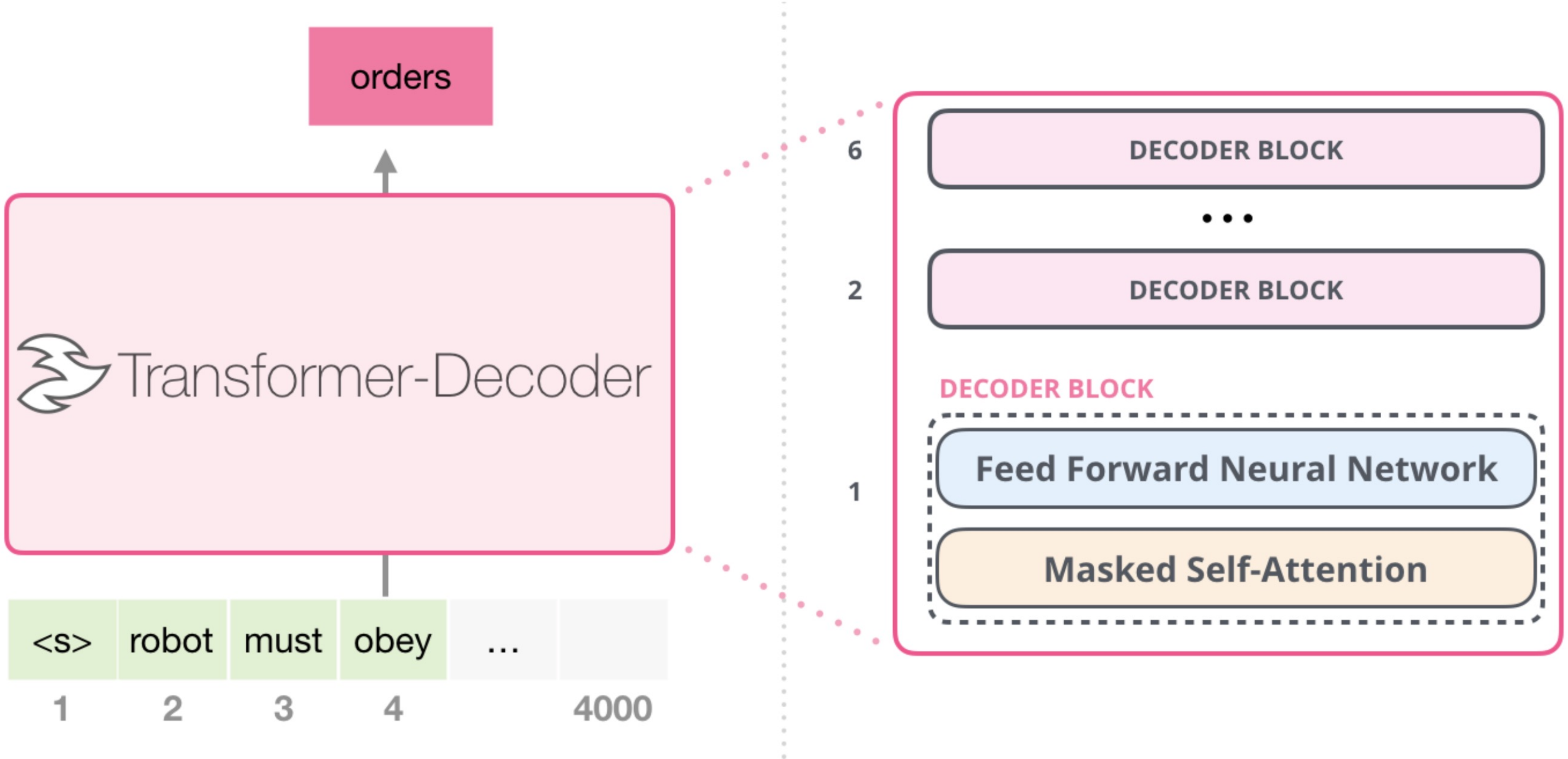
words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



Credit: Stanford CS224n, Winter 2022

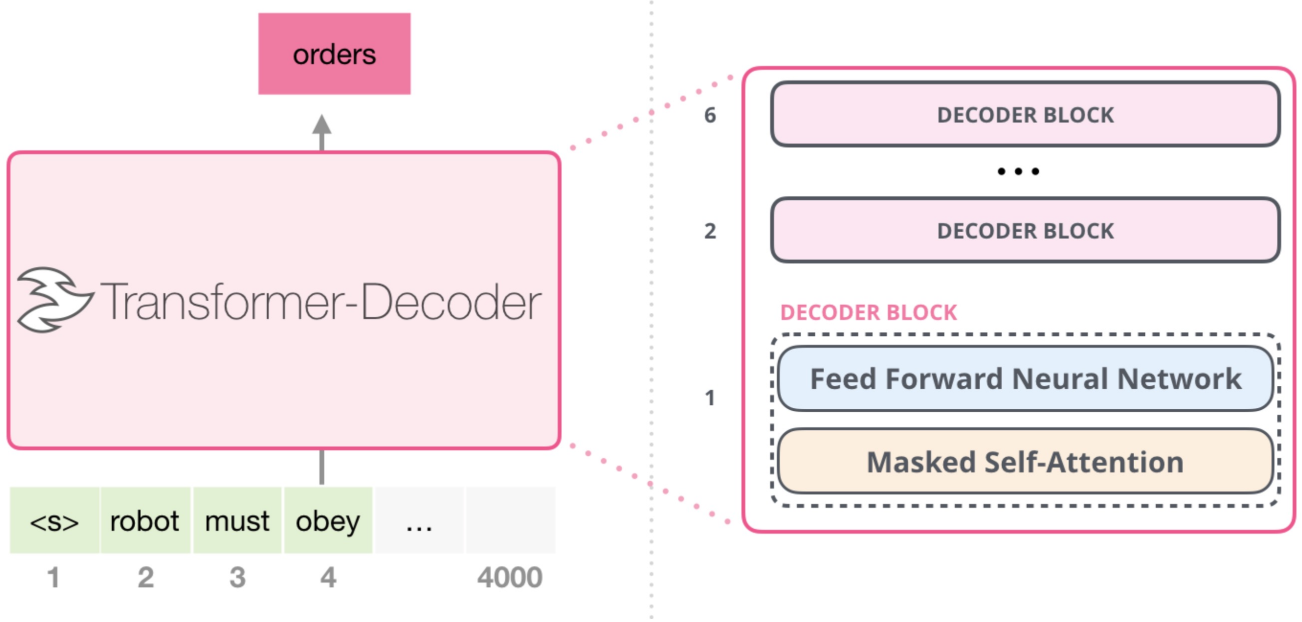
A Transformer Decoder based Language Model



Source: https://jalammar.github.io/illustrated-gpt2/

"Pre-training" a Transformer Decoder based Language Model

- Generative Pre-trained Transformer (GPT)
  - GPT, GPT-2, GPT-3, …



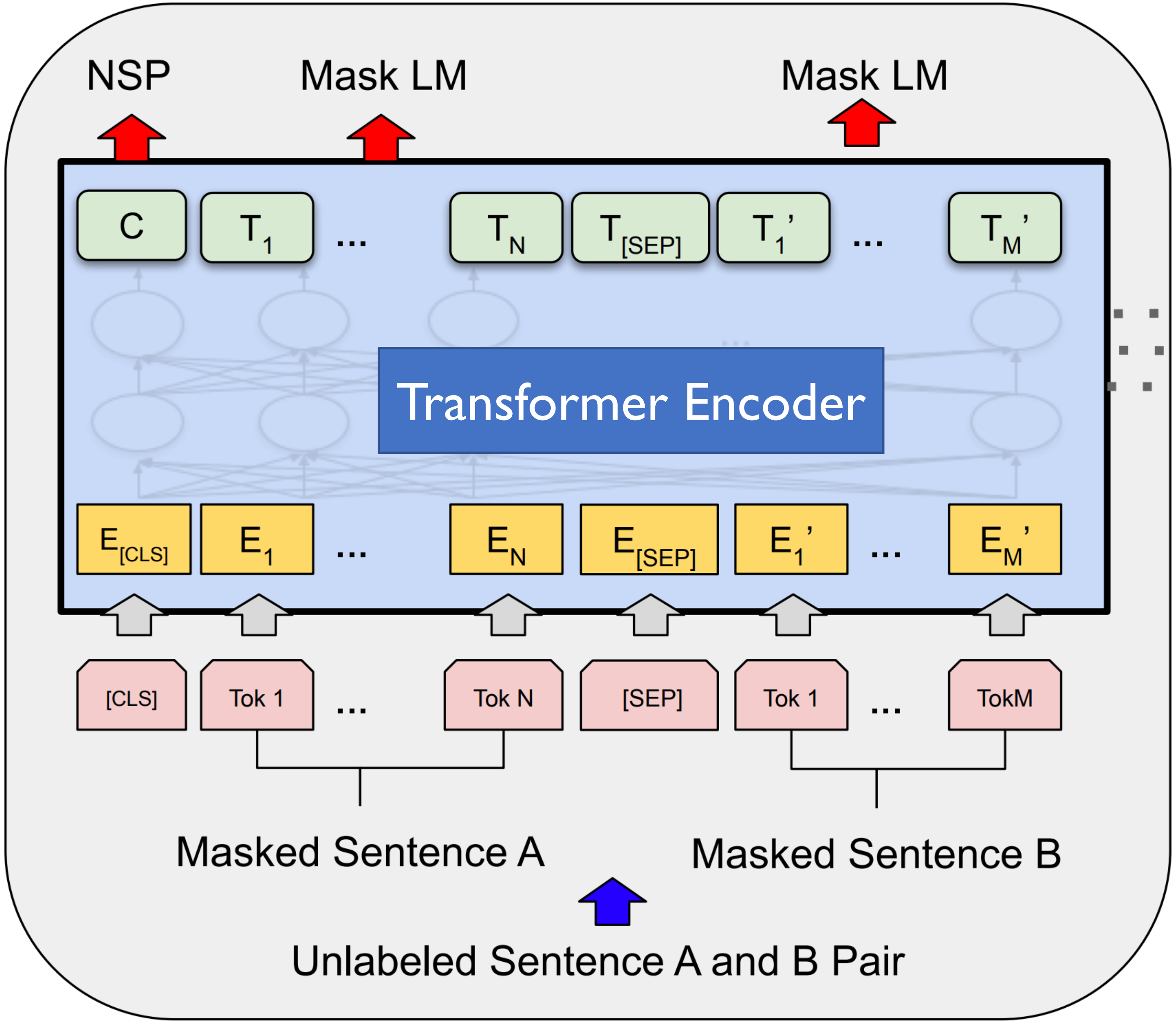
GPT-3 Training Data

Dataset	# Tokens	Weight in Training Mix
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

"self-supervision", "downstream task agnostic"

Source: https://jalammar.github.io/illustrated-gpt2/ & Wikipedia

Pre-training using Masked Language Modeling (MLM)



BERT [Devlin et al., 2019]

MLM example:

"located"

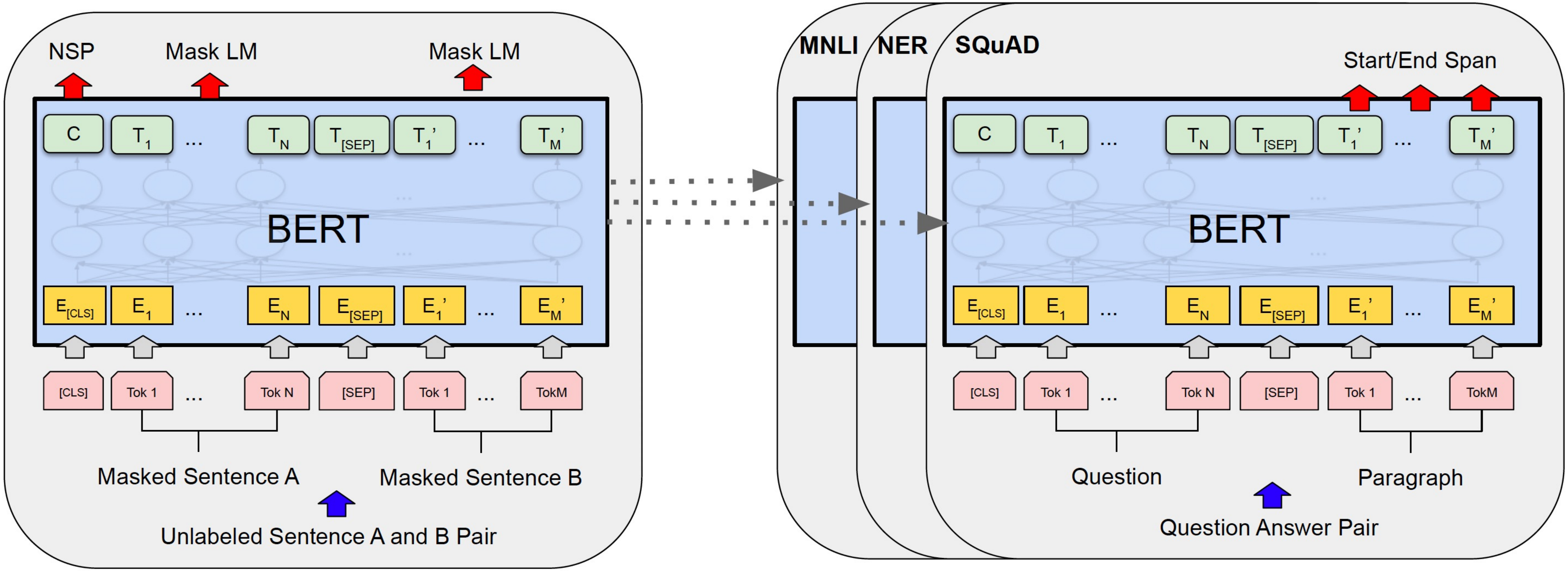
BERT

"Columbus is [MASK] in Ohio."

Bidirectional representation

Pre-training + Fine-tuning

BERT [Devlin et al., 2019]



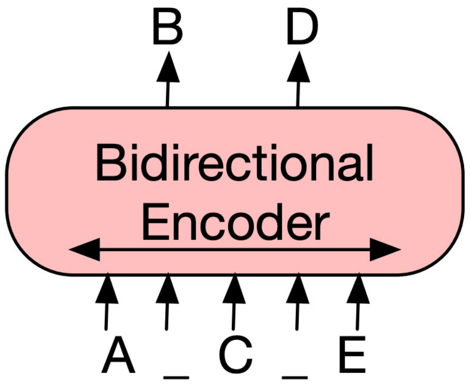
Pre-training

Fine-Tuning

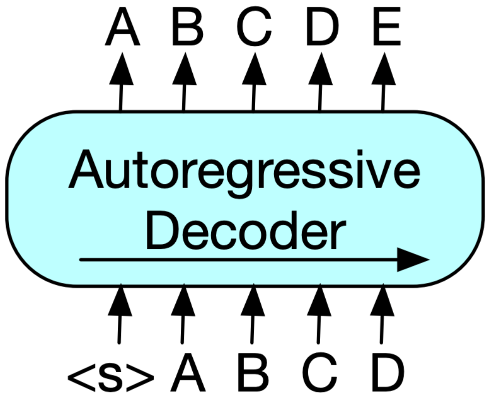
Self-supervision based on natural sentences

Task-specific data

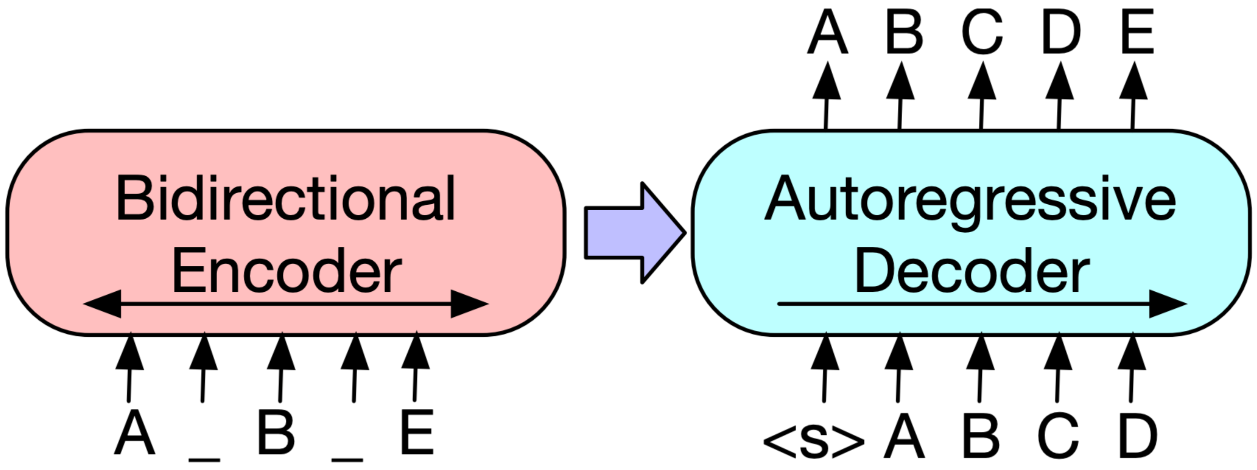
Denoising Sequence-to-Sequence Pre-training



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



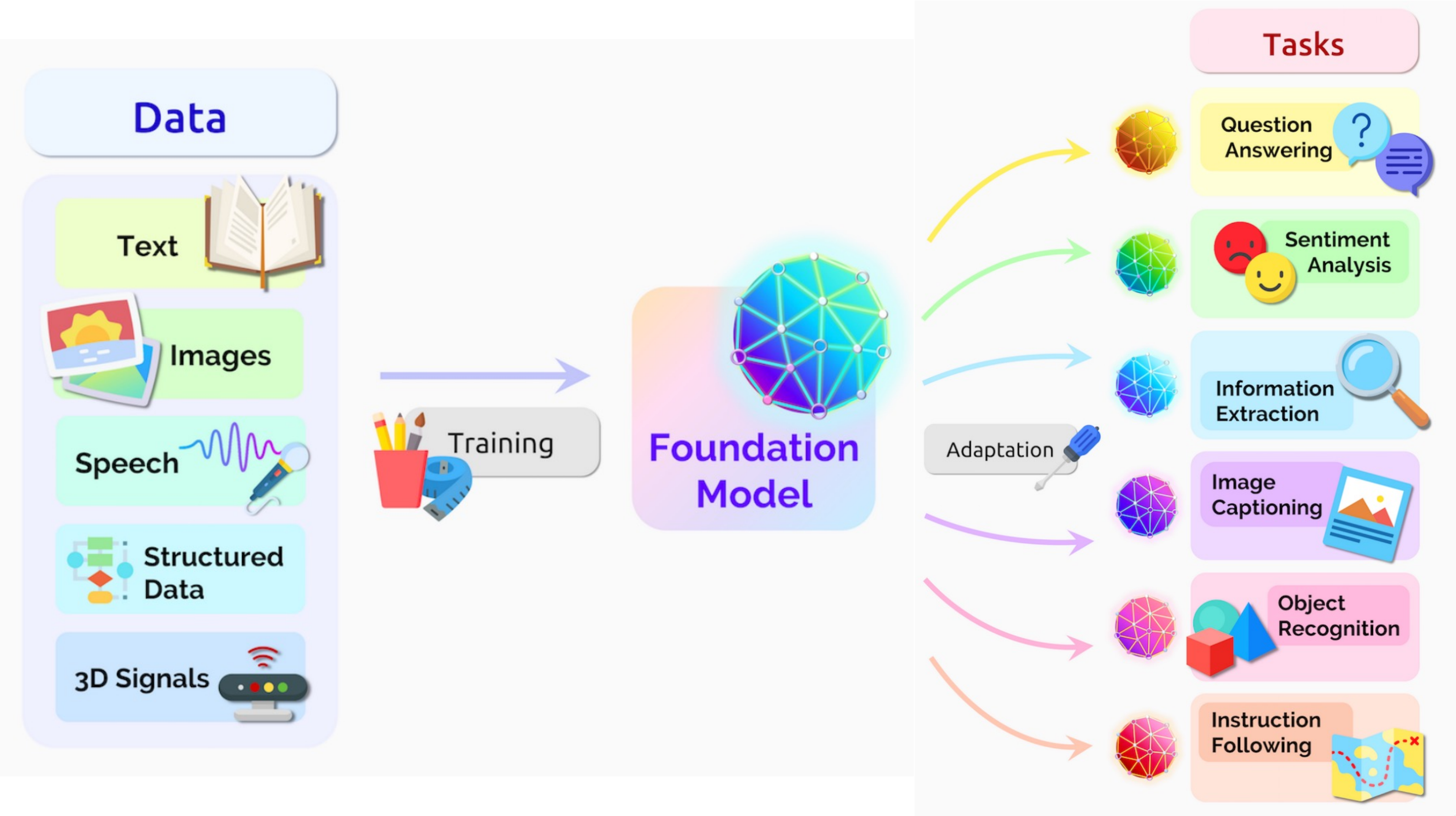
BART [Lewis et al., 2019]:

Pre-training sequence-to-sequence models

“Foundation Models”

(BERT, RoBERTa, BART, T5, GPT-3, PaLM…)

- Pre-trained on broad data (usually with self-supervised data at scale)
- Adaptable to a wide range of downstream tasks with minimal effort



“On the Opportunities and Risks of Foundation Models,” Stanford HAI, 2021.

Tutorial Structure

Part I (~75 mins):

- Tasks
- Deep Learning Models

Break (~15mins)

Part II: (~45 mins):

- Large Language Models
- Demo

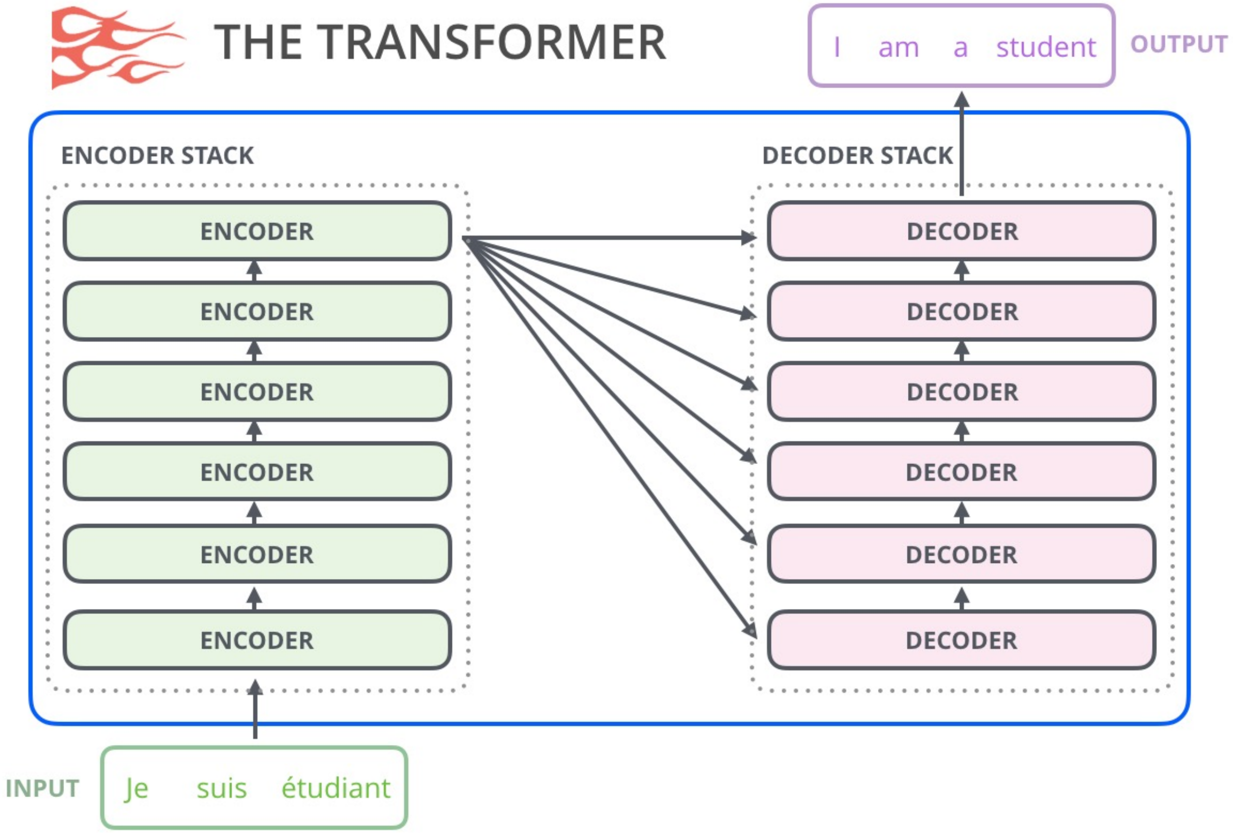
QA (~15 mins)

Part II:
Large Language Models
& Demo

Outline: Further Discussion on Large Language Models

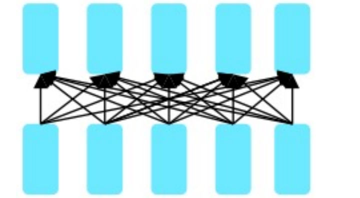
- An overview of popular large language models
- A general recipe of training large language models
- What can large language models do now?
- Promising future directions

Three Types of Language Models

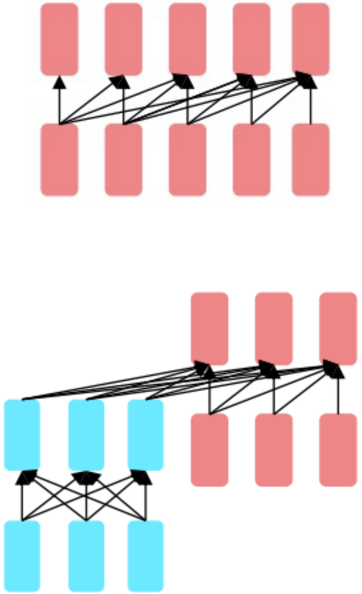


Source: https://movieweb.com/transformers-projects-annoucement-paramount/; https://jalammar.github.io/illustrated-gpt2

Recap: Three Types of Large Language Models



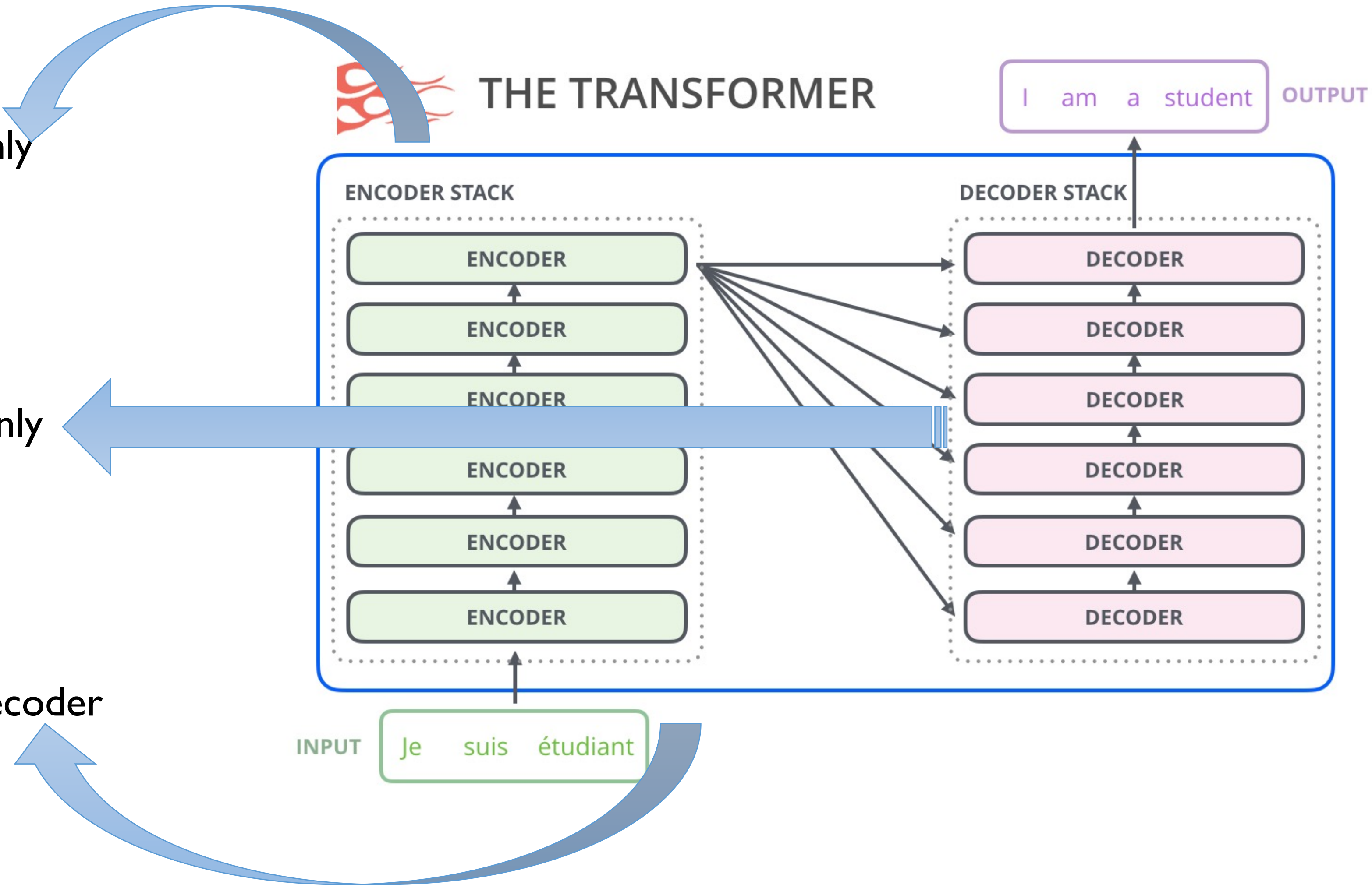
Type



Encoder-only

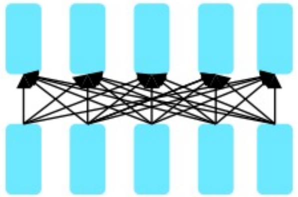
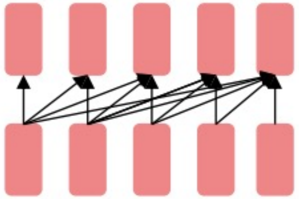
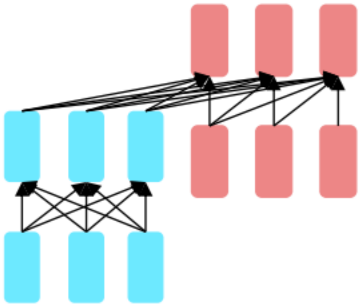
Decoder-only

Encoder-decoder



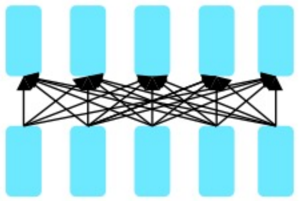
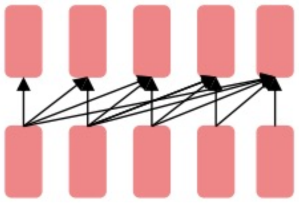
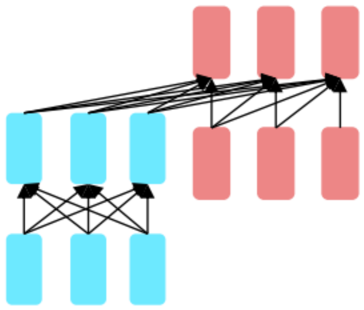
Source: Stanford CS224N: NLP with Deep Learning

Recap: Three Types of Large Language Models

	Type	Features	Exemplars
	Encoder-only	1. Gets bidirectional context – can condition on future! 2. Good at Natural Language Understanding (NLU)	BERT and its many variants, (e.g., RoBERTa, ALBERT) XLNet, ELECTRA
	Decoder-only		
	Encoder-decoder		

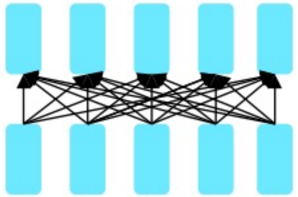
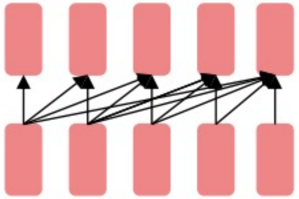
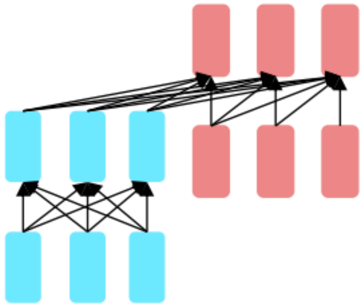
Source: Stanford CS224N: NLP with Deep Learning

Recap: Three Types of Large Language Models

	Type	Features	Exemplars
	Encoder-only	1. Gets bidirectional context – can condition on future! 2. Good at Natural Language Understanding (NLU)	BERT and its many variants, (e.g., RoBERTa, ALBERT) XLNet, ELECTRA
	Decoder-only	1. Predicting the next word 2. Good at Natural Language Generation (NLG)	GPT/GPT-2/GPT3, LaMDA, Gopher, PaLM
	Encoder-decoder		

Source: Stanford CS224N: NLP with Deep Learning

Recap: Three Types of Large Language Models

	Type	Features	Exemplars
	Encoder-only	1. Gets bidirectional context – can condition on future! 2. Good at Natural Language Understanding (NLU)	BERT and its many variants, (e.g., RoBERTa, ALBERT) XLNet, ELECTRA
	Decoder-only	1. Predicting the next word 2. Good at Natural Language Generation (NLG)	GPT/GPT-2/GPT3, LaMDA, Gopher, PaLM
	Encoder-decoder	Suitable for sequence-to-sequence tasks	T5, BART, Meena

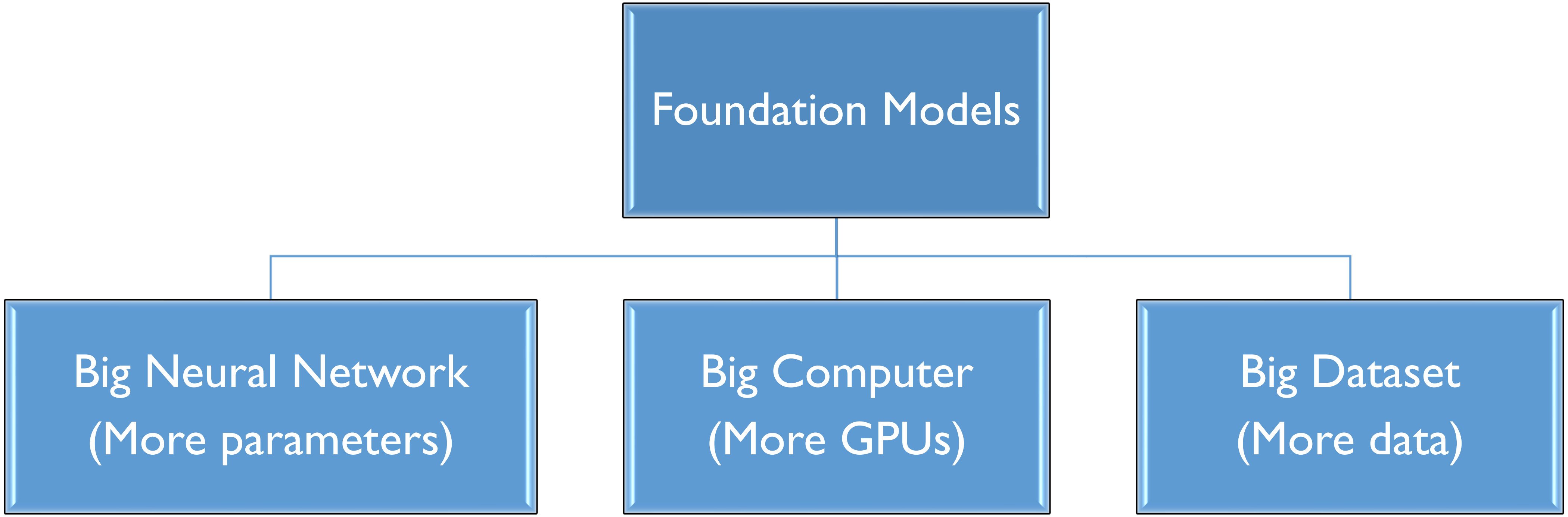
Source: Stanford CS224N: NLP with Deep Learning

Further Discussion on Large Language Models

- An overview of popular large language models
- **A general recipe of training large language models**
- What can large language models do now?
- Promising future directions

How to Train Large Language Models?

A Recipe for Modern LLMs!



Recipe Credit: [Ilya Sutskever's talk on HAI Spring Conference 2022: Foundation Models]

More Parameters: An Exponential Growth

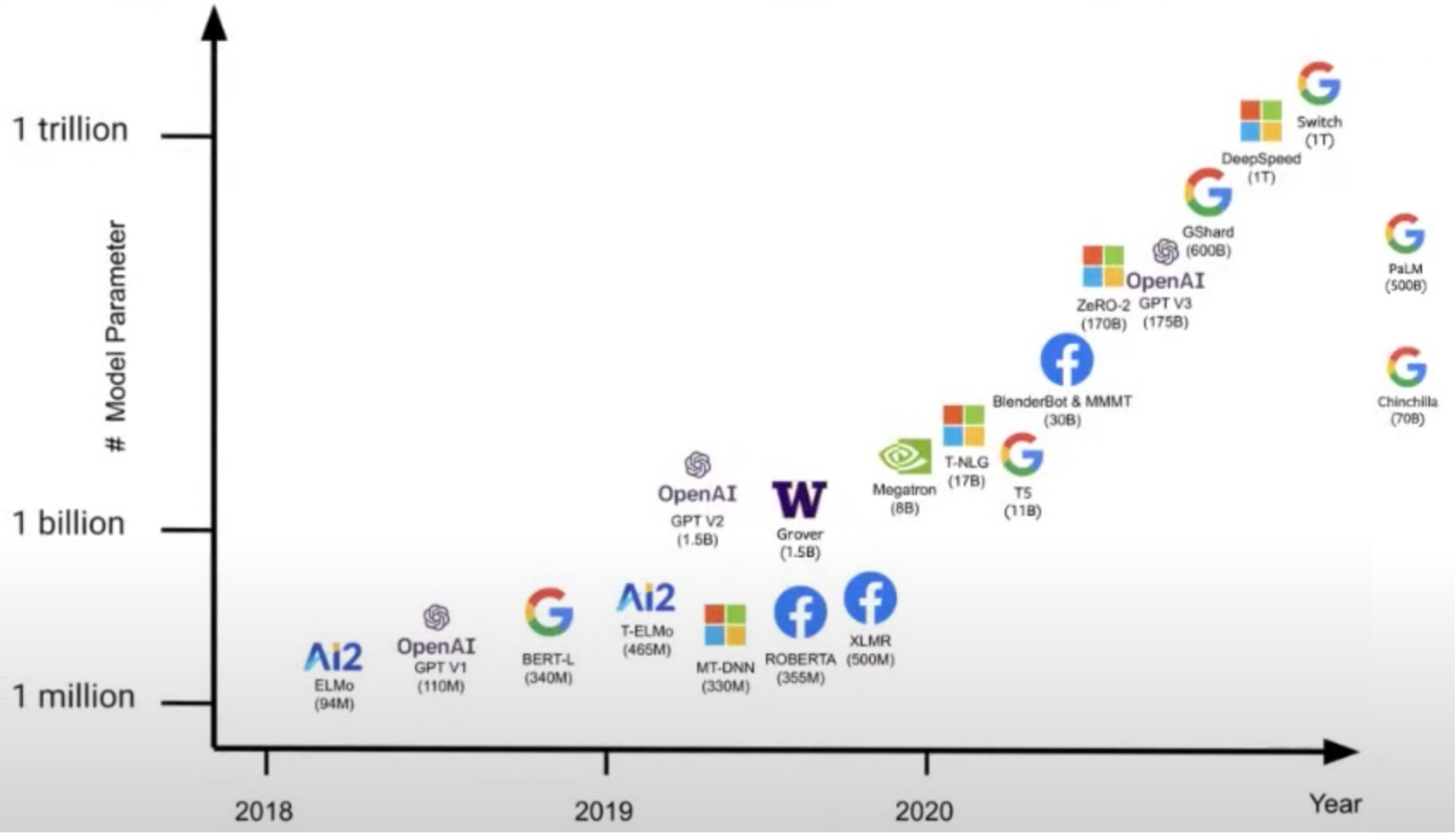
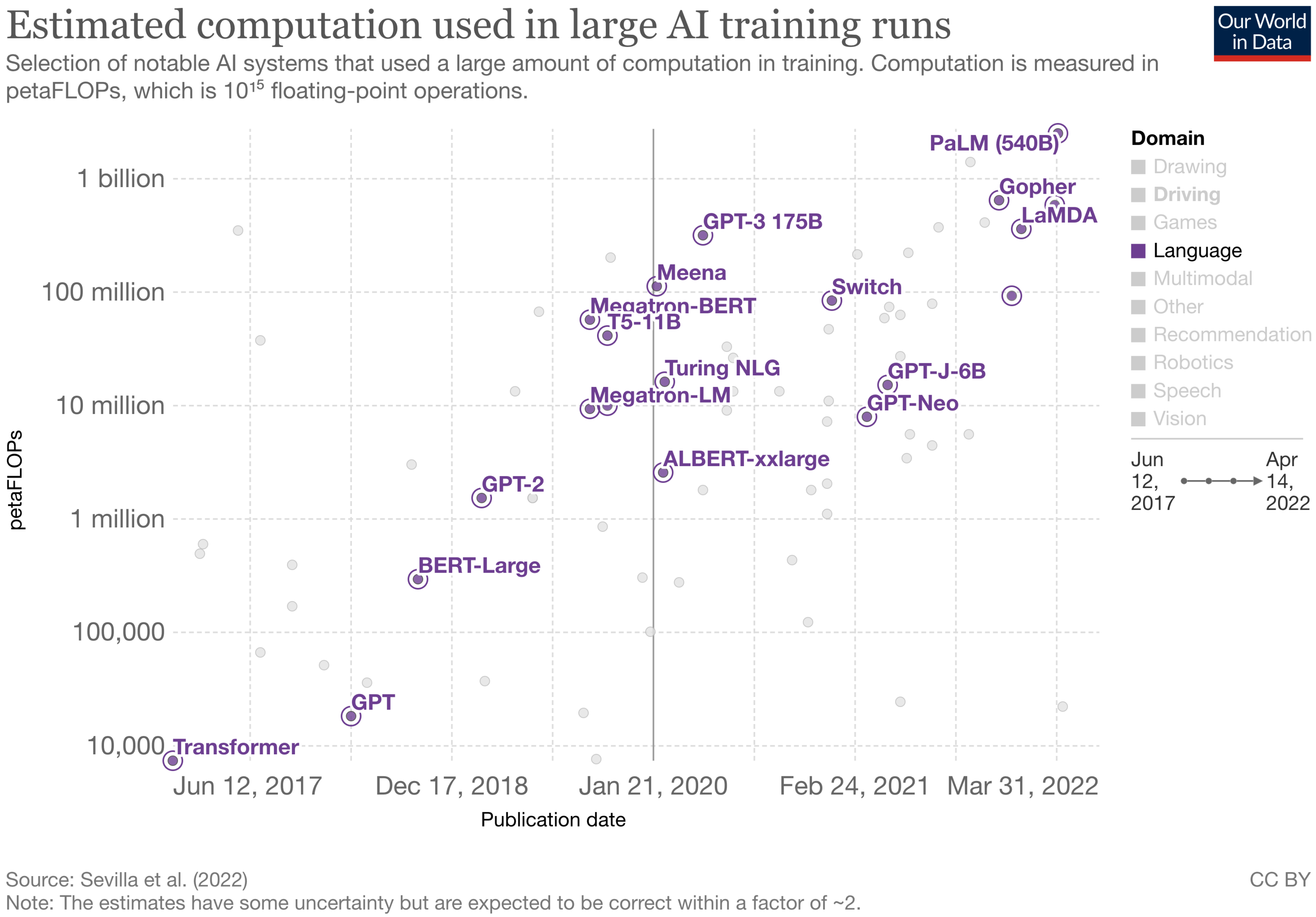


Image credit: [El Seminar - Luke Zettlemoyer - Large Language Models: Will they keep getting bigger?]

More GPUs: Computation Cost for Training LLMs



More Data: MassiveText Dataset

- Many huge datasets are collected
- MassiveText
- Diverse10-lingual textual dataset composed of web, Github, news, Wikipedia, Books, C4
- Disk size is **10.5 TB**
- Token count is around 5T tokens
- Document count is 2.32B with average 2k tokens per document

Source	Language	Token count (M)	Documents
Web	En	483,002	604,938,816
	Ru	103,954	93,004,882
	Es	95,762	126,893,286
	Zh	95,152	121,813,451
	Fr	59,450	76,612,205
	De	57,546	77,242,640
	Pt	44,561	62,524,362
	It	35,255	42,565,093
	Sw	2,246	1,971,234
	Ur	631	455,429
Books	En	3,423,740	20,472,632
News	En	236,918	397,852,713
Wikipedia	En	3,977	6,267,214
	De	2,155	3,307,818
	Fr	1,783	2,310,040
	Ru	1,411	2,767,039
	Es	1,270	2,885,013
	It	1,071	2,014,291
	Zh	927	1,654,772
	Pt	614	1,423,335
	Ur	61	344,811
	Sw	15	58,090
Github	-	374,952	142,881,832
Total	-	5,026,463	1,792,260,998

Table credit: https://vaclavkosar.com/ml/massivetext-dataset-pretraining-deepminds-gopher

Further Discussion on Large Language Models

- An overview of popular large language models
- A general recipe of training large language models
- **What can large language models do now?**
- Promising future directions

What Large Language Models Can Do Now?

Backbone model for nearly all NLP tasks now

- Small or medium language models: Pre-training & fine-tuning paradigm

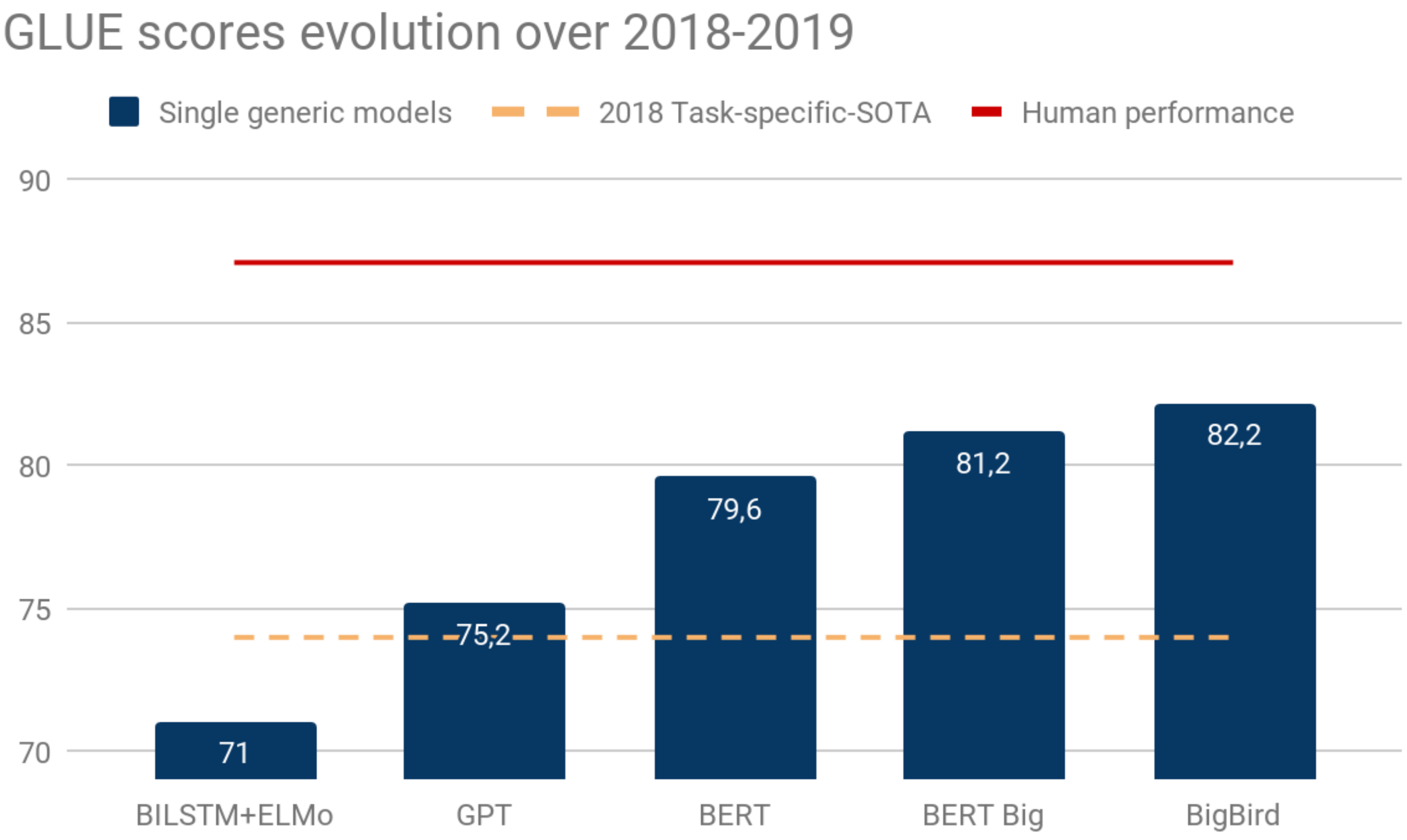
In-context learning without gradient updates

- Very large language models: Generalization with natural language instructions

Multimodal learning

- Language, vision, speech

Pre-training & Fine-tuning: Superior Performance



Source: https://medium.com/synapse-dev/understanding-bert-transformer-attention-isnt-all-you-need-5839ebd396db

A New Paradigm: In-context Learning or Prompting

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée        ←
plush girafe => girafe peluche      ←
cheese => ...........................  ← prompt
```

GPT-3 (Brown et al., 2020)

Generating Coherent Story

PROMPT

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

OUTPUT

They also were found to have perfectly coiffed hair, and wore what appeared to be Dior makeup.

"We were shocked to discover the unicorns," said anthropologist Daniel St. Maurice. "They were like nothing we had ever seen before. We had heard legends of the unicorns, but never thought they actually existed."

When the scientists first arrived in the valley, the unicorns were surprised and startled by the presence of humans, but were also excited. The unicorns welcomed the researchers and explained that they had been waiting for them for a very long time.

GPT-3 (Brown et al., 2020)

Source: https://www.buildgpt3.com/post/88/

Chain-of-thought Prompting

Explaining a Joke

Input: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

Model Output: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

PaLM (Chowdhery et al., 2022)

Chain-of-thought Prompting

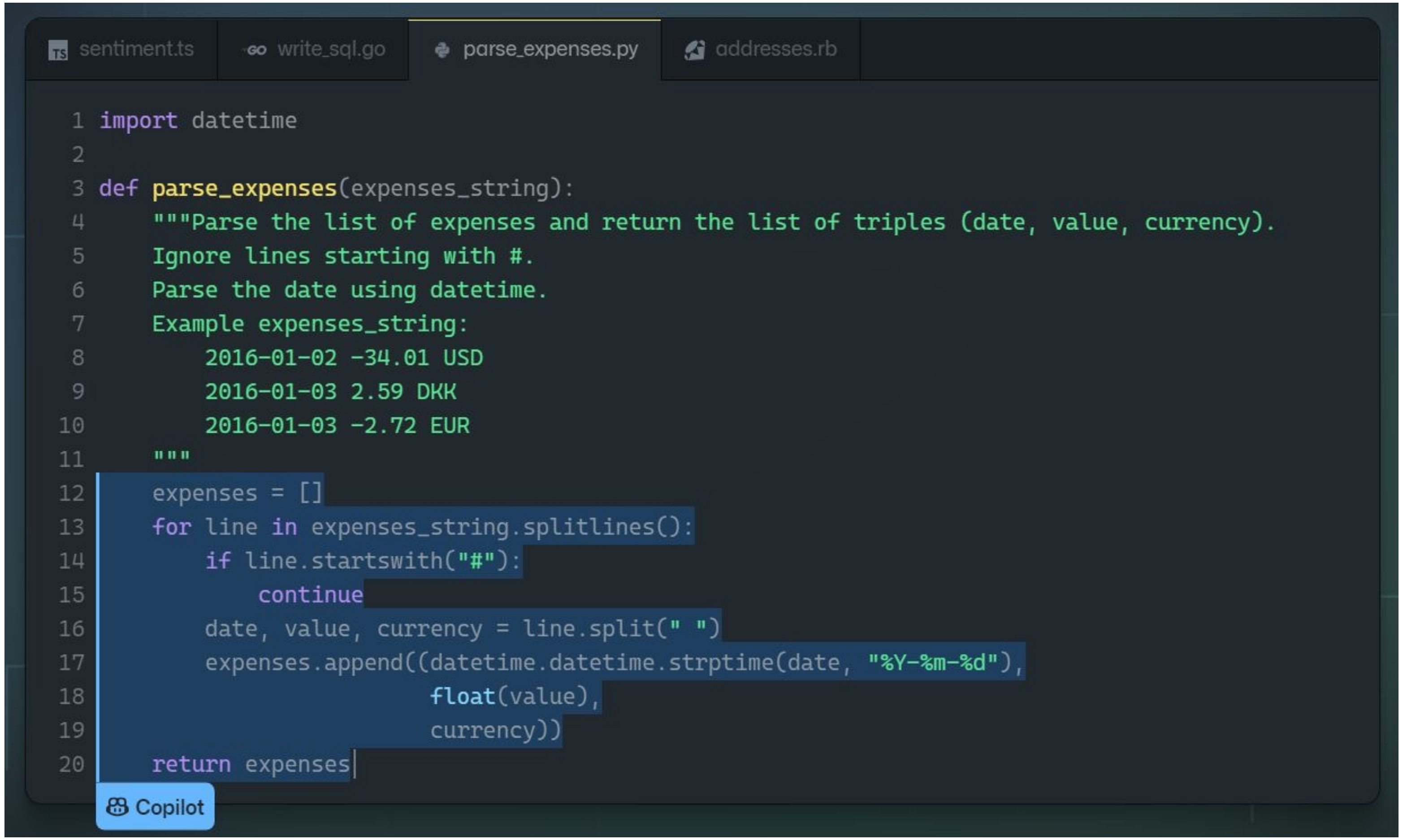
Logical Inference

Input: Shelley is from Virginia, but is visiting that city with that famous market where they throw the fish. Going home next Tuesday!
Question: Is it likely that Shelley will be near the Pacific Ocean this weekend?
Model Output: The city with the famous market where they throw the fish is Seattle, Washington. Seattle is on the Pacific Ocean. Shelley is visiting Seattle, so she will be near the Pacific Ocean this weekend. The answer is "yes", it is likely that Shelley will be near the Pacific Ocean this weekend.

PaLM (Chowdhery et al., 2022)

GitHub Copilot: Writing Useable Code

- Synthesize 28.8% functionally correct programs based on the docstrings



Codex (Chen et al., 2021)

Creating Images based on Text Captions

- A teddy bear on a skateboard in times square

DALLE-2 (Ramesh et al., 2022)

Creating Images based on Text Captions

- An astronaut riding a horse in a photorealistic style.

DALLE-2 (Ramesh et al., 2022)

Creating Images based on Text Captions

- A dramatic renaissance painting of Elon Musk buying Twitter

DALLE-2 (Ramesh et al., 2022)

Creating Images based on Text Captions

- Teddy bears working on new AI research on moon in the 1980s

DALLE-2 (Ramesh et al., 2022)

"In hindsight, the development of large-scale self-supervised learning approaches may well be viewed as the fundamental change, and the third era might be extended until 2017. The impact of pretrained self-supervised approaches has been revolutionary: it is now possible to train models on huge amounts of unlabeled human language material in such a way as to produce one large pretrained model that can be very easily adapted, via fine-tuning or prompting, to give strong results on all sorts of natural language understanding and generation tasks. As a result, progress and interest in NLP have exploded. There is a sense of optimism that we are starting to see the emergence of knowledge-imbued systems that have a degree of general intelligence."

Christopher Manning. "Human Language Understanding & Reasoning" in Daedalus, Spring 2022

Further Discussion on Large Language Models

- An overview of popular large language models
- A general recipe of training large language models
- What can large language models do now?
- **Promising future directions**

The Future of Large Language Models

Social Responsibility

- Benchmarking foundation models
- Documenting the ecosystem
- Economic impact on writing jobs
- Homogenization of outcomes
- Reducing model biases
- Enhance model fairness
- Reducing negative impacts on the environment (Green AI)

Technical Advances

- Diffusion models
- Retrieval-based models
- Efficient training
- Lightweight fine-tuning
- Decentralized training
- Understanding in-context learning
- Understanding the role od data
- Approximating optimal representations
- Structured state space sequence models

Applications

- Domain adaptation
- Differential privacy
- Writing assistance
- Prototyping social spaces
- Robotics (video, control)
- Audio (speech, music)
- Neuroscience
- Medicine (images, text)
- Bioinformatics
- Chemistry
- Law

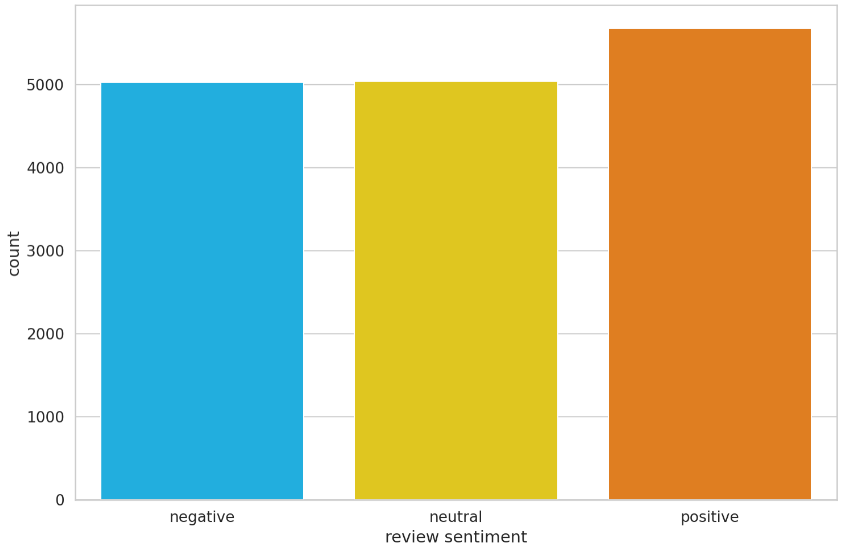
Partially adapted from [Percy Liang's talk on HAI Spring Conference 2022: Foundation Models]

Demo

I. Sentiment Analysis with BERT
II. Text Generation on GPT-3

Demo 1: Sentiment Analysis with BERT

- We will show how to fine-tune BERT for sentiment analysis
- Colab: [TDAI Summer School Tutorial]
  - Adapted from [Venelin Valkov's Tutorial]
- Data: Google Play app reviews dataset with five review scores
  - ~16K samples in total
- We normalize scores to three classes (negative, neutral, positive)



Hands on: Fine-tuning BERT on Sentiment Analysis

Key Points:

- Keep the main body of BERT unchanged
- Add a linear output layer on top of the model

Fine-tuning Procedure:

- Tokenize review text and map them to corresponding vocabulary ids
- Input the tokens into BERT and extract the last hidden state in [CLS]
- Pass the [CLS]'s hidden state in the linear output layer with a softmax to obtain class probabilities

Hands on: Fine-tuning BERT on Sentiment Analysis

Key Steps:

- Data Preprocessing
  1. Tokenization
  2. Truncate and pad
  3. Special tokens
  4. Attention masking
  5. Convert to ids

```
encoding = tokenizer.encode_plus(
  sample_txt,
  max_length=32,
  add_special_tokens=True, # Add '[CLS]' and '[SEP]'
  return_token_type_ids=False,
  pad_to_max_length=True,
  return_attention_mask=True,
  return_tensors='pt',  # Return PyTorch tensors
)
```

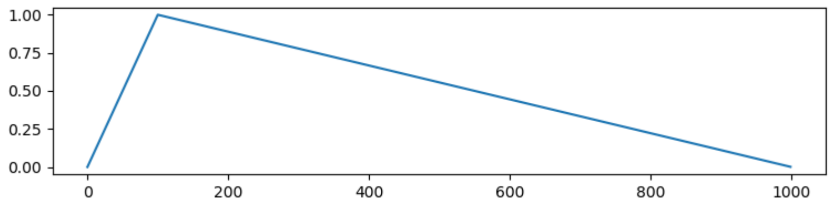
- Model Building
  - Load original BERT
  - Add a linear output layer

```
class SentimentClassifier(nn.Module):

  def __init__(self, n_classes):
    super(SentimentClassifier, self).__init__()
    self.bert = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME)
    self.drop = nn.Dropout(p=0.3)
    self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

  def forward(self, input_ids, attention_mask):
    outputs = self.bert(
      input_ids=input_ids,
      attention_mask=attention_mask
    )
    pooled_output = outputs.pooler_output
    output = self.drop(pooled_output)
    return self.out(output)
```

Hands on: Fine-tuning BERT on Sentiment Analysis

Key Steps:

- Training and Inference
- Fine-tuning hyper-parameters
  - AdamW optimizer
  - Fine-tune for 3 epochs
  - Learning rate: 2e-5 to 0
  - Linear schedule
    - Linearly increate to lr
    - Linearly decrease to 0




```python
def train_epoch(model, data_loader, loss_fn, optimizer, device, scheduler, n_examples):
  model = model.train()
  losses = []
  correct_predictions = 0

  for d in data_loader:
    input_ids = d["input_ids"].to(device)
    attention_mask = d["attention_mask"].to(device)
    targets = d["targets"].to(device)

    outputs = model(input_ids=input_ids, attention_mask=attention_mask)

    _, preds = torch.max(outputs, dim=1)
    loss = loss_fn(outputs, targets)
    correct_predictions += torch.sum(preds == targets)
    losses.append(loss.item())

    loss.backward()
    nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
    optimizer.step()
    scheduler.step()
    optimizer.zero_grad()

  return correct_predictions.double() / n_examples, np.mean(losses)
```

```python
def eval_model(model, data_loader, loss_fn, device, n_examples):
  model = model.eval()
  losses = []
  correct_predictions = 0

  with torch.no_grad():
    for d in data_loader:
      input_ids = d["input_ids"].to(device)
      attention_mask = d["attention_mask"].to(device)
      targets = d["targets"].to(device)

      outputs = model(input_ids=input_ids, attention_mask=attention_mask)
      _, preds = torch.max(outputs, dim=1)
      loss = loss_fn(outputs, targets)
      correct_predictions += torch.sum(preds == targets)
      losses.append(loss.item())

  return correct_predictions.double() / n_examples, np.mean(losses)
```

Demo II: Text Generation on GPT-3

- We will show how to generate coherent text with OpenAI API
  - https://beta.openai.com/playground

Goals:

- Learn important generation parameters
- Get a sense of how to craft prompts for GPT-3

Hands on: Text Generation on GPT-3

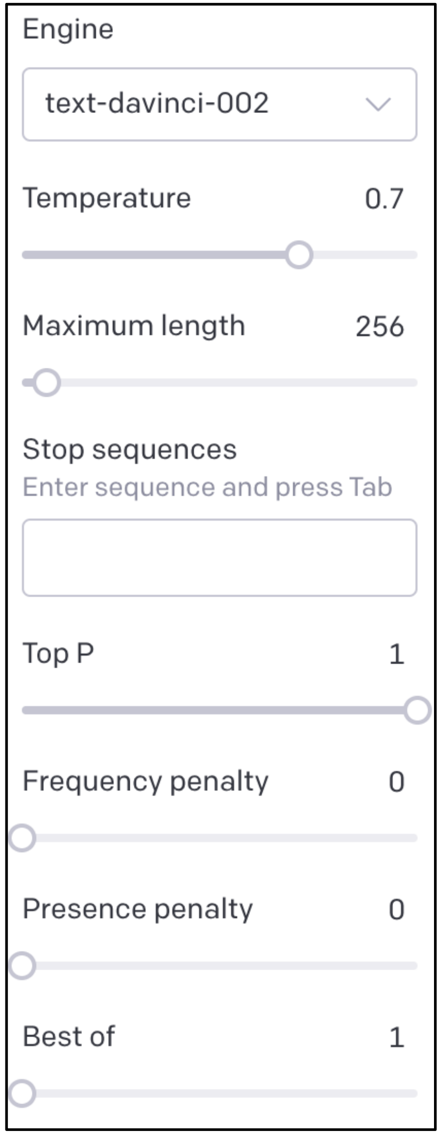
Important generation parameters

- Engine – different GPT-3 models
- Temperature – control generation randomness
- Maximum length

Example

Q: What is human life expectancy in the United States?

A: As of 2010, the life expectancy for a baby born in the US is 78 years.



Tutorial Structure

Part I (~75 mins):

- Tasks

- Deep Learning Models

Break (~15mins)

Part II: (~45 mins):

- Large Language Models

- Demo

QA (~15 mins)

Thank You
& QA