STATS / DATA SCI 315

Transformers

slides credit: Dr. Simon J. D. Prince
https://udlbook.github.io/udlbook/

Natural language processing (NLP)

- Translation
- Question answering
- Summarizing
- Generating new text
- Correcting spelling and grammar
- Finding entities
- Classifying bodies of text
- Changing style etc.

Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

Motivation

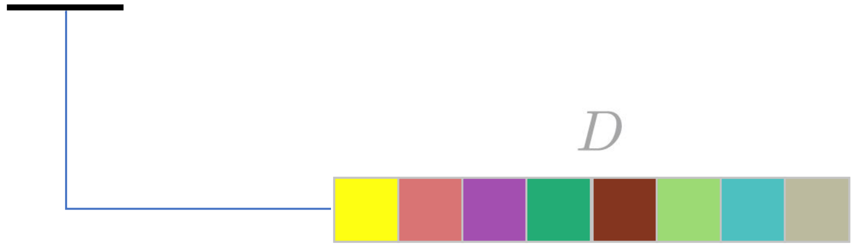
Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

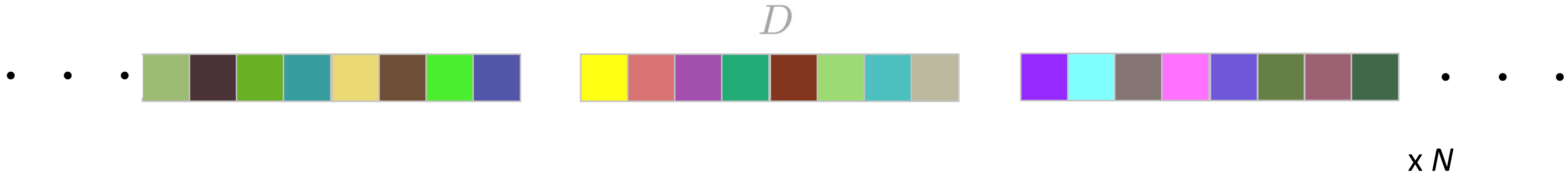


[Word embeddings](#) convert words into fixed dimensional vectors
[word2vec](#) is a popular family of word embeddings

Motivation

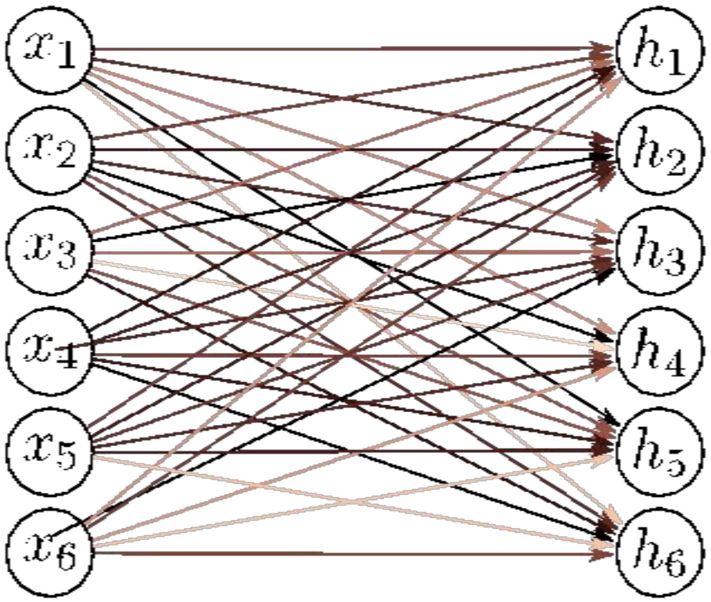
Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.



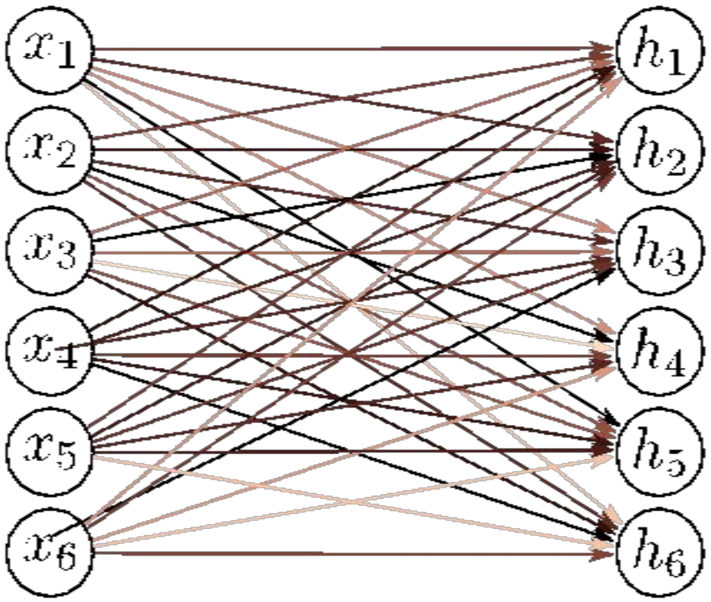
Standard fully-connected layer

$$\mathbf{h} = \mathbf{a}[\boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{x}]$$



Standard fully-connected layer

$$\mathbf{h} = \mathbf{a}[\boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{x}]$$



$\underbrace{\hphantom{xxxxxxxxxxxx}}_{\substack{\boldsymbol{\Phi} \text{ contains} \\ D^2 \text{ connections}}}$

Standard fully-connected layer

$$\mathbf{h} = \mathbf{a}[\boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{x}]$$

Problem:

- A very large number of parameters
- Can't cope with text of different lengths

Conclusion:

- We need a model where parameters don't increase with input length

Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

The word their must “attend to” the word restaurant.

Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

The word their must "attend to" the word restaurant.

Conclusions:

- There must be connections between the words.
- The strength of these connections will depend on the words themselves.

Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

Dot-product self attention

- Takes N inputs of size Dx1 and returns N inputs of size Dx1
- Computes N values (no ReLU)

$$\mathbf{v}_n = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \mathbf{x}_n$$

Dot-product self attention

- Takes N inputs of size Dx1 and returns N inputs of size Dx1
- Computes N values (no ReLU)

$$\mathbf{v}_n = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \mathbf{x}_n$$

- N outputs are weighted sums of these values

$$\mathbf{sa}[\mathbf{x}_n] = \sum_{m=1}^{N} a[\mathbf{x}_n, \mathbf{x}_m]\mathbf{v}_m$$

Dot-product self attention

- Takes N inputs of size Dx1 and returns N inputs of size Dx1
- Computes N values (no ReLU)

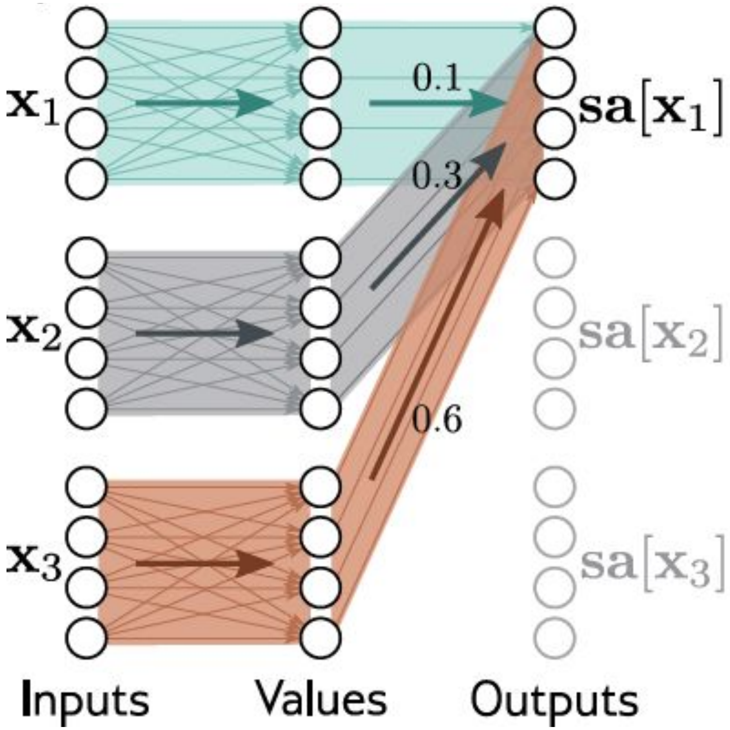
$$\mathbf{v}_n = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \mathbf{x}_n$$

- N outputs are weighted sums of these values

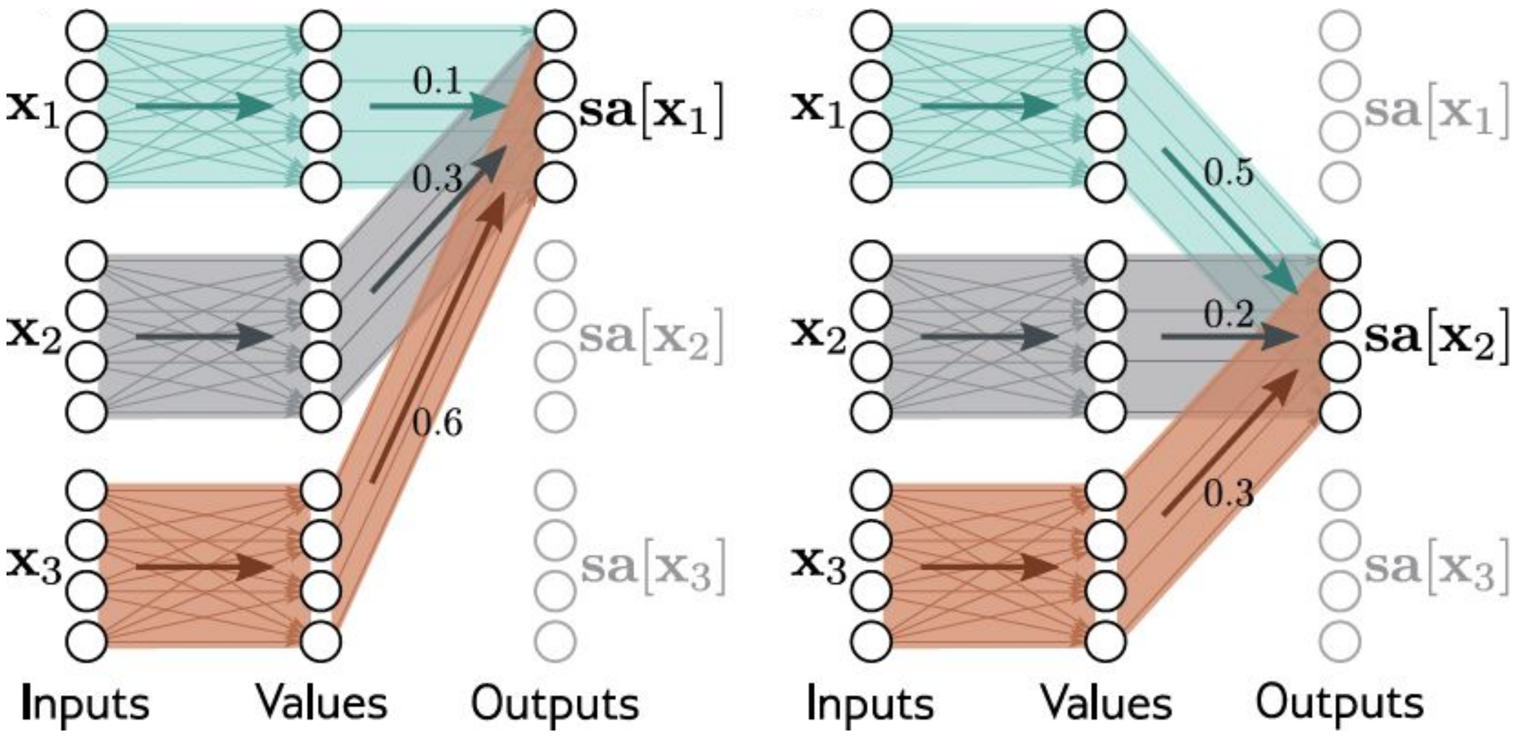
$$\mathbf{sa}[\mathbf{x}_n] = \sum_{m=1}^{N} a[\mathbf{x}_n, \mathbf{x}_m]\mathbf{v}_m$$

- Weights depend on the inputs themselves

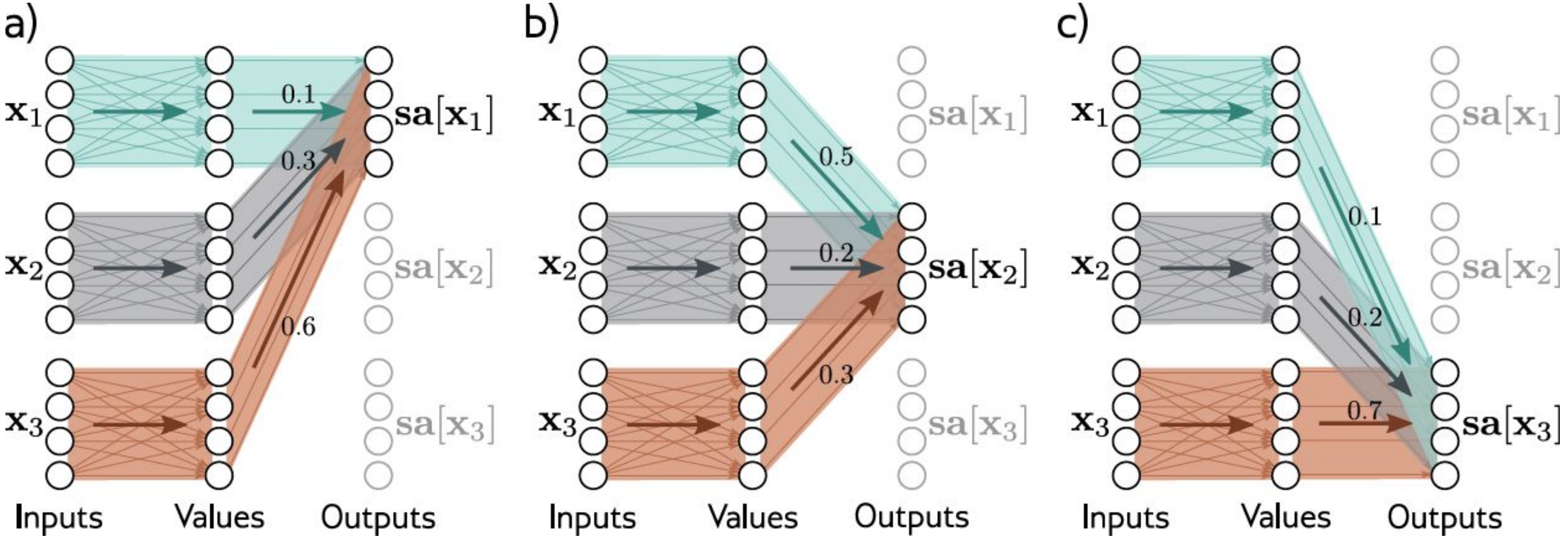
Attention as routing



Attention as routing



Attention as routing



Attention weights

- Compute N "queries" and N "keys" from input

$$\mathbf{q}_n = \boldsymbol{\beta}_q + \boldsymbol{\Omega}_q \mathbf{x}_n$$
$$\mathbf{k}_n = \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{x}_n,$$

- Calculate similarity and pass through softmax:

$$\begin{aligned} a[\mathbf{x}_n, \mathbf{x}_m] &= \mathrm{softmax}_m \left[\mathrm{sim}[\mathbf{k}_m \mathbf{q}_n]\right] \\ &= \frac{\exp\left[\mathrm{sim}[\mathbf{k}_m \mathbf{q}_n]\right]}{\sum_{m'=1}^{N} \exp\left[\mathrm{sim}[\mathbf{k}'_m \mathbf{q}_n]\right]}, \end{aligned}$$

Attention weights

- Compute N "queries" and N "keys" from input

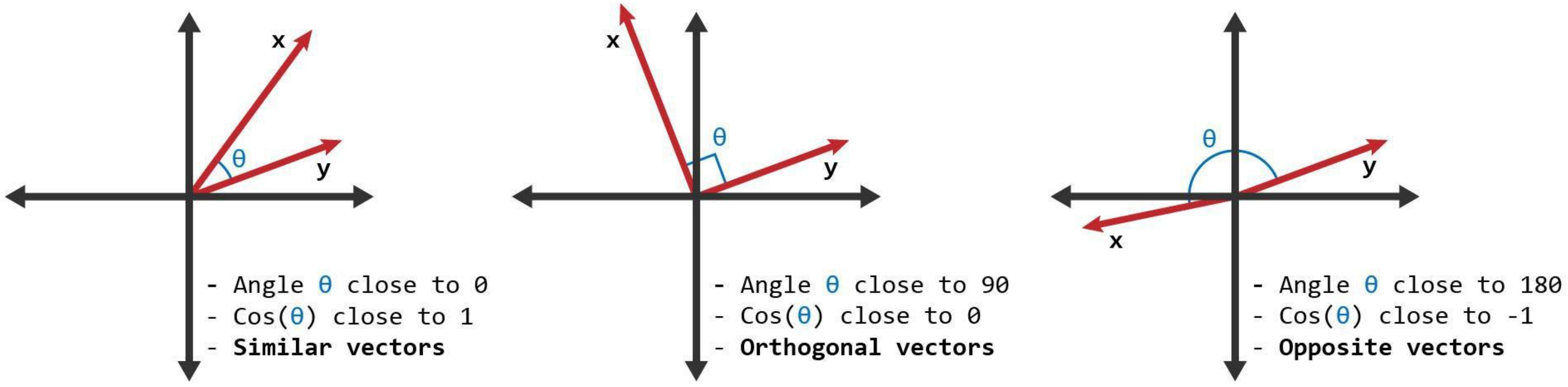
$$\mathbf{q}_n = \boldsymbol{\beta}_q + \boldsymbol{\Omega}_q \mathbf{x}_n$$
$$\mathbf{k}_n = \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{x}_n,$$

- Take dot products and pass through softmax:

$$\begin{aligned} a[\mathbf{x}_n, \mathbf{x}_m] &= \text{softmax}_m \left[\mathbf{k}_m^T \mathbf{q}_n \right] \\ &= \frac{\exp\left[\mathbf{k}_m^T \mathbf{q}_n\right]}{\sum_{m'=1}^{N} \exp\left[\mathbf{k}_{m'}^T \mathbf{q}_n\right]} \end{aligned}$$

Dot product = measure of similarity

$$\mathbf{x}^T\mathbf{y} = |\mathbf{x}| \cdot |\mathbf{y}| \cdot \boldsymbol{\theta}$$



Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

Conclusions:

- We need a model where parameters don't increase with input length

$$\phi = \{\boldsymbol{\beta}_v, \boldsymbol{\Omega}_v, \boldsymbol{\beta}_q, \boldsymbol{\Omega}_q, \boldsymbol{\beta}_k, \boldsymbol{\Omega}_k\}$$

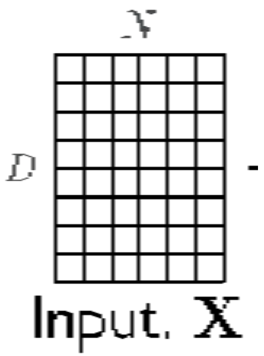
- There must be connections between the words.
- The strength of these connections will depend on the words themselves.

Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

Matrix form

- Store N input vectors in matrix X



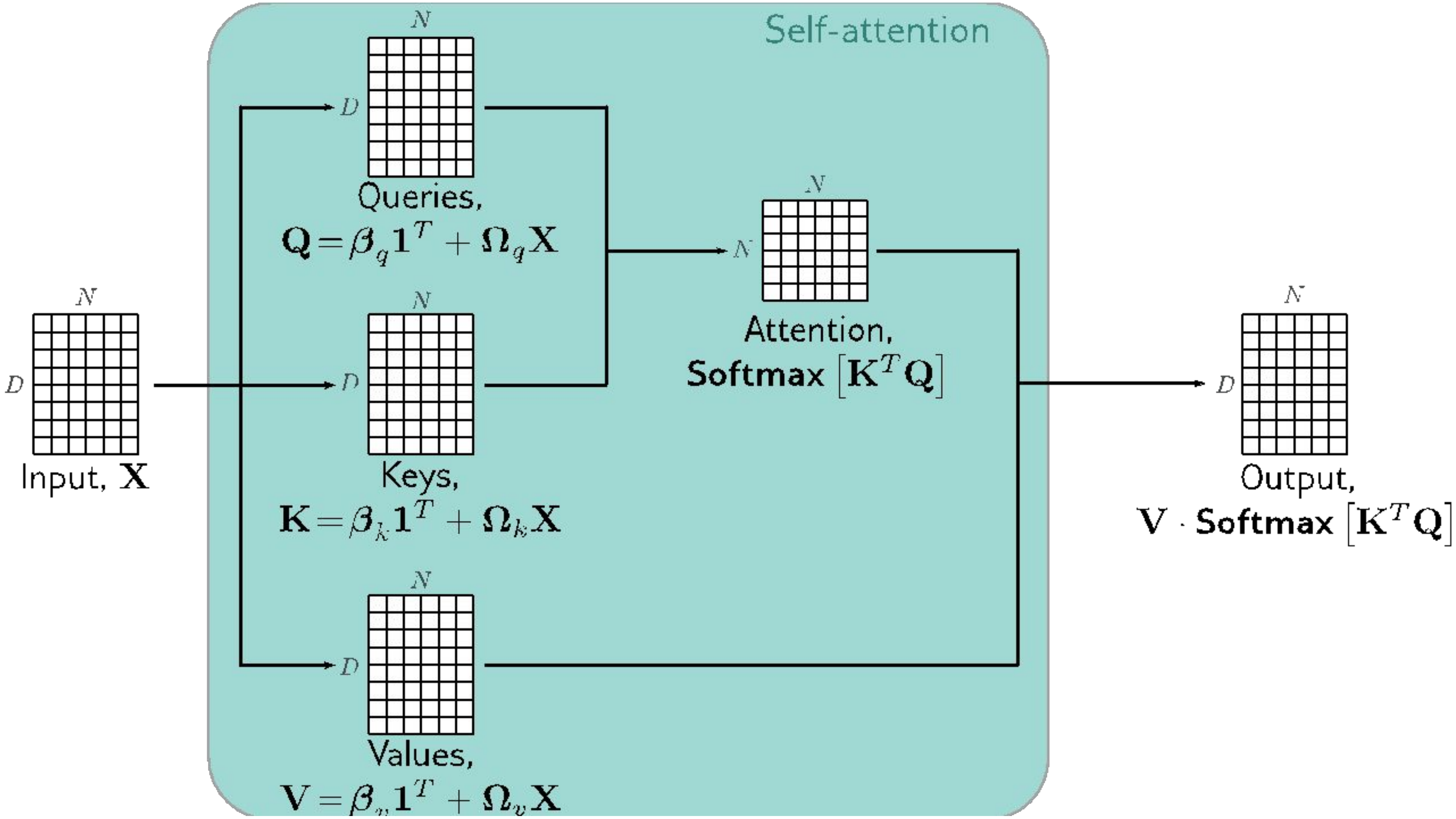
- Compute values, queries and keys (all dims are (D x 1) x (1 x N) + D x D x (D x N)):

$$\mathbf{V}[\mathbf{X}] = \boldsymbol{\beta}_v \mathbf{1}^\mathbf{T} + \boldsymbol{\Omega}_\mathbf{v}\mathbf{X}$$
$$\mathbf{Q}[\mathbf{X}] = \boldsymbol{\beta}_q \mathbf{1}^\mathbf{T} + \boldsymbol{\Omega}_\mathbf{q}\mathbf{X}$$
$$\mathbf{K}[\mathbf{X}] = \boldsymbol{\beta}_k \mathbf{1}^\mathbf{T} + \boldsymbol{\Omega}_\mathbf{k}\mathbf{X},$$

- Combine self-attentions (D x N x ColSoftmax(N x D x D x N))

$$\mathbf{Sa}[\mathbf{X}] = \mathbf{V}[\mathbf{X}] \cdot \mathbf{Softmax}\Big[\mathbf{K}[\mathbf{X}]^T\mathbf{Q}[\mathbf{X}]\Big]$$

Matrix form



Self-attention
N
D
Queries,
$\mathbf{Q} = \boldsymbol{\beta}_q \mathbf{1}^T + \boldsymbol{\Omega}_q \mathbf{X}$
Keys,
$\mathbf{K} = \boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \mathbf{X}$
Values,
$\mathbf{V} = \boldsymbol{\beta}_v \mathbf{1}^T + \boldsymbol{\Omega}_v \mathbf{X}$
Attention,
Softmax $\left[\mathbf{K}^T \mathbf{Q}\right]$
Input, $\mathbf{X}$
Output,
$\mathbf{V} \cdot$ **Softmax** $\left[\mathbf{K}^T \mathbf{Q}\right]$

Positional encoding

Self-attention is equivariant to permuting word order

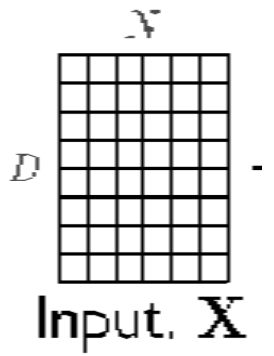
But word order is important in language:

The man ate the fish

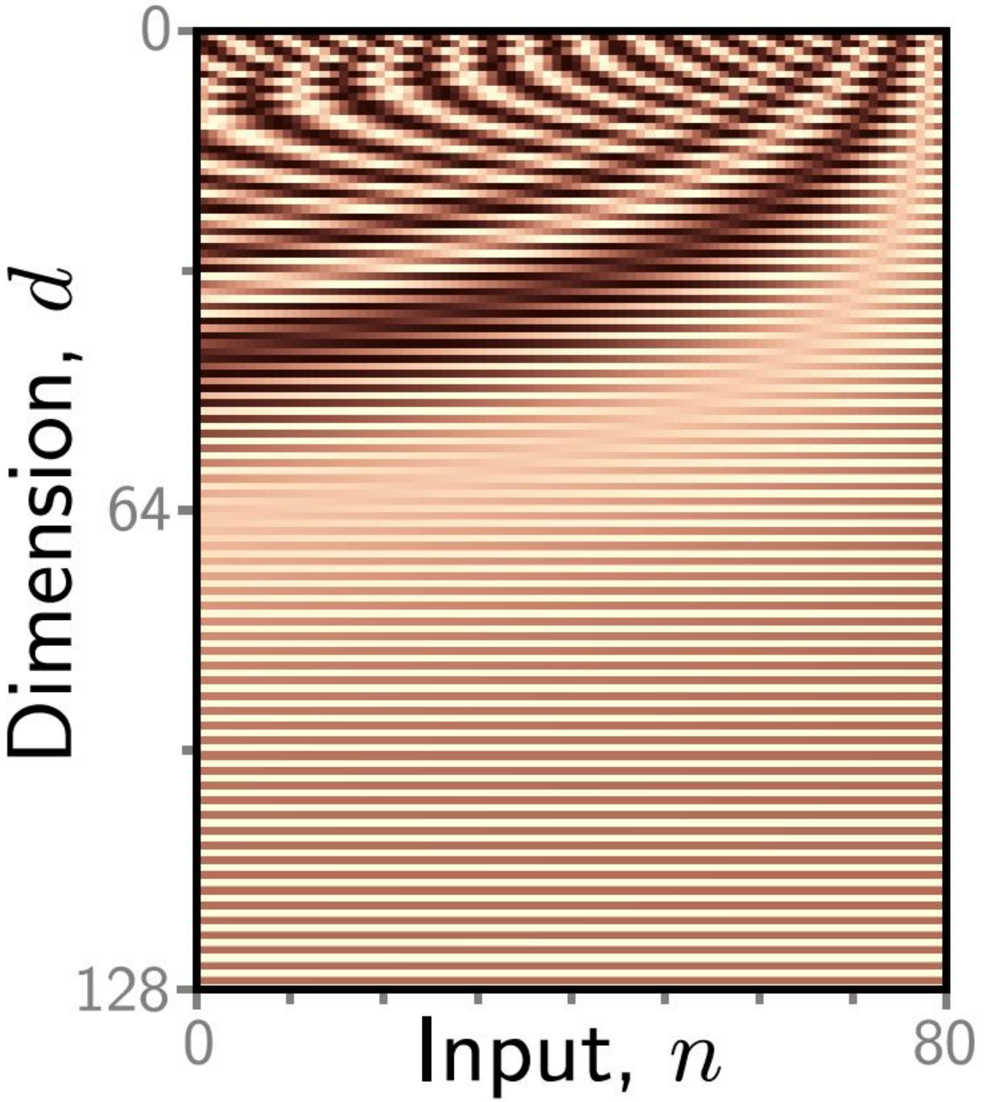
vs.

The fish ate the man

Positional encoding



$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

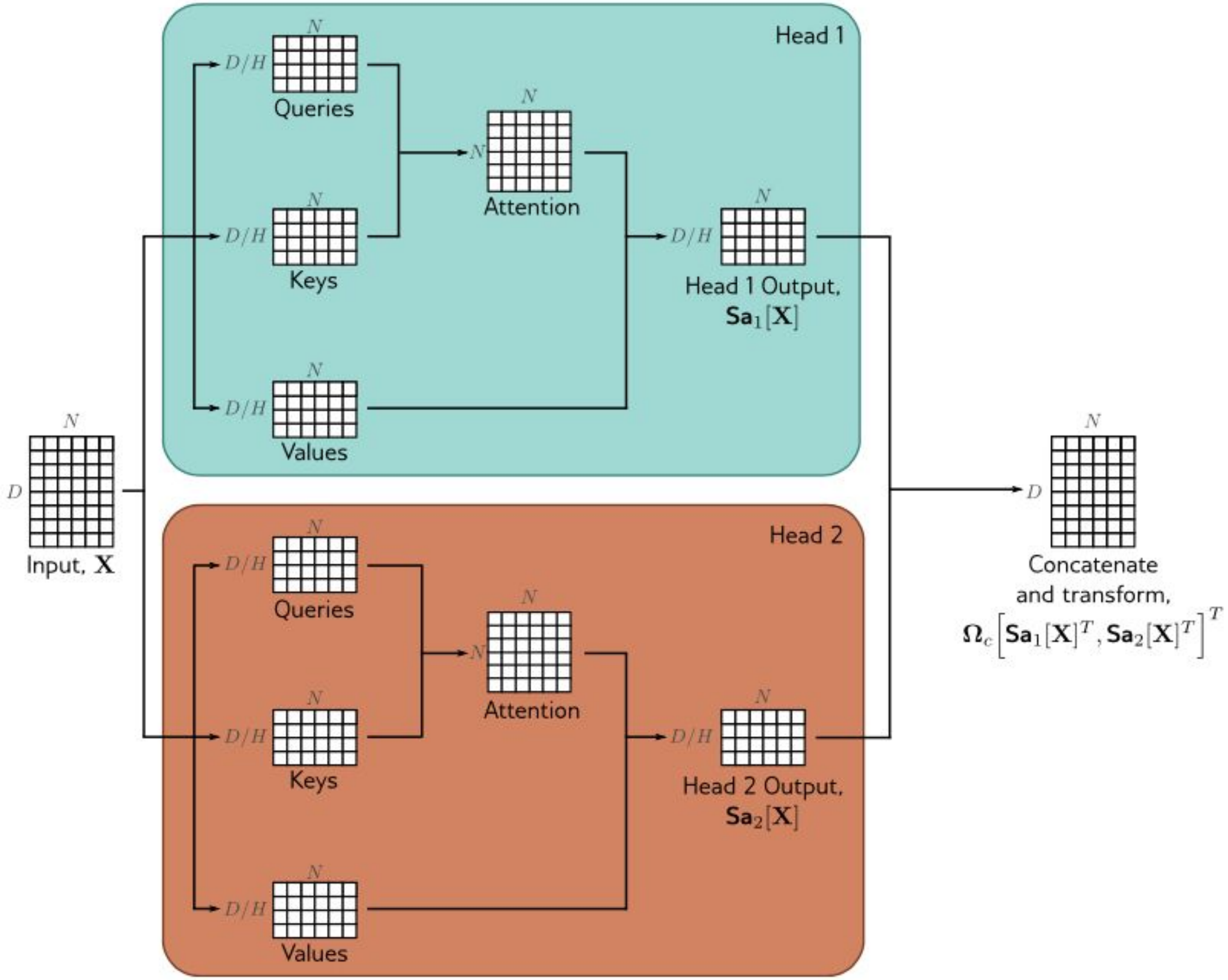


Positional encoding

$$\mathbf{Sa}[\mathbf{X}] = \mathbf{V} \cdot \mathbf{Softmax}[\mathbf{K}^T \mathbf{Q}]$$

$$\mathbf{Sa}[\mathbf{X}] = (\mathbf{V} + \mathbf{\Pi}) \cdot \mathbf{Softmax}[(\mathbf{K} + \mathbf{\Pi})^T (\mathbf{Q} + \mathbf{\Pi})]$$

Multi-head self-attention



Why do we learn attention weights?

- Without learning attention weights, a self-attention layer is closely related to Nadaraya-Watson kernel regression
  - https://d2l.ai/chapter_attention-mechanisms-and-transformers/attention-pooling.html
  - https://en.wikipedia.org/wiki/Kernel_regression
- Self-attention layer is more powerful because it can learn the attention weights as opposed to using a fixed hard-coded function (like the kernel in NW kernel regression)

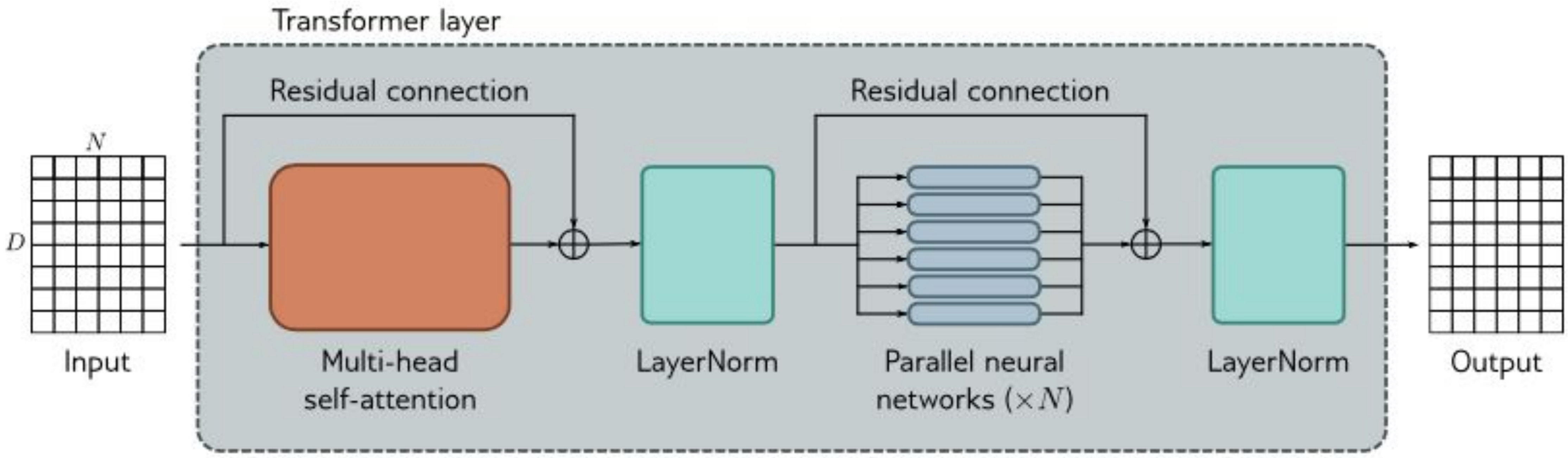
Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Encoders, decoders, and encoder-decoders
- Transformers for vision

Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

The transformer



In Keras: https://keras.io/api/layers/normalization_layers/layer_normalization/

Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

Tokenizer

Goal: Tokenizer chooses input "units"

- Inevitably, some words (e.g., names) will not be in the vocabulary.
- It's not clear how to handle punctuation
- The vocabulary would need different tokens for versions of the same word with different suffixes (e.g., walk, walks, walked, walking) and there is no way to clarify that these variations are related

Solution: Sub-word tokenization
One particular algorithm: Byte pair encoding

a)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

a)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

b)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	se	a	t	o	h	l	u	b	d	w	c	s	f	i	m	n	p	r
33	15	13	12	11	8	6	6	4	3	3	3	2	2	1	1	1	1	1	1

a)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

b)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	se	a	t	o	h	l	u	b	d	w	c	s	f	i	m	n	p	r
33	15	13	12	11	8	6	6	4	3	3	3	2	2	1	1	1	1	1	1

c)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	se	a	e_	t	o	h	l	u	b	d	e	w	c	s	f	i	m	n	p	r
21	13	12	12	11	8	6	6	4	3	3	3	3	2	2	1	1	1	1	1	1

a)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

b)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	se	a	t	o	h	l	u	b	d	w	c	s	f	i	m	n	p	r
33	15	13	12	11	8	6	6	4	3	3	3	2	2	1	1	1	1	1	1

c)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	se	a	e_	t	o	h	l	u	b	d	e	w	c	s	f	i	m	n	p	r
21	13	12	12	11	8	6	6	4	3	3	3	3	2	2	1	1	1	1	1	1

⋮ ⋮

d)

see_	sea_	e	b	l	w	a	could_	hat_	he_	o	t	t_	the_	to_	u	a_	d	f	m	n	p	s	sailor_	to
7	6	4	3	3	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1

a)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

b)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	se	a	t	o	h	l	u	b	d	w	c	s	f	i	m	n	p	r
33	15	13	12	11	8	6	6	4	3	3	3	2	2	1	1	1	1	1	1

c)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	se	a	e_	t	o	h	l	u	b	d	e	w	c	s	f	i	m	n	p	r
21	13	12	12	11	8	6	6	4	3	3	3	3	2	2	1	1	1	1	1	1

⋮ ⋮

d)

see_	sea_	e	b	l	w	a	could_	hat_	he_	o	t	t_	the_	to_	u	a_	d	f	m	n	p	s	sailor_	to
7	6	4	3	3	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1

⋮ ⋮ ⋮

e)

see_	sea_	could_	he_	the_	a_	all_	blue_	bottom_	but_	deep_	of_	sailor_	that_	to_	was_	went_	what_
7	6	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1

a)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

b)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	e	se	a	t	o	h	l	u	b	d	w	c	s	f	i	m	n	p	r
33	15	13	12	11	8	6	6	4	3	3	3	2	2	1	1	1	1	1	1

c)

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

_	se	a	e_	t	o	h	l	u	b	d	e	w	c	s	f	i	m	n	p	r
21	13	12	12	11	8	6	6	4	3	3	3	3	2	2	1	1	1	1	1	1

⋮ ⋮

d)

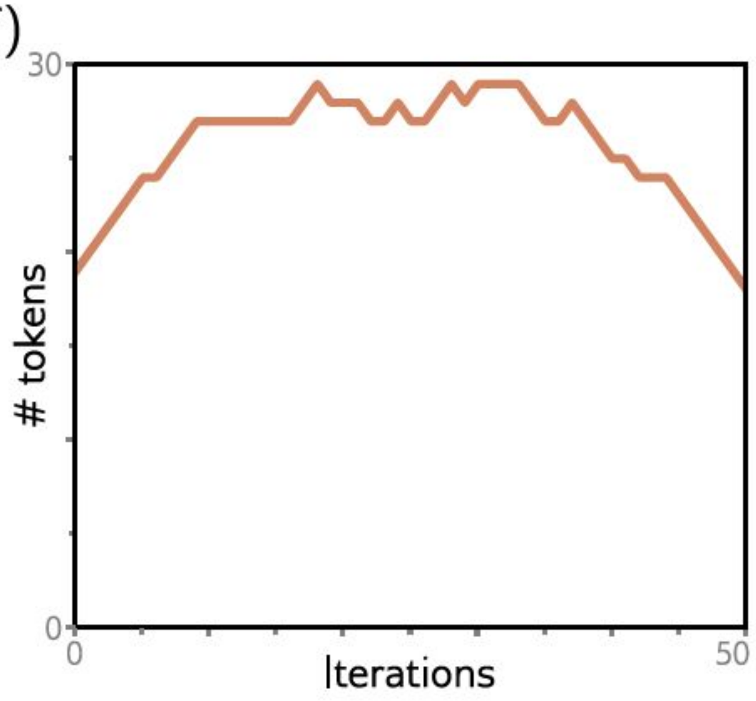
see_	sea_	e	b	l	w	a	could_	hat_	he_	o	t	t_	the_	to_	u	a_	d	f	m	n	p	s	sailor_	to
7	6	4	3	3	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1

⋮ ⋮ ⋮

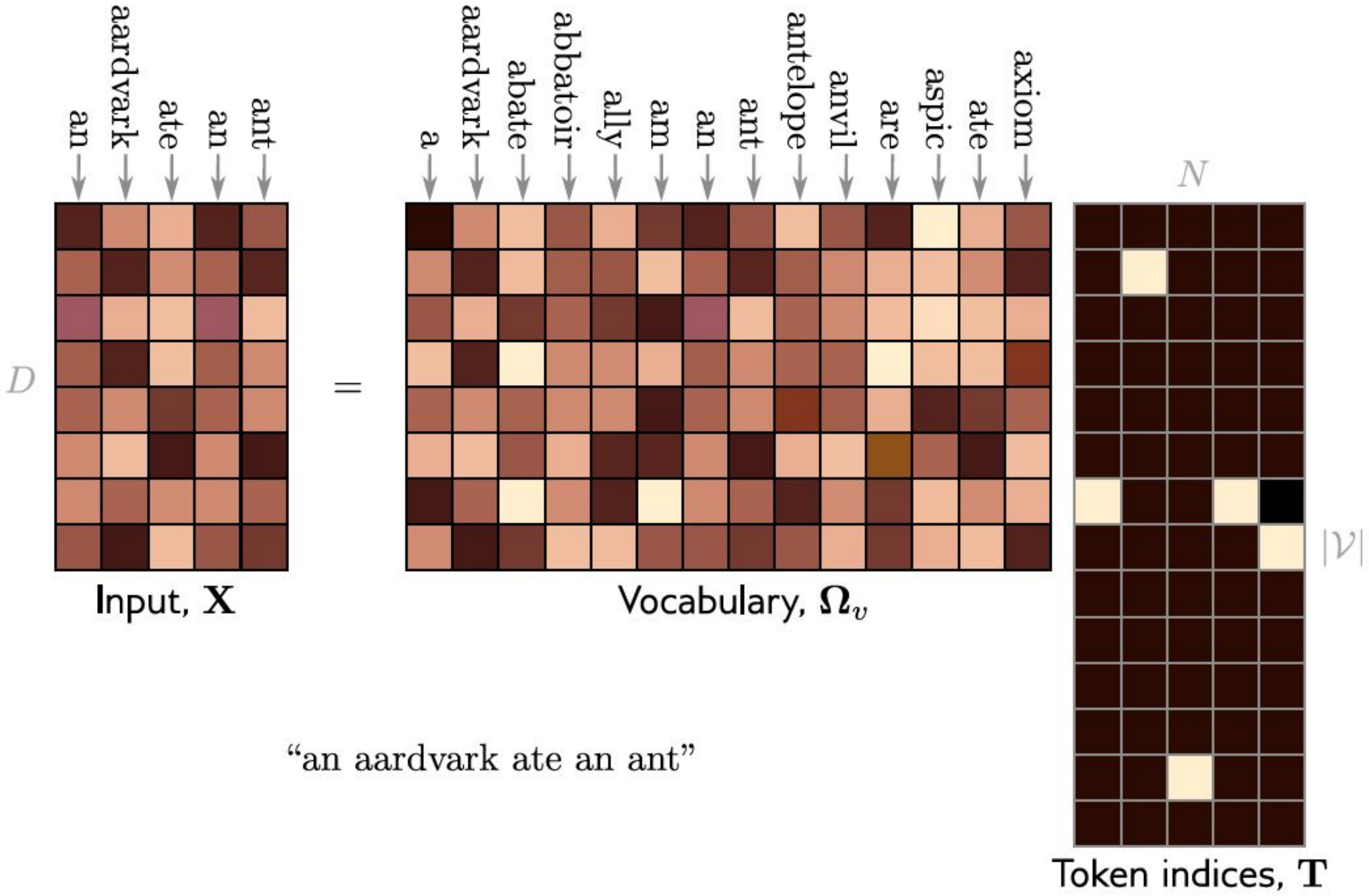
e)

see_	sea_	could_	he_	the_	a_	all_	blue_	bottom_	but_	deep_	of_	sailor_	that_	to_	was_	went_	what_
7	6	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1

f)



Learning vocabulary



”One hot encoding”

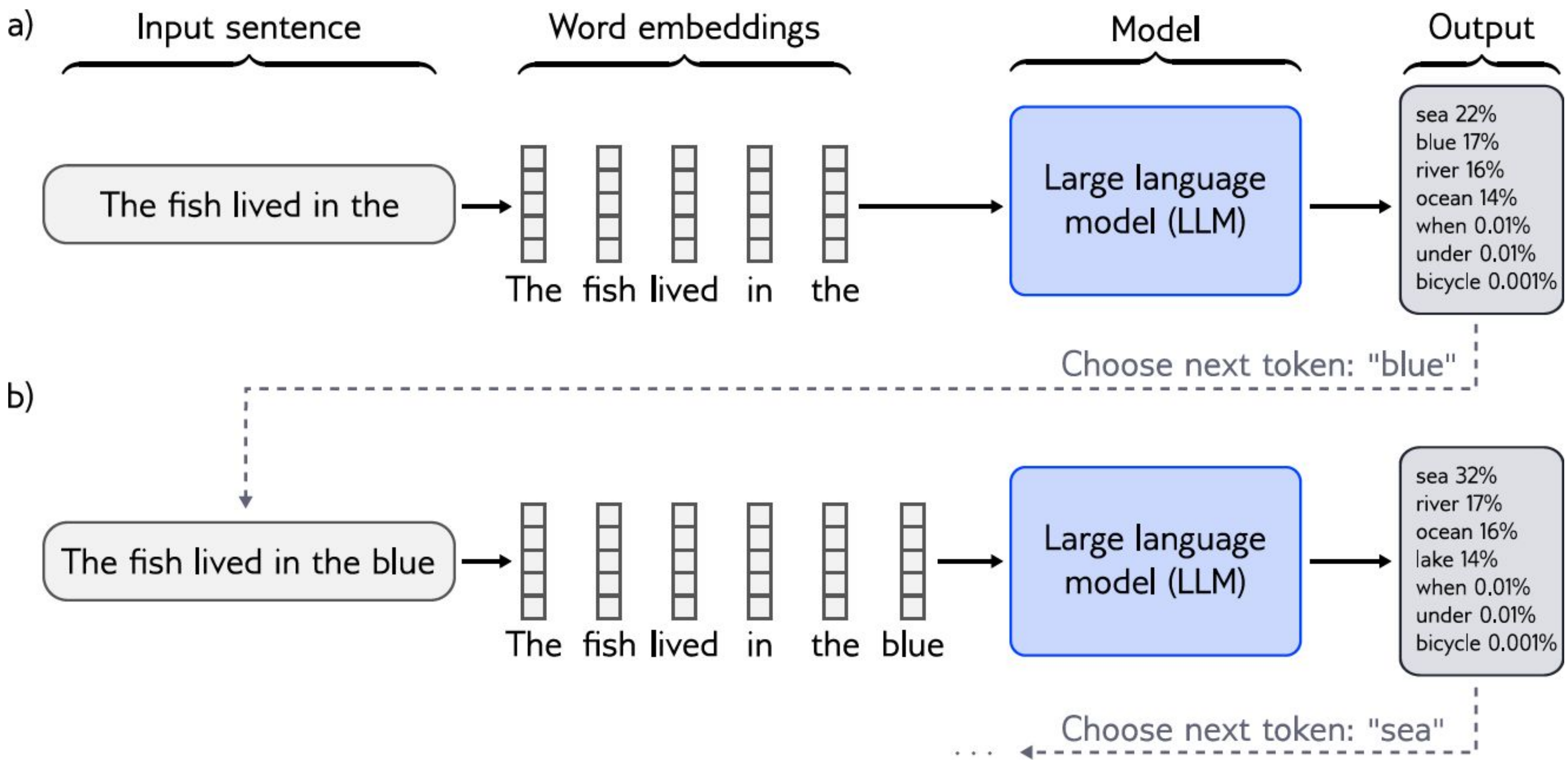
Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

Three types of transformer layer

- Encoder (BERT)
- Decoder (GPT3) ← will only look at this today
- Encoder-decoder (Translation)

Decoder model



Decoder model: GPT3

- One job: predict the next word in a sequence
- More formally builds an autoregressive probability model

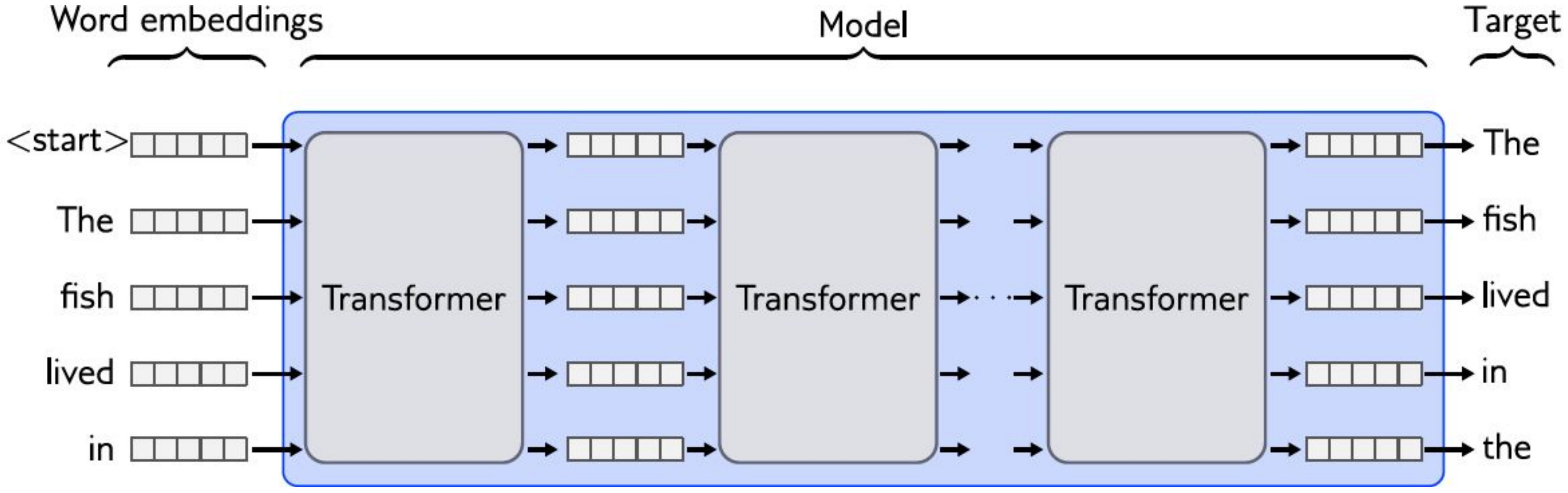
$$Pr(t_1, t_2, \ldots t_N) = Pr(t_1) \prod_{n=2}^{N} Pr(t_n | t_1 \ldots t_{n-1})$$

Decoder model: GPT3

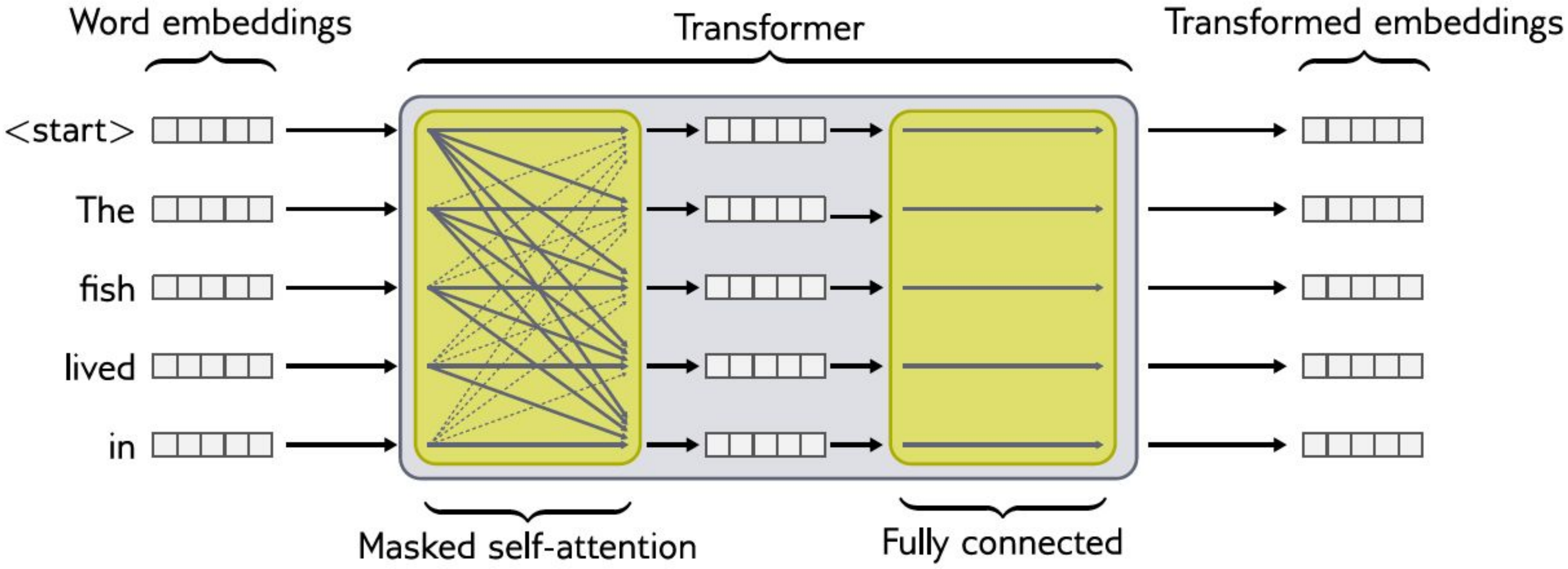
- Builds autoregressive probability model
- E.g. “It takes great courage to let yourself appear weak”

$$
\begin{aligned}
&Pr(\text{It takes great personal courage to let yourself appear weak}) = \\
&Pr(\text{It}) \times Pr(\text{takes}|\text{It}) \times Pr(\text{great}|\text{It takes}) \times Pr(\text{courage}|\text{It takes great})\times \\
&Pr(\text{to}|\text{It takes great courage}) \times Pr(\text{let}|\text{It takes great courage to})\times \\
&Pr(\text{yourself}|\text{It takes great courage to let})\times \\
&Pr(\text{appear}|\text{It takes great courage to let yourself})\times \\
&Pr(\text{weak}|\text{It takes great courage to let yourself appear}).
\end{aligned}
$$

Predicting all next words simultaneously



Masked self-attention



Transformers

- Motivation
- Dot-product self-attention
- Matrix form
- The transformer
- NLP pipeline
- Decoders
- Large Language models

GPT3 (Brown et al. 2020)

- Sequence lengths are 2048 tokens long
- Batch size is 3.2 million tokens.
- 96 transformer layers (some of which implement a sparse version of attention), each of which processes a word embedding of size 12288.
- 96 heads in the self-attention layers and the value, query, and key dimension is 128.
- 300 billion tokens
- 175 billion parameters

try some out at
https://umgpt.umich.edu/

What does it learn?

- Syntax

  “Tomorrow, let’s <VERB>...”

- General knowledge:

  “The train pulled into the …”

Text completion

Understanding Deep Learning is a new textbook from MIT Press by Simon Prince that's designed to offer an accessible, broad introduction to the field. Deep learning is a branch of machine learning that is concerned with algorithms that learn from data that is unstructured or unlabeled. The book is divided into four sections:

- Introduction to deep learning
- Deep learning architecture
- Deep learning algorithms
- Applications of deep learning

The first section offers an introduction to deep learning, including its history and origins. The second section covers deep learning architecture, discussing various types of neural networks and their applications. The third section dives into deep learning algorithms, including supervised and unsupervised learning, reinforcement learning, and more. The fourth section applies deep learning to various domains, such as computer vision, natural language processing, and robotics.

Few shot learning:

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

ChatGPT

- GPT3.5 fine-tuned with human annotations
- Trained to predict the next word + be “helpful, honest, harmless”

Prompt:
ELI5: What’s the cause of the "anxiety lump" in our chest during stressful or disheartening experiences?

Prompt:
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

ChatGPT

- GPT3.5 fine-tuned with human annotations
- Trained to predict the next word + be “helpful, honest, harmless”

Prompt:
ELI5: What’s the cause of the "anxiety lump" in our chest during stressful or disheartening experiences?

Labeler demonstration
The änxiety lumpïn your throat is caused by muscular tension keeping your glottis dilated to maximize airflow. The c̈lenched chestör ḧeartachef̈eeling is caused by the vagus nerve which tells the organs to pump blood faster, stop digesting, and produce adrenaline and cortisol.

Prompt:
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

Labeler demonstration
Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.