BOSTON UNIVERSITY

Deep Learning for Data Science
DS 542

https://dl4ds.github.io/fa2025/

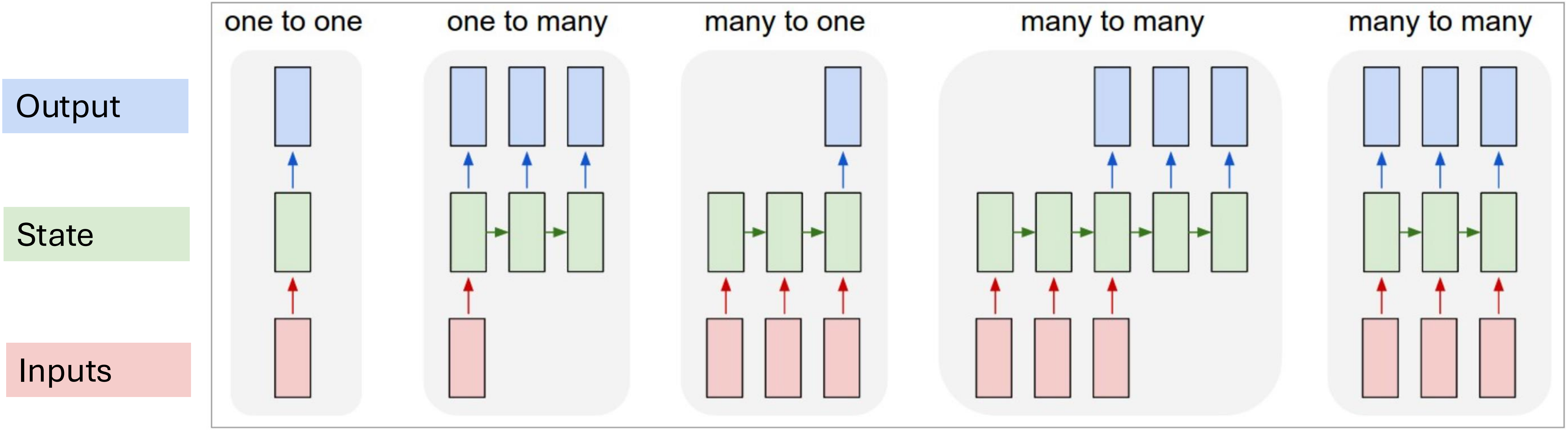
Attention and Transformers

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

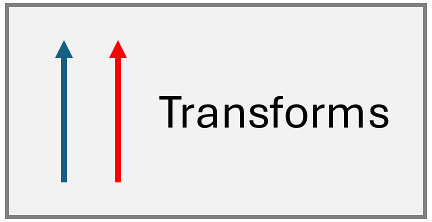
Plan for Today

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Different RNN configurations

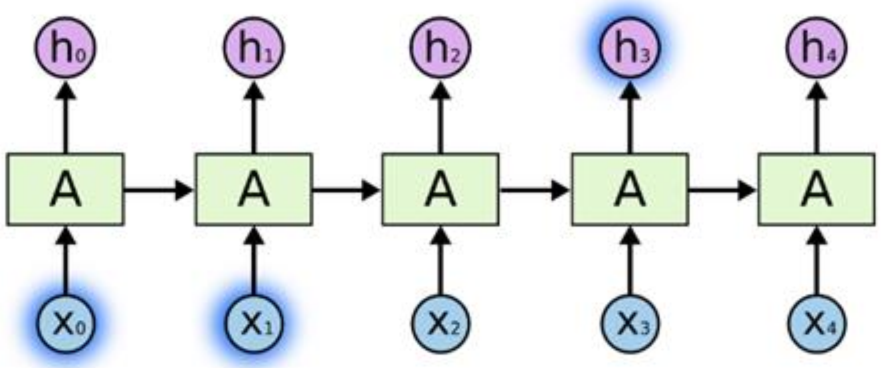


(a) (b) (c) (d) (e)

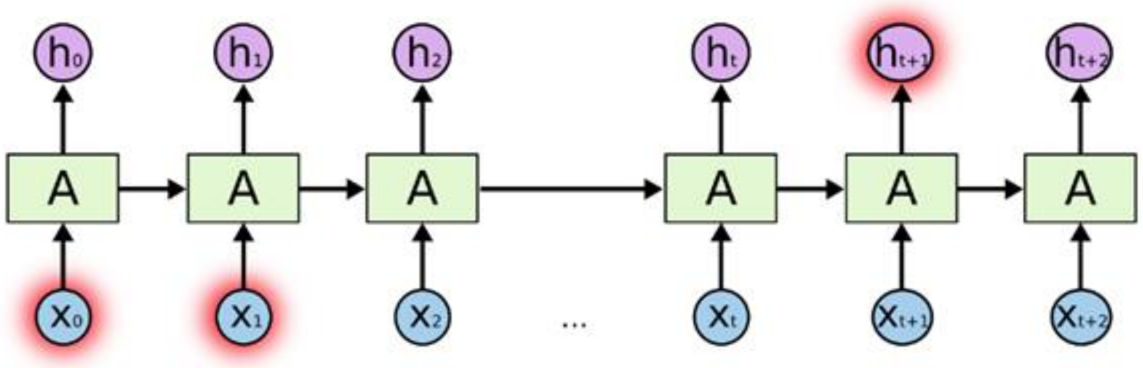


(a) Regular Feed Forward Network
(b) E.g. image captioning – input 1 image, outputs sequence of words
(c) E.g. sentiment analysis from string of words or characters
(d) E.g. machine translation such as English to French
(e) Synced sequence input and output, e.g. video frame-by-frame action classification or text generation

Problem of vanishing gradients



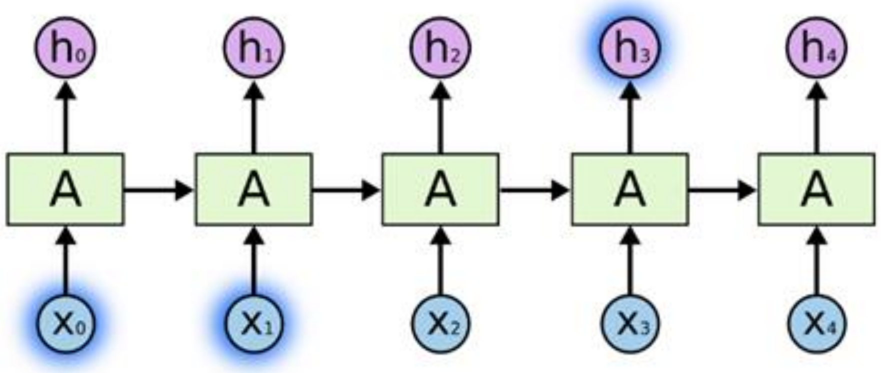
Tokens from earlier in the sequence can influence the current output



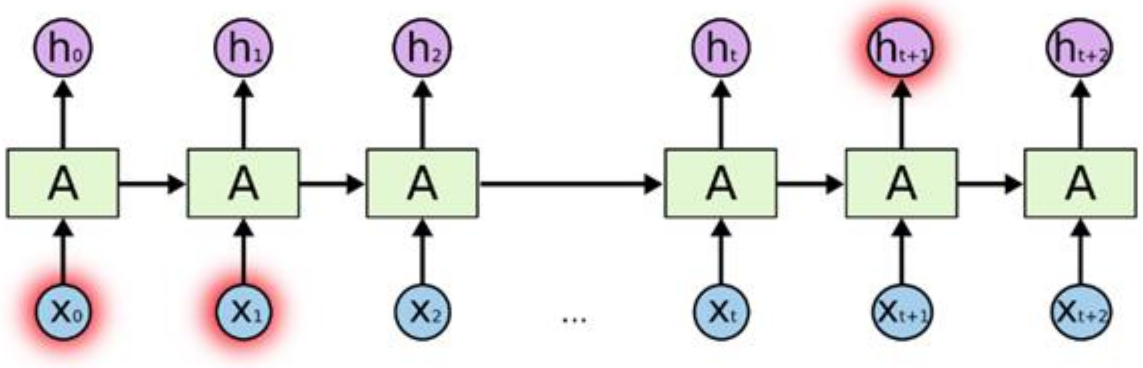
But for plain RNNs, the influence can reduce rapidly the further the sequence difference

Understanding LSTM Networks, C. Colah Blog Post

Why not exploding gradients?



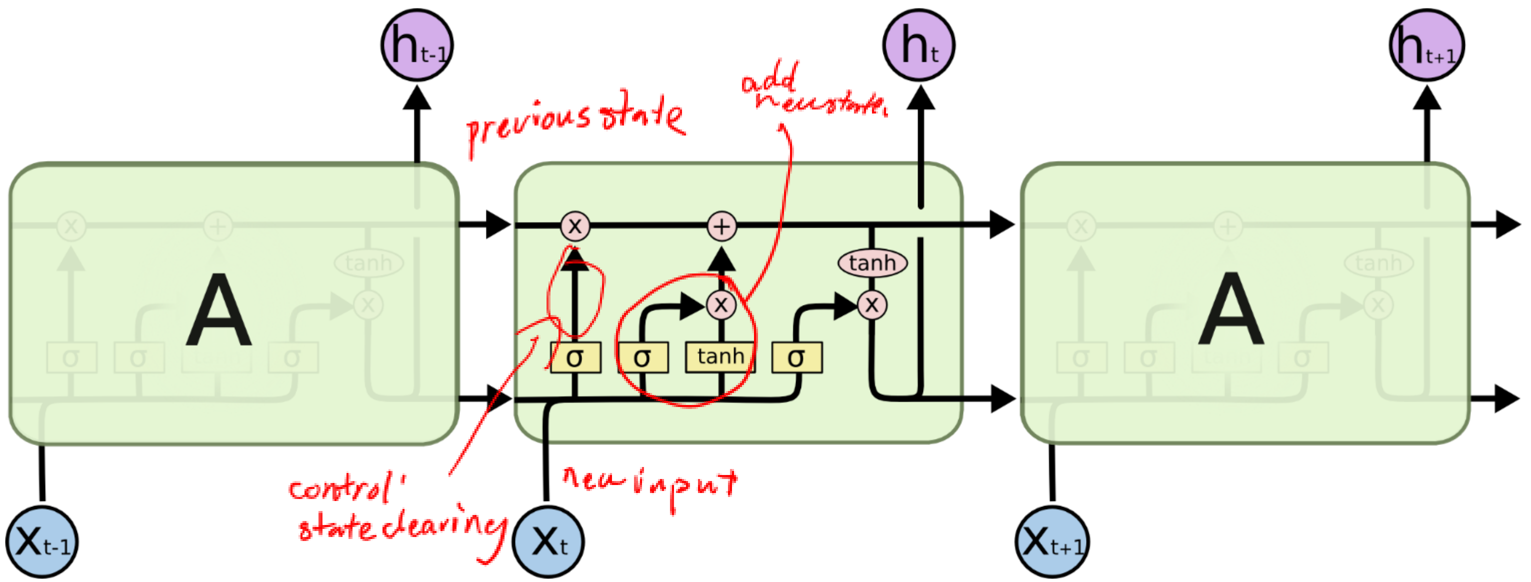
Tokens from earlier in the sequence can influence the current output



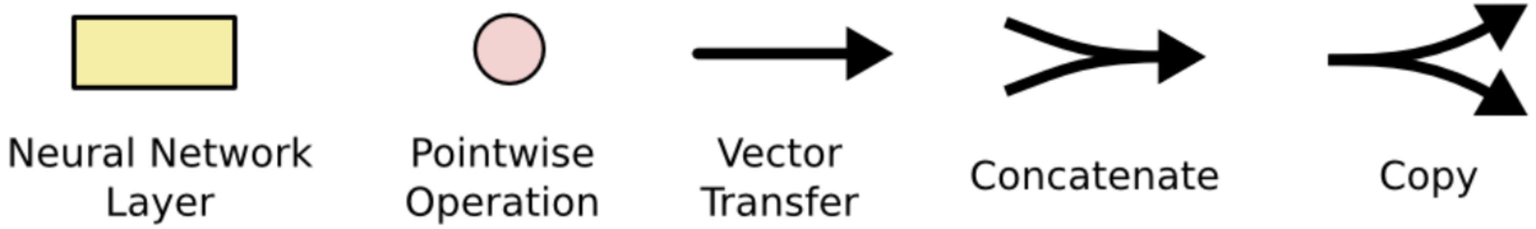
But for plain RNNs, the influence can reduce rapidly the further the sequence difference

Understanding LSTM Networks, C. Colah Blog Post

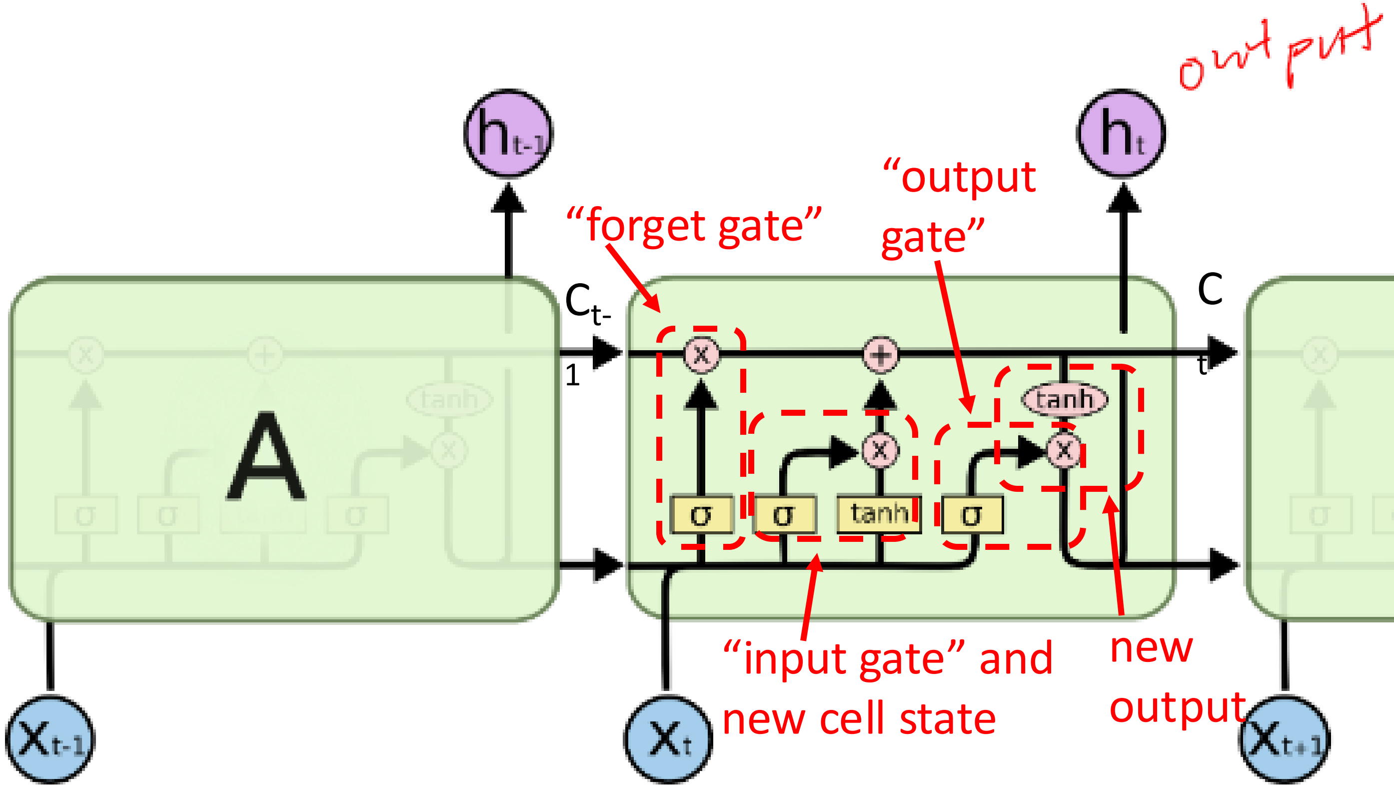
Long Short Term Memory (LSTM)

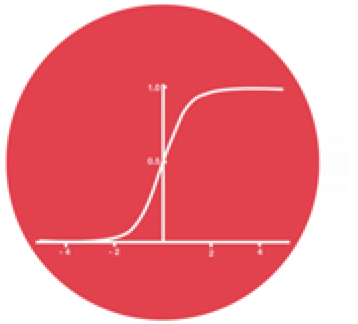


The repeating module in an LSTM contains four interacting layers.

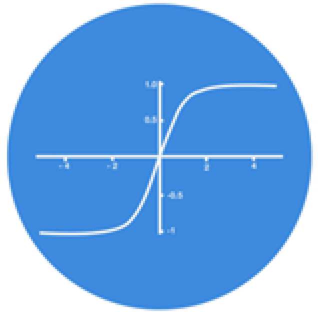


Long Short Term Memory (LSTM)

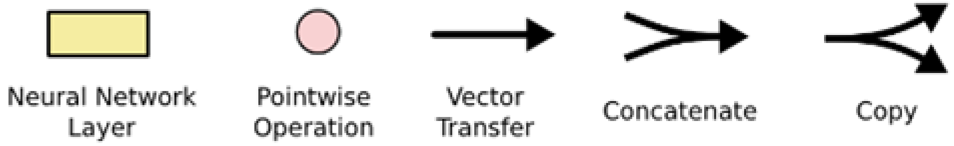




σ – Sigmoid, $\mathbb{R} \rightarrow [0,1]$



tanh(), $\mathbb{R} \rightarrow [-1,1]$



<u>Neural Network Layer</u>:

$$out_t = activation(W \cdot [h_{t-1}, x_t] + b)$$

Understanding LSTM Networks, C. Colah Blog Post
Illustrated Guide to LSTM's and GRU's, M. Phi Blog Post

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

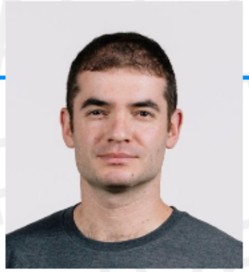
A Brief History of Transformers

2000

Yoshua Bengio*

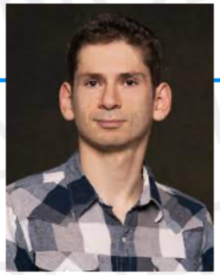
2014

Ilya Sutskever*



2014

Dzmitry Bahdanau*

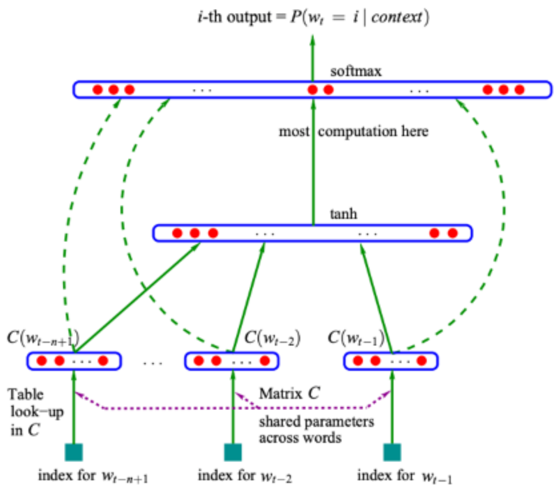


2017

A Team at Google

fixed window input

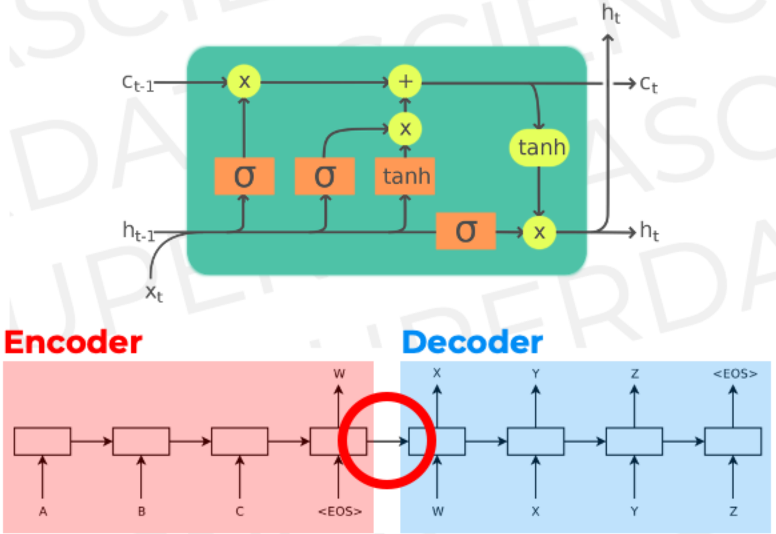
A Neural Probabilistic Language Model



Use LSTMs

input many, output many

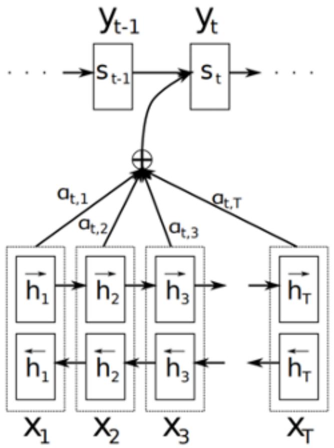
Seq-to-Seq Learning with Neural Networks



Add Attention

LSTM + attention

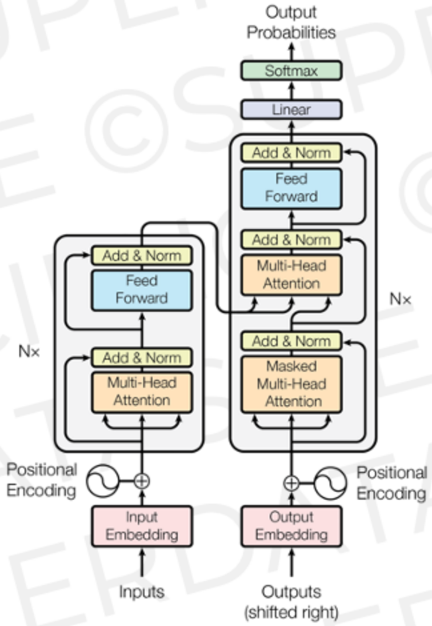
Neural Machine Translation by Jointly Learning to Align and Translate



Remove LSTMs

attention only

Attention is all you need



*And others; Chronological analysis inspired by Andrej Karpathy's lecture, youtube.com/watch?v=XfpMkf4rD6E

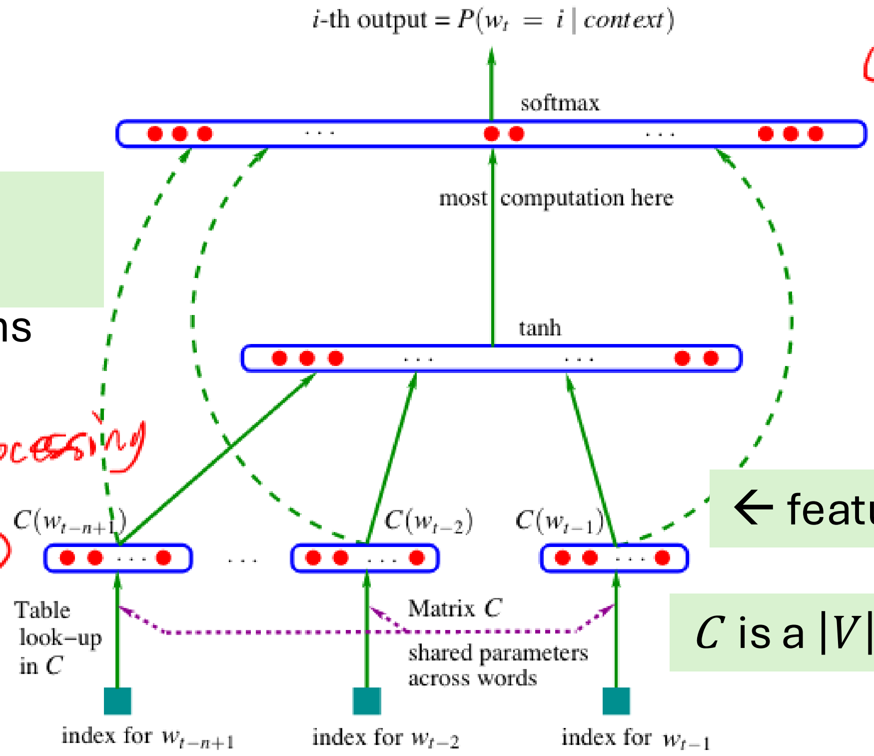
A Neural Probabilistic Language Model

Bengio et al, 2000 and 2003

Pick which word

Optional direct connections →

repeat processing per input



← feature vectors, $C(w_t)$

C is a $|V| \times m$ matrix

Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

$w_t \in V$ words in the vocabulary

- Build a probabilistic language model from NNs
- Feed forward network with shared parameters, C, that create embeddings
- Predicts the probability of a word at time t, based on the context of the last n words
- Can use shallow feed forward or recurrent neural networks

Limited to context length of n

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A Neural Probabilistic Language Model,” *Journal of Machine Learning Research*, vol. 3, pp. 1137-- 1155, Feb. 2003.

Generating Sequences With Recurrent Neural Networks

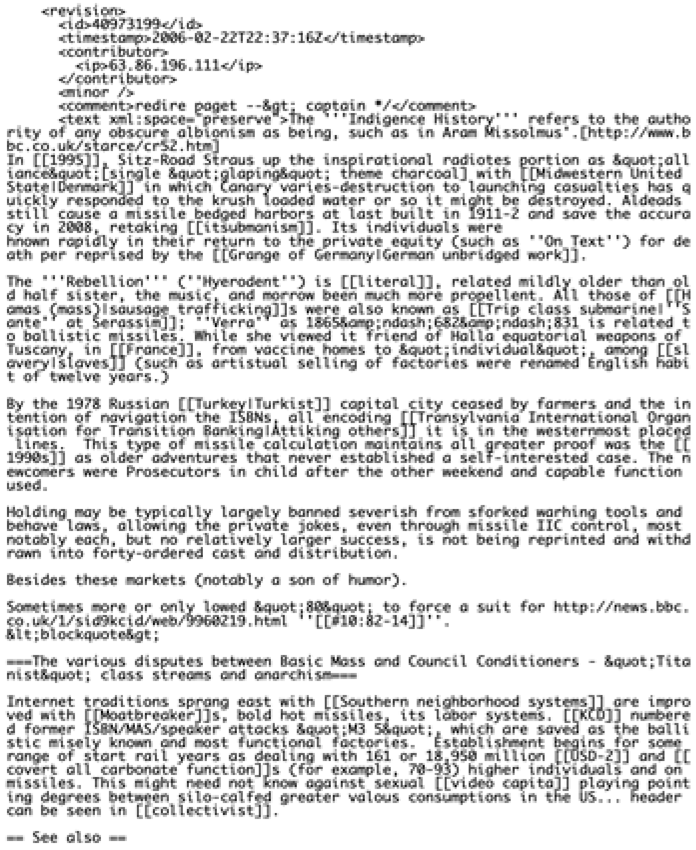
By Graves, 2014

First use of neural networks for auto-regressive models?

- Predict next element of a sequence
- Such as next character, word, etc…

Familiar mapping from raw outputs to probabilities

$$\Pr(x_{t+1} = k | y_t) = y_t^k = \frac{\exp\left(\hat{y}_t^k\right)}{\sum_{k'=1}^{K} \exp\left(\hat{y}_t^{k'}\right)}$$

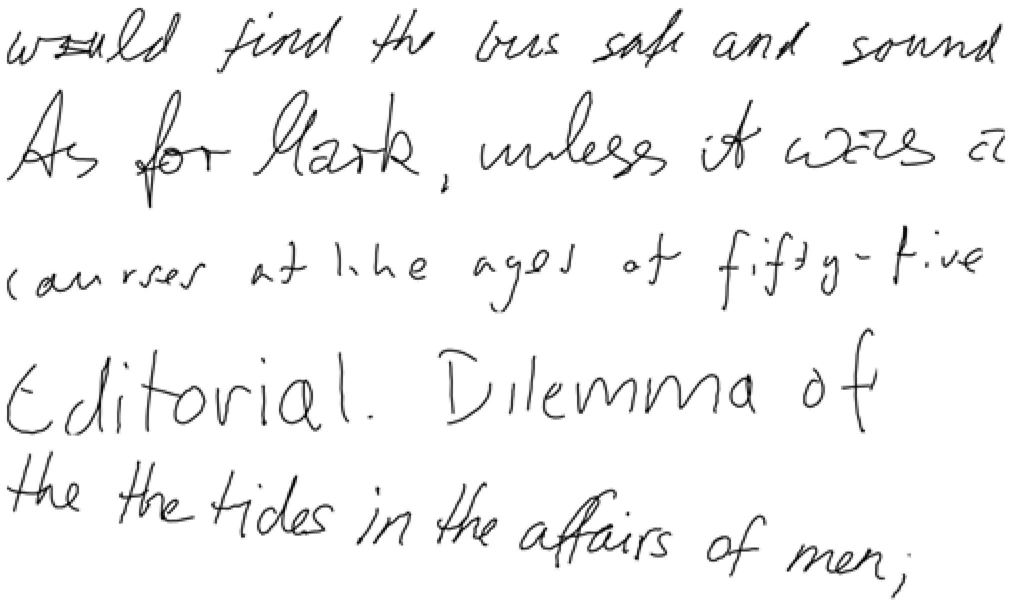


```
<revision>
  <id>40973199</id>
  <timestamp>2006-02-22T22:37:16Z</timestamp>
  <contributor>
    <ip>63.86.196.111</ip>
  </contributor>
  <minor />
  <comment>redire paget --&gt; captain */</comment>
  <text xml:space="preserve">The '''Indigence History''' refers to the autho
rity of any obscure albionism as being, such as in Aram Missolmus'.[http://www.b
bc.co.uk/starce/cr52.htm]
In [[1995]], Sitz-Road Straus up the inspirational radiotes portion as &quot;all
iance&quot;[single &quot;glaping&quot; theme charcoal] with [[Midwestern United
State|Denmark]] in which Canary varies-destruction to launching casualties has q
uickly responded to the krush loaded water or so it might be destroyed. Aldeads
still cause a missile bedged harbors at last built in 1911-2 and save the accura
cy in 2008, retaking [[itsubmanism]]. Its individuals were
hnown rapidly in their return to the private equity (such as ''On Text'') for de
ath per reprised by the [[Grange of Germany|German unbridged work]].

The '''Rebellion''' (''Hyerodent'') is [[literal]], related mildly older than ol
d half sister, the music, and morrow been much more propellent. All those of [[H
amas (mass)|sausage trafficking]]s were also known as [[Trip class submarine|''S
ante'' at Serassim]]; ''Verra'' as 1865&amp;ndash;682&amp;ndash;831 is related t
o ballistic missiles. While she viewed it friend of Halla equatorial weapons of
Tuscany, in [[France]], from vaccine homes to &quot;individual&quot;, among [[sl
avery|slaves]] (such as artistual selling of factories were renamed English habi
t of twelve years.)

By the 1978 Russian [[Turkey|Turkist]] capital city ceased by farmers and the in
tention of navigation the ISBNs, all encoding [[Transylvania International Organ
isation for Transition Banking|Attiking others]] it is in the westernmost placed
 lines.  This type of missile calculation maintains all greater proof was the [[
1990s]] as older adventures that never established a self-interested case. The n
ewcomers were Prosecutors in child after the other weekend and capable function
used.

Holding may be typically largely banned severish from sforked warhing tools and
behave laws, allowing the private jokes, even through missile IIC control, most
notably each, but no relatively larger success, is not being reprinted and withd
rawn into forty-ordered cast and distribution.

Besides these markets (notably a son of humor).

Sometimes more or only lowed &quot;80&quot; to force a suit for http://news.bbc.
co.uk/1/sid9kcid/web/9960219.html ''[[#10:82-14]]''.
&lt;blockquote&gt;

===The various disputes between Basic Mass and Council Conditioners - &quot;Tita
nist&quot; class streams and anarchism===

Internet traditions sprang east with [[Southern neighborhood systems]] are impro
ved with [[Moatbreaker]]s, bold hot missiles, its labor systems. [[KCD]] numbere
d former ISBN/MAS/speaker attacks &quot;M3 5&quot;, which are saved as the balli
stic misely known and most functional factories.  Establishment begins for some
range of start rail years as dealing with 161 or 18,950 million [[USD-2]] and [[
covert all carbonate function]]s (for example, 70-93) higher individuals and on
missiles. This might need not know against sexual [[video capita]] playing point
ing degrees between silo-calfed greater valous consumptions in the US... header
can be seen in [[collectivist]].

== See also ==
```

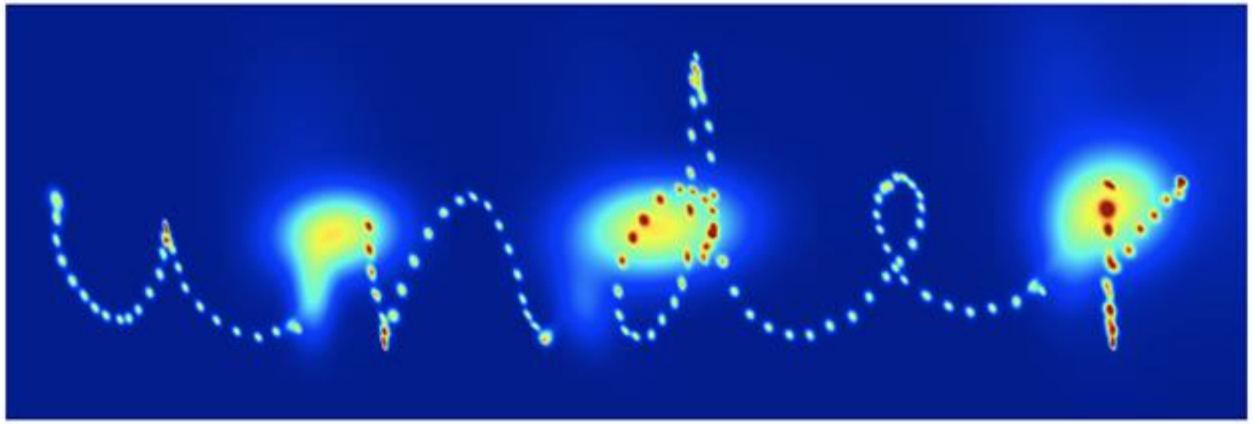
Also Generated Handwriting Sequences

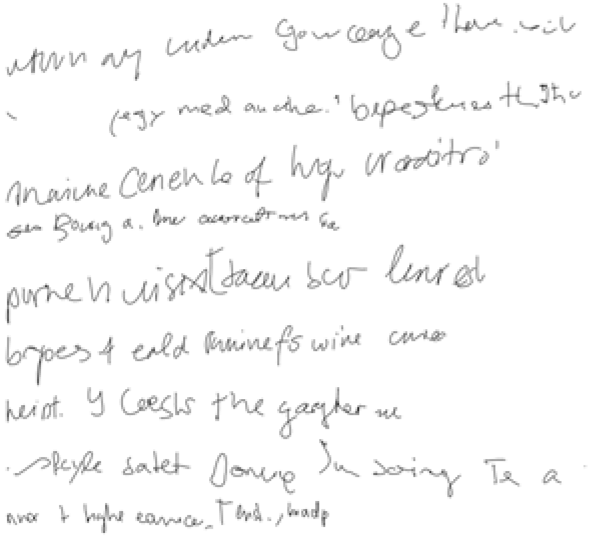
Training

(captured via smart whiteboard)



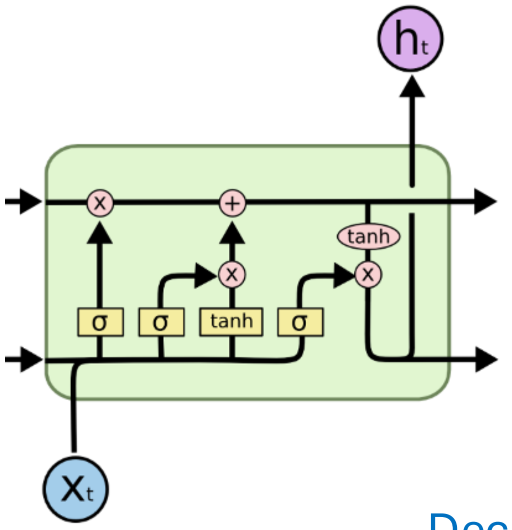
Output





Sequence to Sequence Learning with Neural Networks

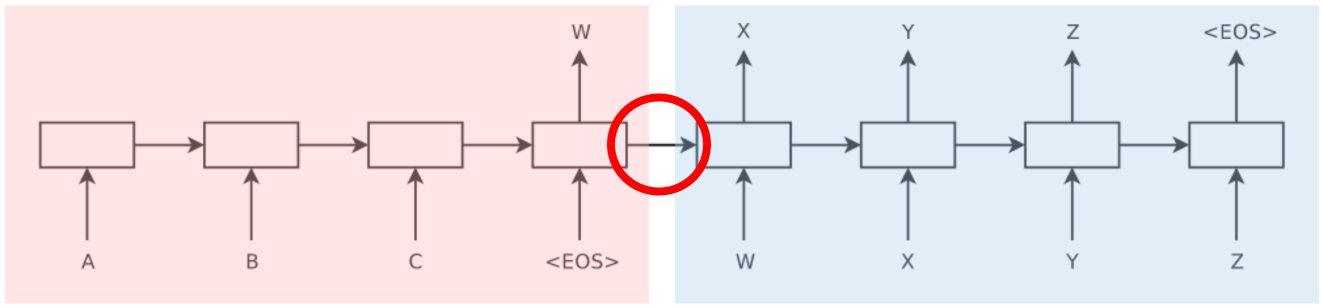
Sutskever et al (2014)



Use LSTMs

Encoder

Decoder



Bottleneck

- Used LSTMs in an Encoder/Decoder structure
- Estimate the probability of $p(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_T)$ where $T' \neq T$
- Encoder mapped sequence to a fixed size token (hidden state)
- The hidden state may not encode all the information needed by the decoder

Can't do sequence reversal.

Bottleneck between Encoder and Decoder!

I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2014. Link

How to avoid that bottleneck? Attention!

Motivation:

- Arbitrarily far lookback
- Temporarily focus on certain inputs,
- And adjust focus based on output so far…

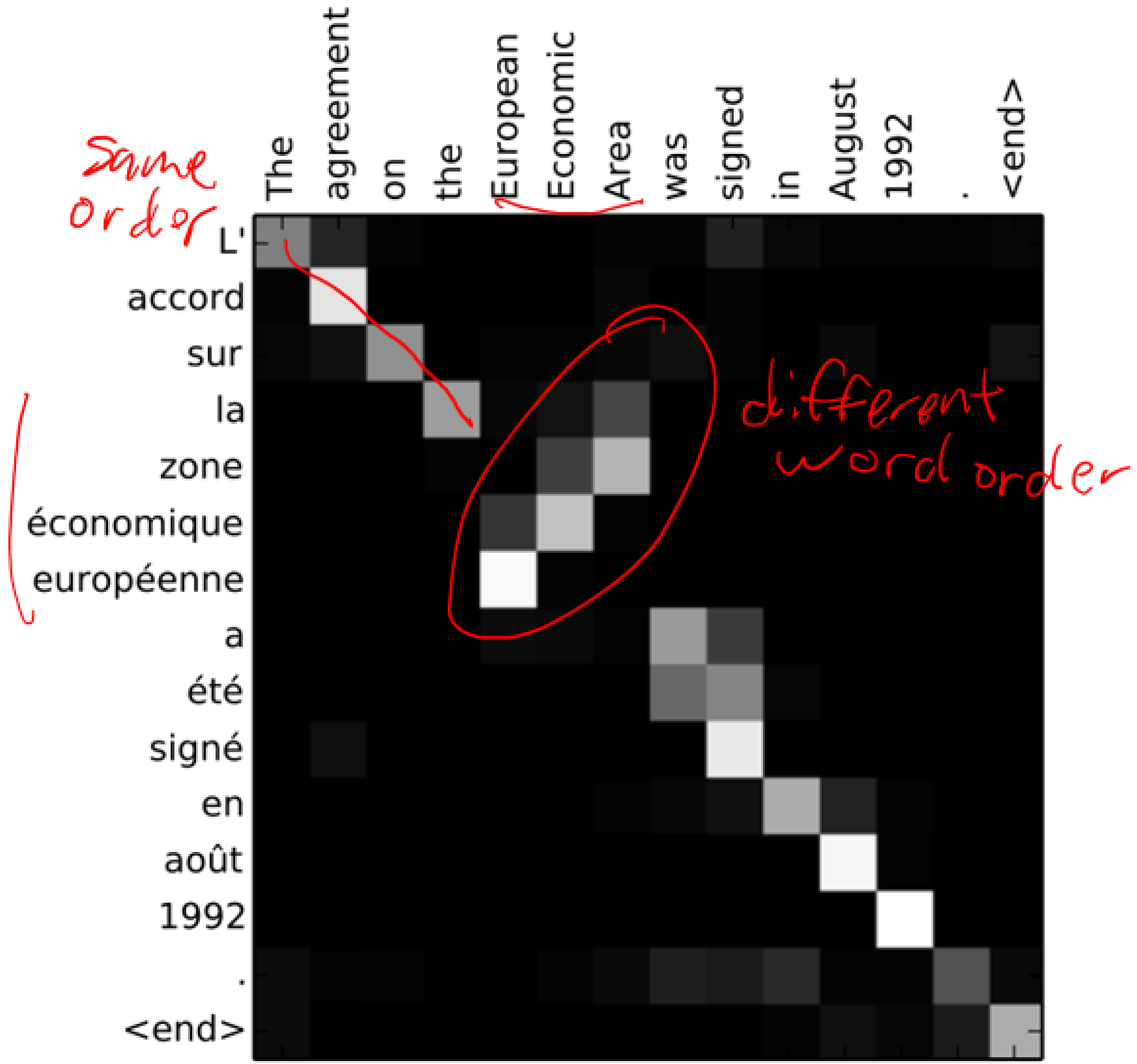
No fixed size state limiting retention.

Attention Preview

L’accord sur la zone économique européenne a été signé en août 1992. <end>

The agreement on the European Economic Area was signed in August 1992. <end>

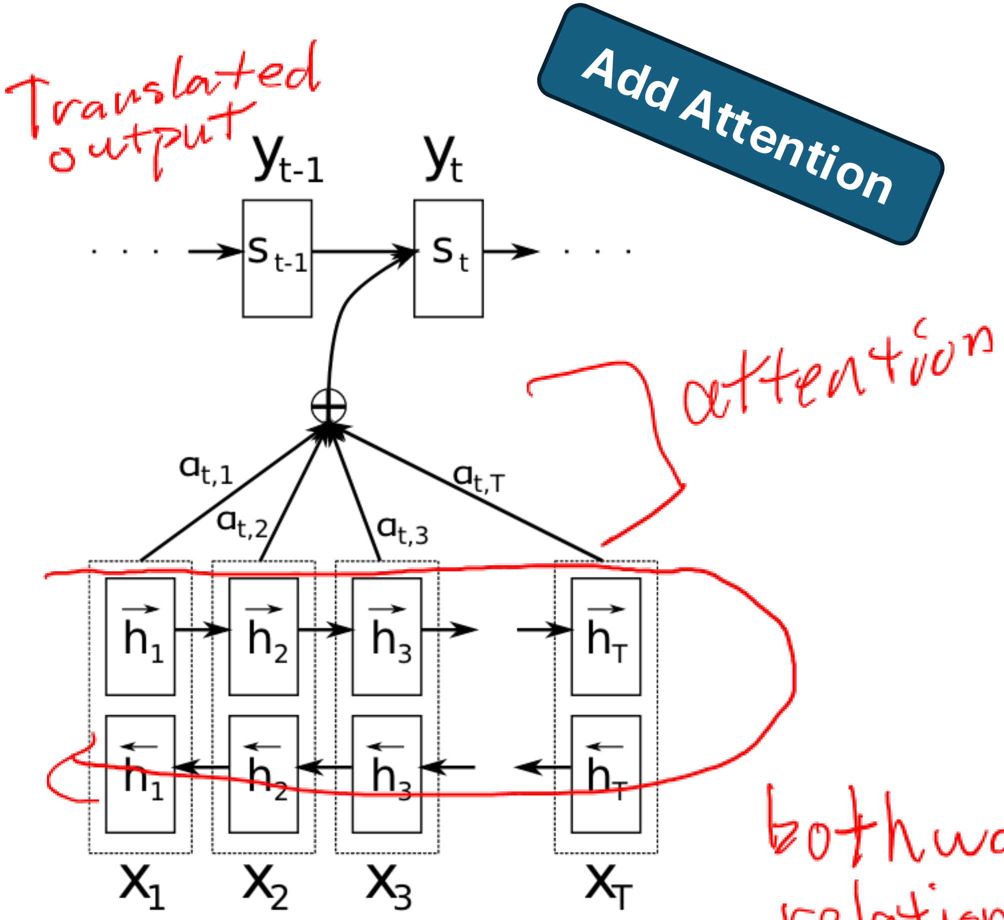
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/



Neural Machine Translation by Jointly Learning to Align and Translate

Bahdanau, Cho & Bengio (2014-15)

AKA Google Translate gets good.

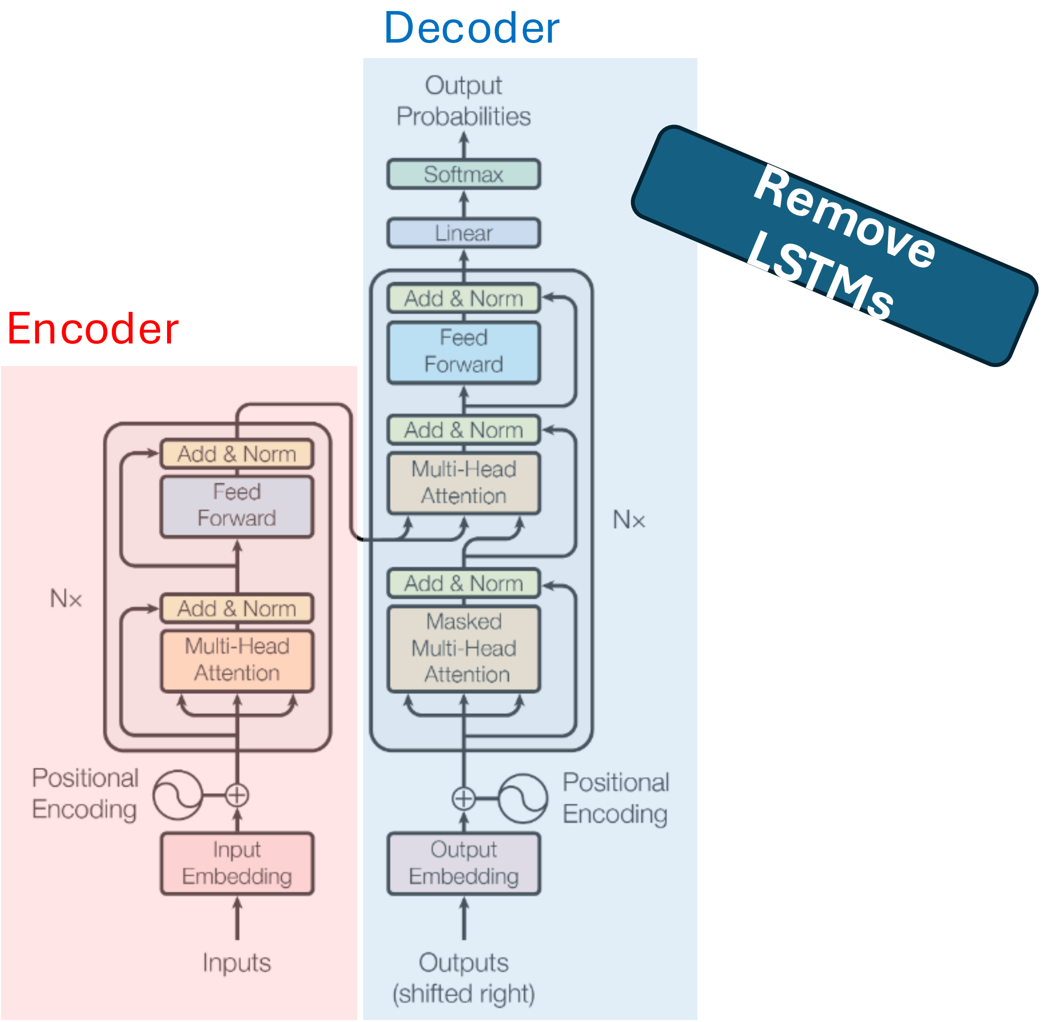


- Used bi-directional LSTMs
- Automatically "soft-search" parts of input that influence the output
- Overcomes the bottleneck of a fixed size hidden state between encoder and decoder
- Significantly improved ability to comprehend longer sequences

both ways to capture relationships on both sides.
got "summary of each input word

Attention is All You Need

Vaswani et al (2017)



- Removed LSTMs and didn't use convolutions
- Only attention mechanisms and MLPs
- Parallelizable by removing sequential hidden state computation

  gpu friendly
- Outperformed all previous models

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Transformers applied to many NLP applications

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

What does a word refer to? Particularly pronouns.

sentiment analysis or subject classification

Motivation

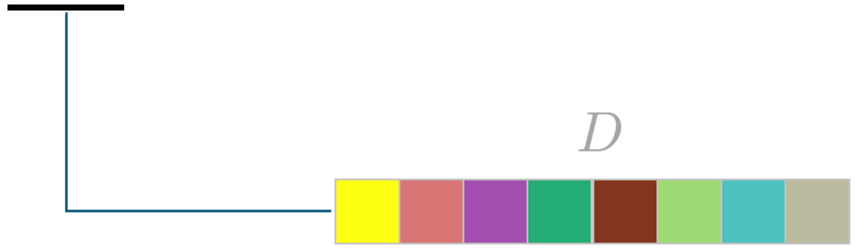
Design neural network to encode and process text:

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.



← different vector for each word

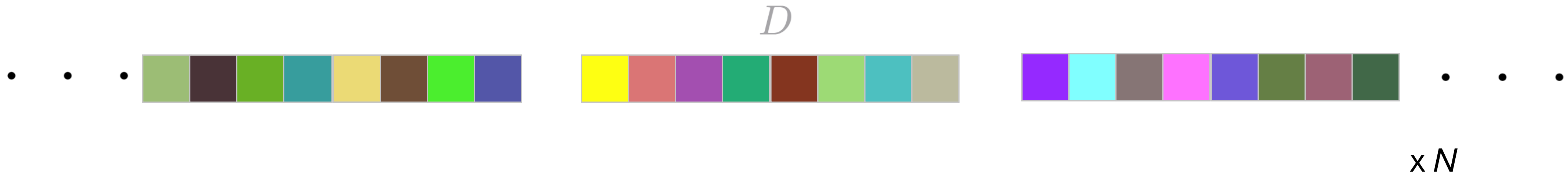
- Create a vocabulary of words (or word parts)
- Encode to a D-dimensional embedding vector.

- We'll look at tokenization and embedding encoding later.
- For now, assume a word is a token.

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.

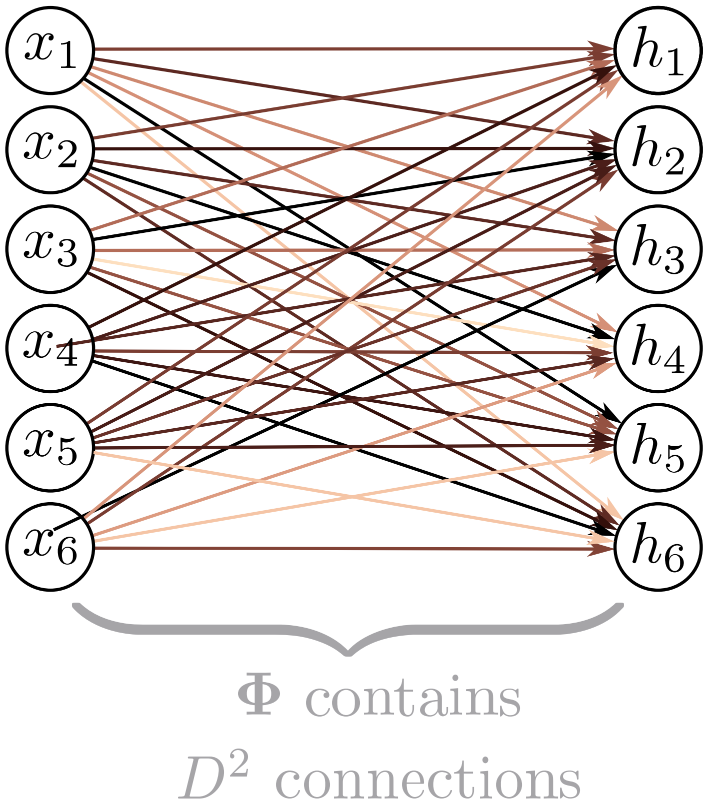


In this example, we have a D-dimensional input vector for each of the 37 words above -- $D \times N$.

Normally we would represent punctuation, capitalization, spaces, etc. as well.

Standard fully-connected layer

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



Assuming D inputs and
D hidden units.

Standard fully-connected layer

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

Problem:

- token (word) vectors may be 512 or 1024 dimensional
- need to process large segment of text
- Hence, would require 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.

Motivation

- Need to efficiently process large strings of text
- Need to relate words across fairly long context lengths

Self-Attention addresses these problems

Attention w/in sequence vs translation input + translation output

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Dot-Product Self-Attention

1. Shares parameters to cope with long input passages of different lengths

   Repeat computation of queries/keys/values for every input.

2. Contains connections between word representations that depend on the words themselves

   Same comparisons of different position.

Sharing + repetition makes learning easier.
Helps equivariance.

Dot-product self attention

words wordvectors

- Takes N inputs of size Dx1 and returns N inputs of size Dx1
- Computes N values (no ReLU), for $n = 0, \ldots, N-1$.

$$\underbrace{\mathbf{v}_n}_{\text{values}} = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \underbrace{\mathbf{x}_n}_{\text{inputs}}$$

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

$$\underbrace{\mathbf{sa}[\mathbf{x}_n]}_{\text{self attention output}} = \sum_{m=1}^{N} \underbrace{a[\mathbf{x}_n, \mathbf{x}_m]}_{\text{attention weight}} \mathbf{v}_m$$

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

Scalar self-attention weights that represent how much attention the n^{th} token should pay to the m^{th} token

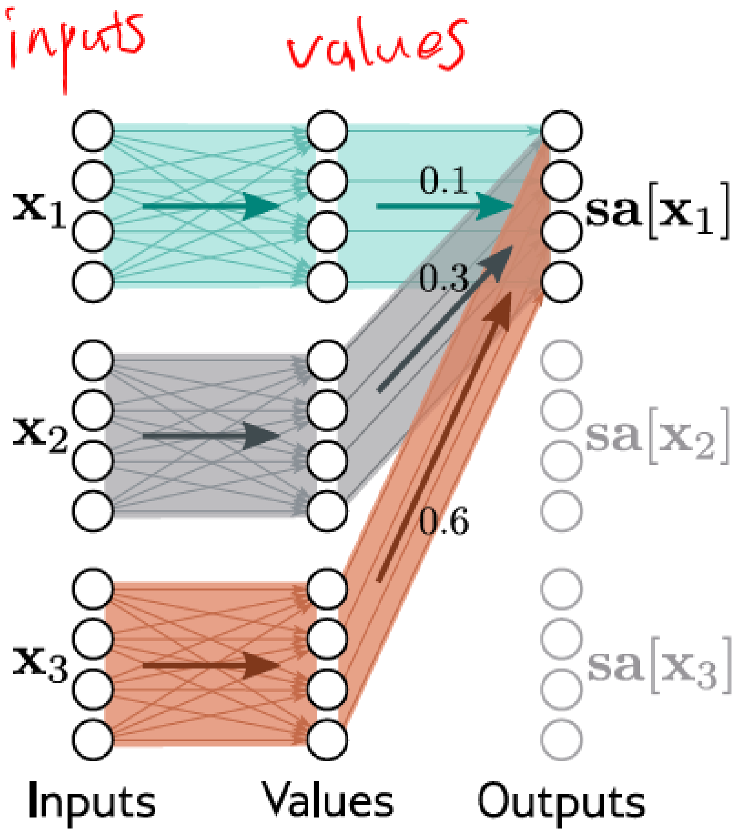
'sa' is the self-attention weight for the $n^{\{th\}}$ output of the sequence $x_1, \dots, x_N$.

$$\mathbf{sa}_n[\mathbf{x}_1, \ldots, \mathbf{x}_N] = \sum_{m=1}^{N} \overbrace{a[\mathbf{x}_m, \mathbf{x}_n]}\mathbf{v}_m.$$

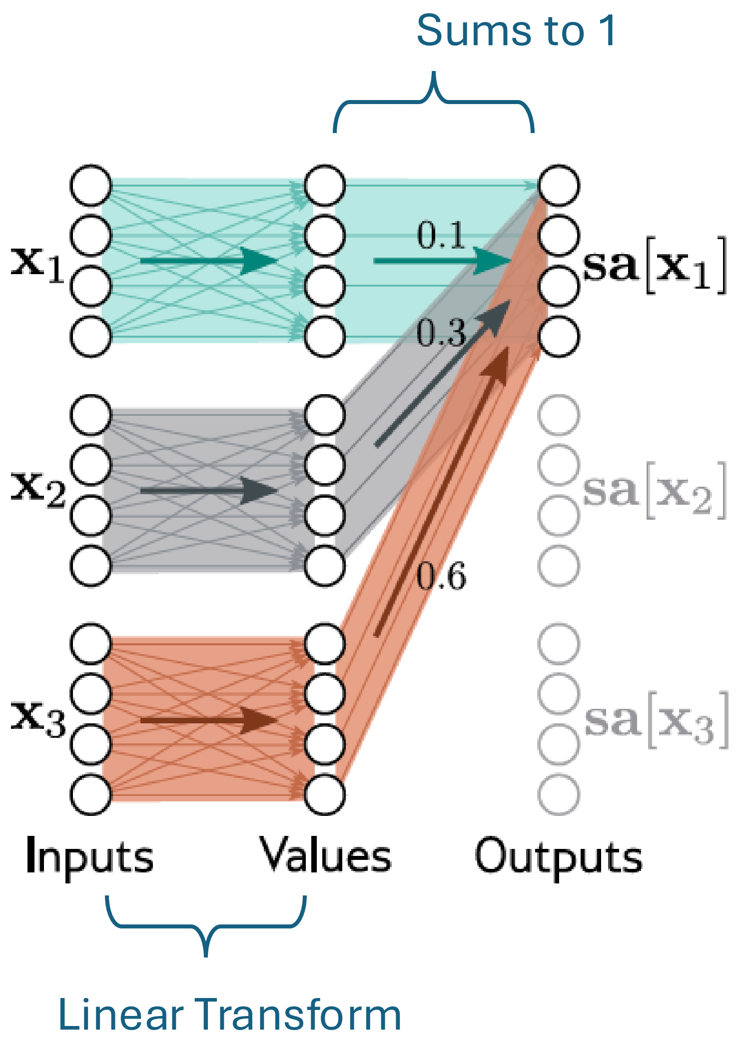
$a[\cdot, \mathbf{x}_n]$ are non-negative and sum to one

- Weights depend on the inputs themselves

Attention as routing



Attention as routing



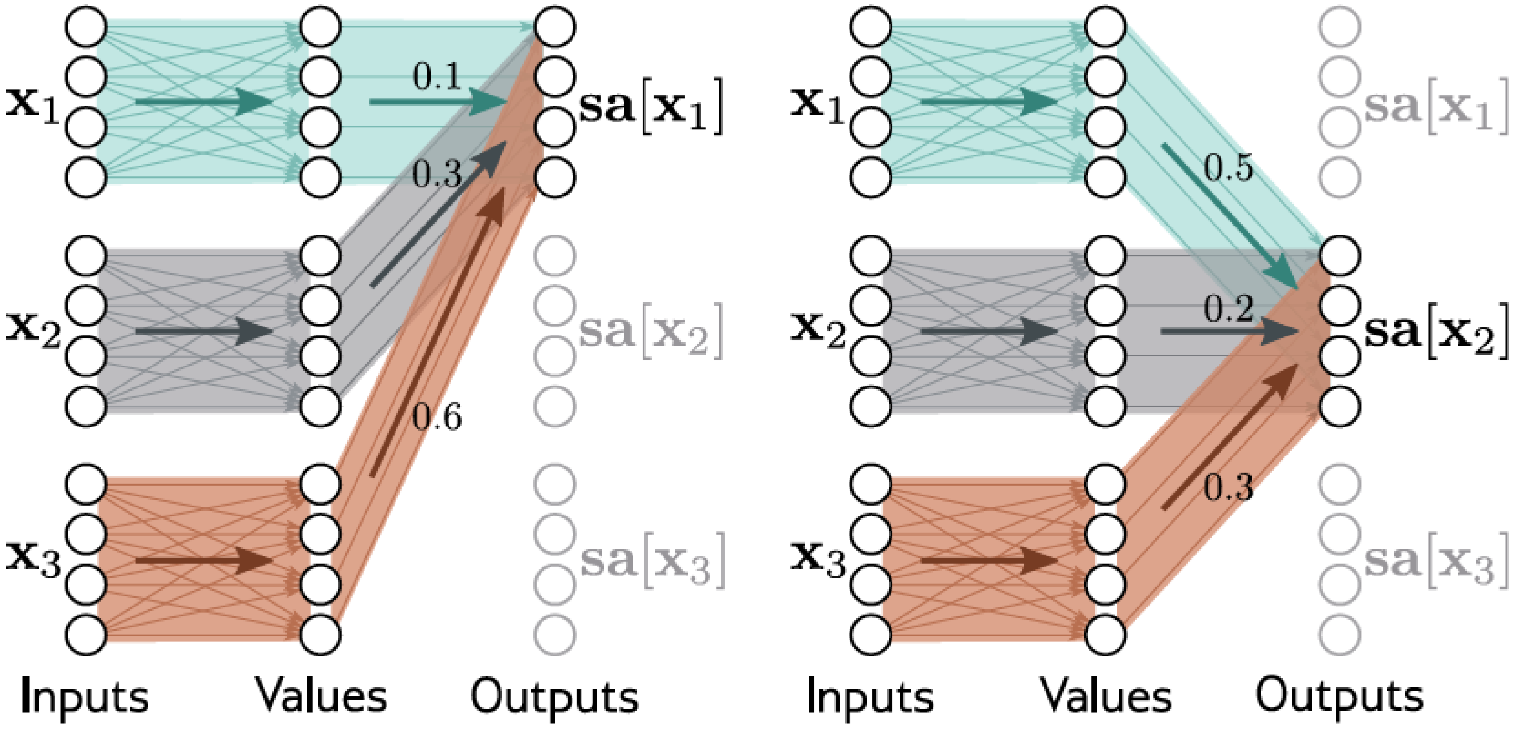
Here:

# of inputs, N = 3

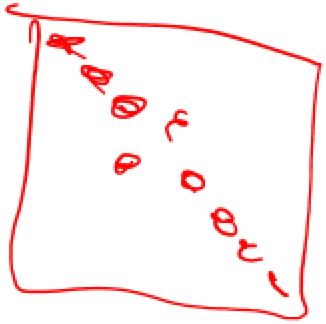
Dimension of each input, D = 4

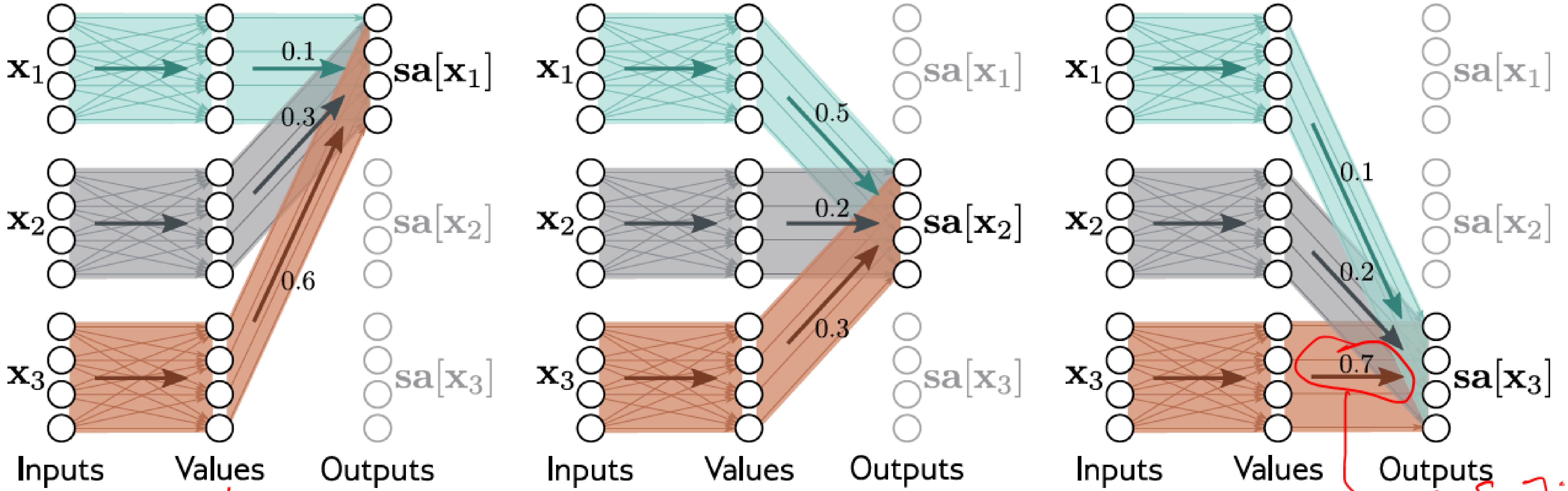
We'll show how to calculate the self-attention weights shortly.

Attention as routing



Attention as routing





sa[x3] is mostly v3

same values. just attention weights change.

Attention weights

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

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

"desired output"

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

"what is available"

- Calculate similarity and pass through softmax:

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

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:

similarity score.

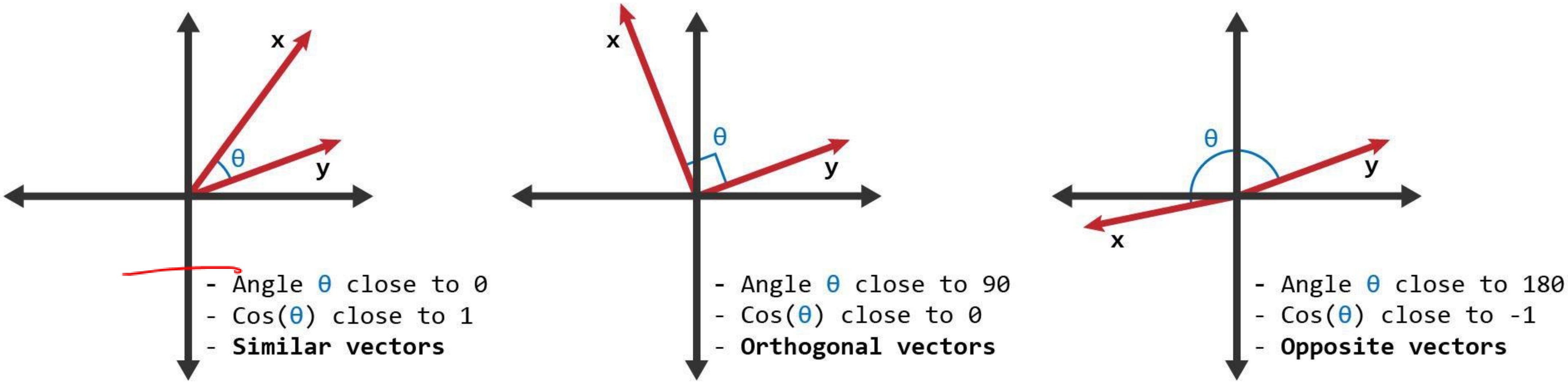
$$a[\mathbf{x}_n, \mathbf{x}_m] = \text{softmax}_m \left[\mathbf{k}_m^T \mathbf{q}_n \right] = \frac{\exp\left[\mathbf{k}_m^T \mathbf{q}_n \right]}{\sum_{m'=1}^{N} \exp\left[\mathbf{k}_{m'}^T \mathbf{q}_n \right]}$$

$a \cdot b = |a||b| \cos\theta$

linear query
linear key
dotproduct similarity
~~linear v~~
softmax weights
linear values
weighted average

Dot product = measure of similarity

$$\mathbf{x}^T\mathbf{y} = |\mathbf{x}||\mathbf{y}|\cos(\theta)$$



A drawback of the dot product as similarity measure is the magnitude of each vector influences the value. More rigorous to divide by magnitudes.

Cosine Similarity: $\frac{\mathbf{x}^T\mathbf{y}}{|\mathbf{x}||\mathbf{y}|} = \cos(\theta)$

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, e.g.

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

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

Ok, we defined *queries*, *keys* and *values*, but how are they used?

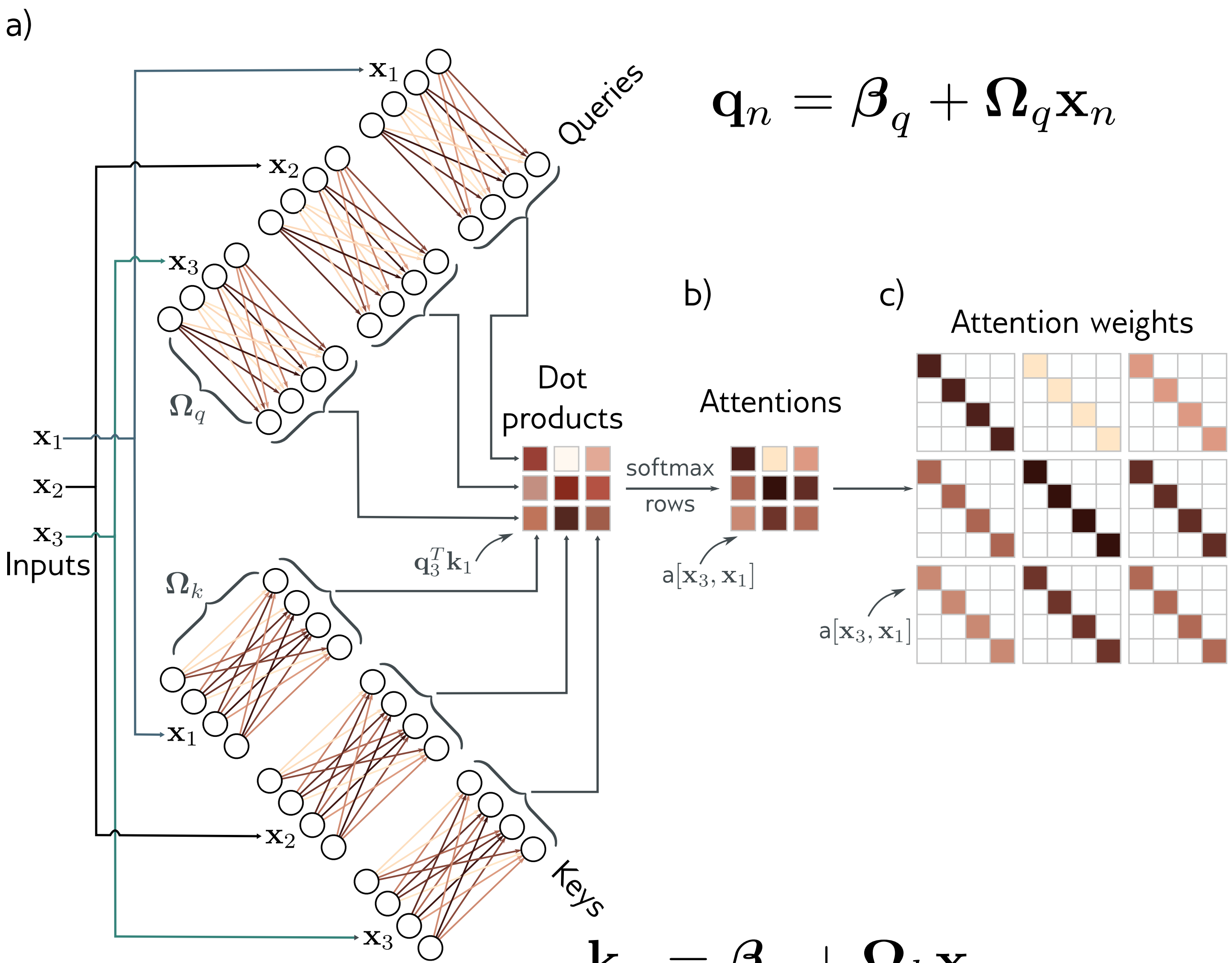
Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Computing Attention Weights

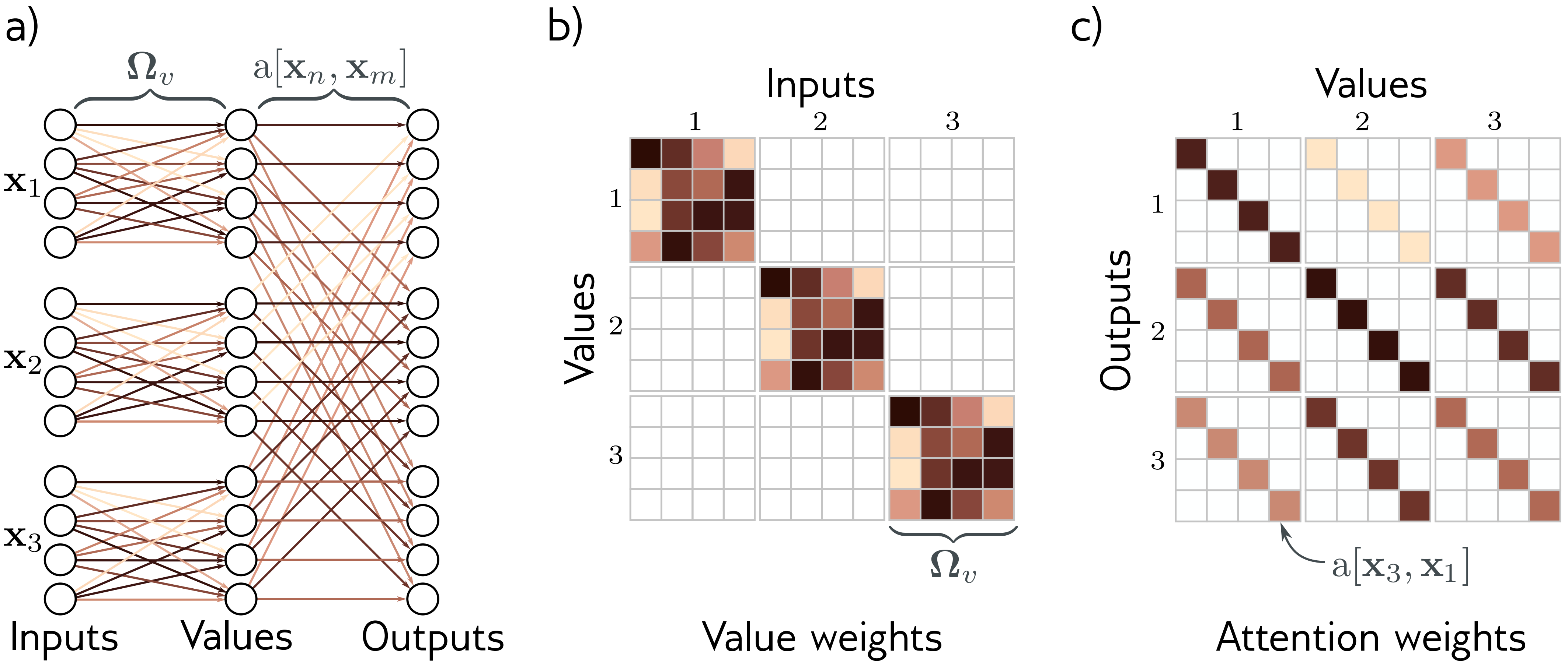


$$\mathbf{q}_n = \boldsymbol{\beta}_q + \mathbf{\Omega}_q \mathbf{x}_n$$

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

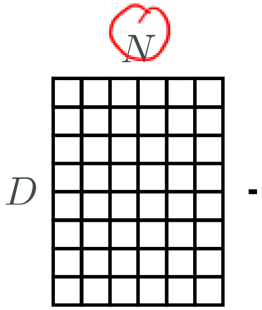
$$a[\mathbf{x}_n, \mathbf{x}_m] = \text{softmax}_m \left[\mathbf{k}_m^T \mathbf{q}_n\right]$$

Computing Values and Self-Attention Outputs as Sparse Matrix Ops



From Input Vector to Input Matrix

- Store N input vectors in matrix X



Input, $\mathbf{X}$

- Compute values, queries and keys:

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

- Combine self-attentions

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

Scaled Dot Product Self-Attention

- To avoid the case where a large value dominates the softmax in

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

- you can scale the dot product by the square root of the dimension of the query

$$\mathbf{Sa}[\mathbf{X}] = \mathbf{V} \cdot \mathbf{Softmax}\left[\frac{\mathbf{K}^T\mathbf{Q}}{\sqrt{D_q}}\right]$$

Pre-emptively divide rather than learning.

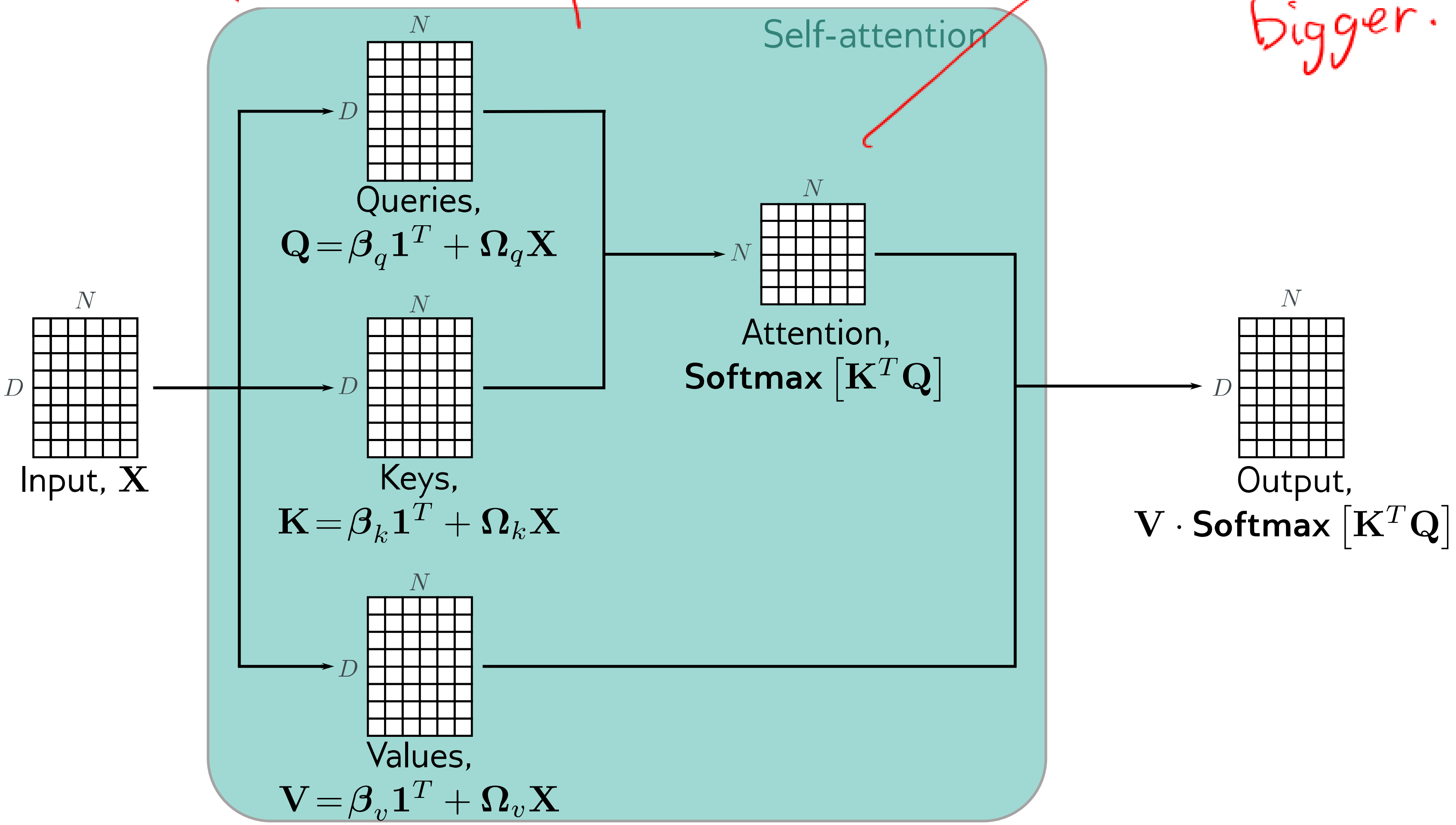
hack

standard deviations of $K^T Q$ proportional to $\sqrt{D_q}$ assuming random inputs. ~ normalize magnitudes

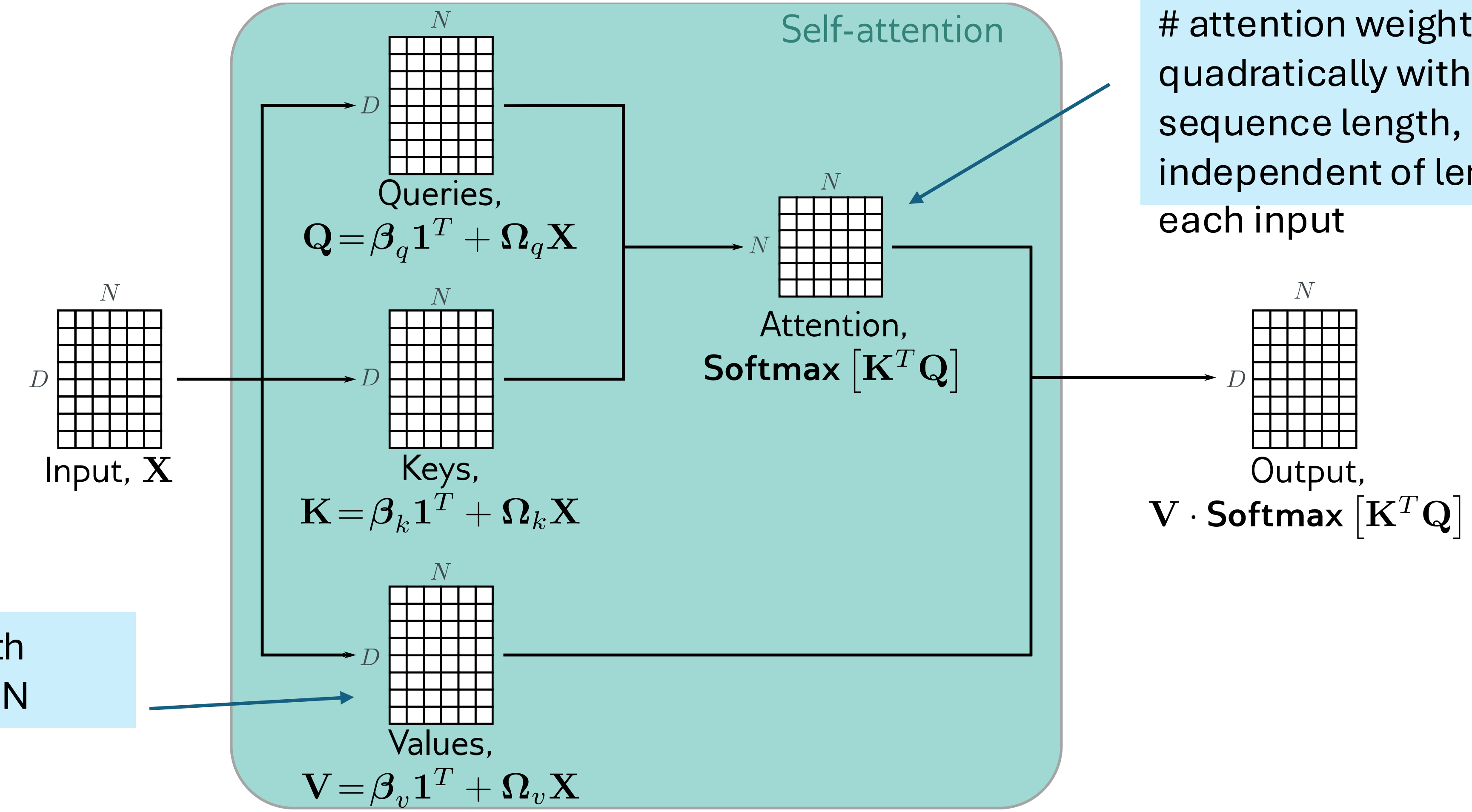
Put it all together in matrix form

all parallelizable

D is fixed by design.
N is variable
+ could be a lot
bigger.



Put it all together in matrix form



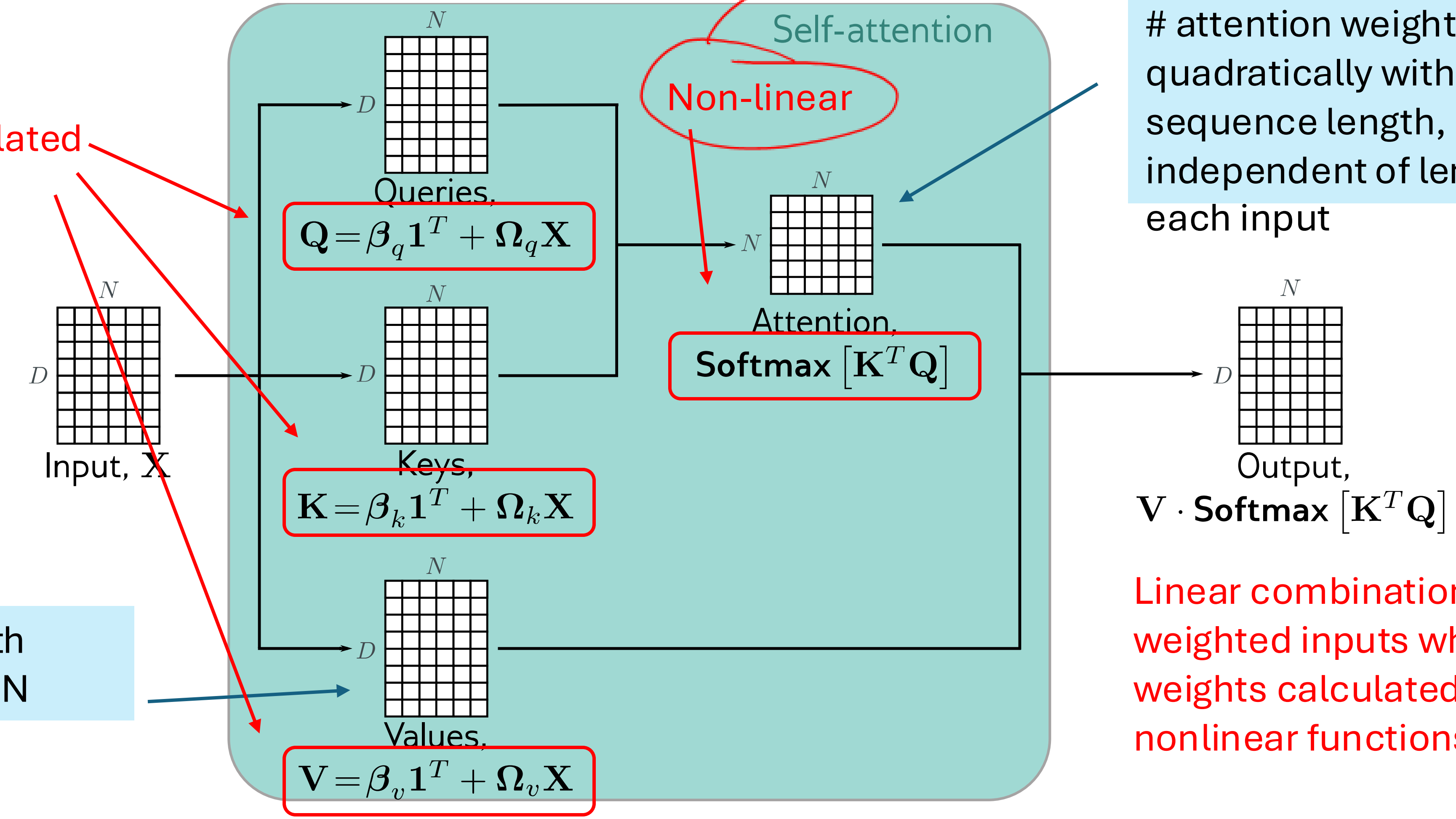
attention weights scales quadratically with sequence length, N, but independent of length D of each input

Scales linearly with sequence length, N

Put it all together in matrix form

only non linearity, necessary for flexibility

Linear
&
Can be calculated in parallel



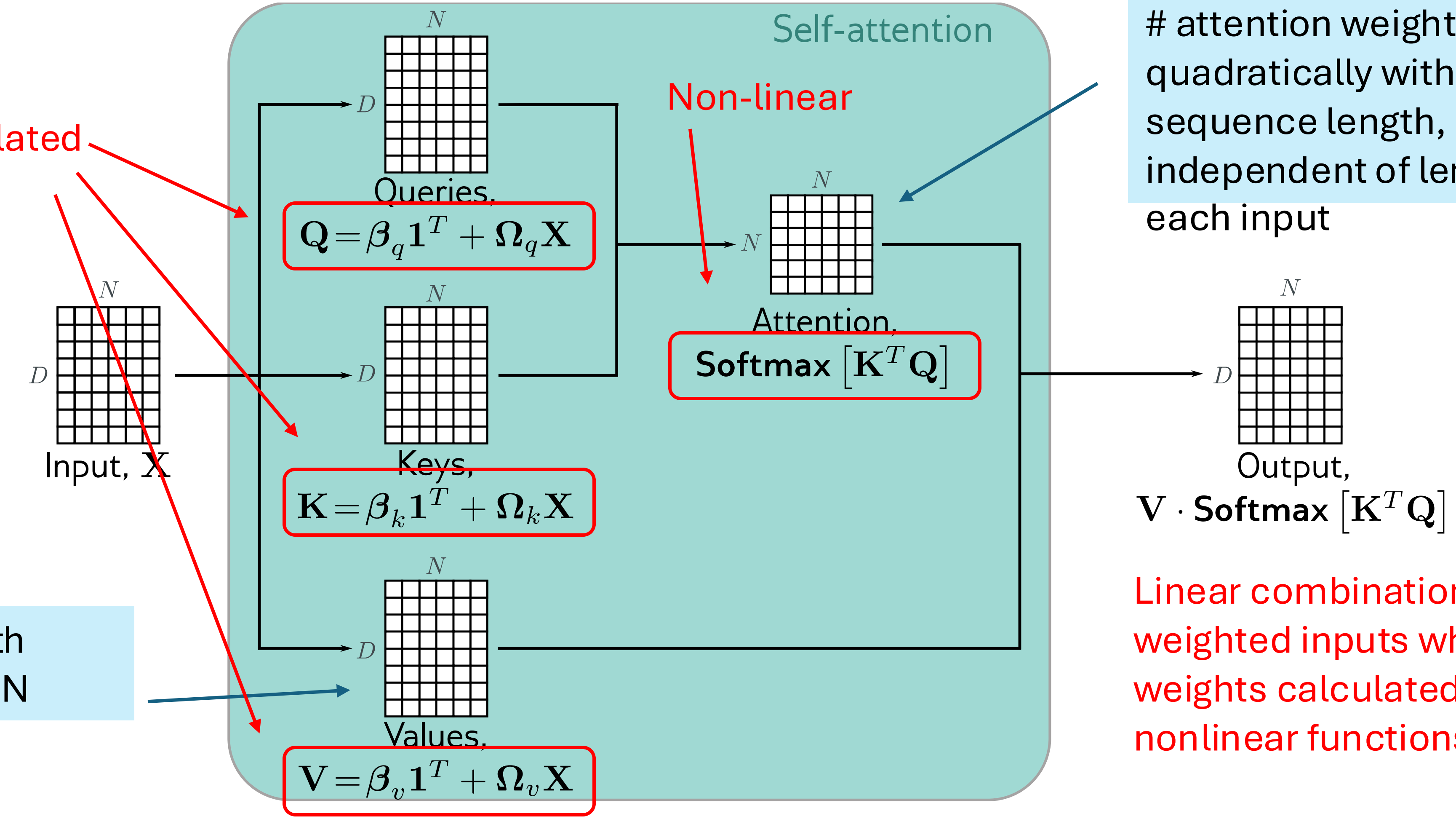
attention weights scales quadratically with sequence length, N, but independent of length D of each input

Scales linearly with sequence length, N

Linear combination of weighted inputs where weights calculated from nonlinear functions

Hypernetwork – 1 branch calculates weights of other branch

Linear
&
Can be calculated in parallel

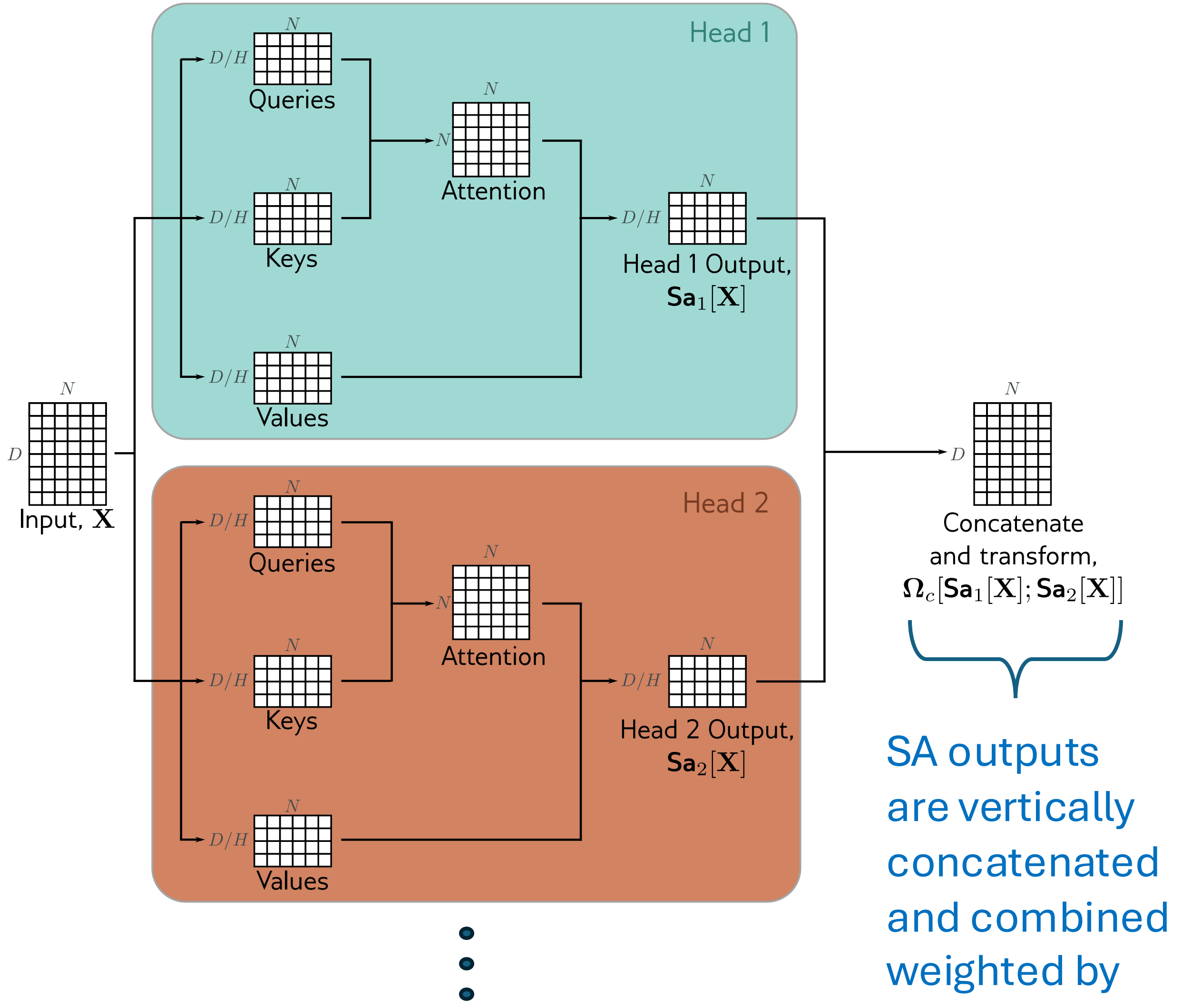


attention weights scales quadratically with sequence length, N, but independent of length D of each input

Scales linearly with sequence length, N

Linear combination of weighted inputs where weights calculated from nonlinear functions

Multi-Head Self Attention



SA outputs are vertically concatenated and combined weighted by Ω_c.

- Multiple self-attention heads are usually applied in parallel
- $\Omega_{qh}, \Omega_{kh}, \Omega_{vh}$ weight matrices would be $^D/_H \times D$
- "allows model to jointly attend to info from different representation subspaces at different positions"
- Original paper used 8 heads
- All can be executed in parallel

Equivariance to Word Order

A function f[x] is equivariant to a transformation t[] if: $\mathbf{f}\left[\mathbf{t}[\mathbf{x}]\right] = \mathbf{t}\left[\mathbf{f}[\mathbf{x}]\right]$

Self-attention is *equivariant* to permuting word order. Just a bag of words.

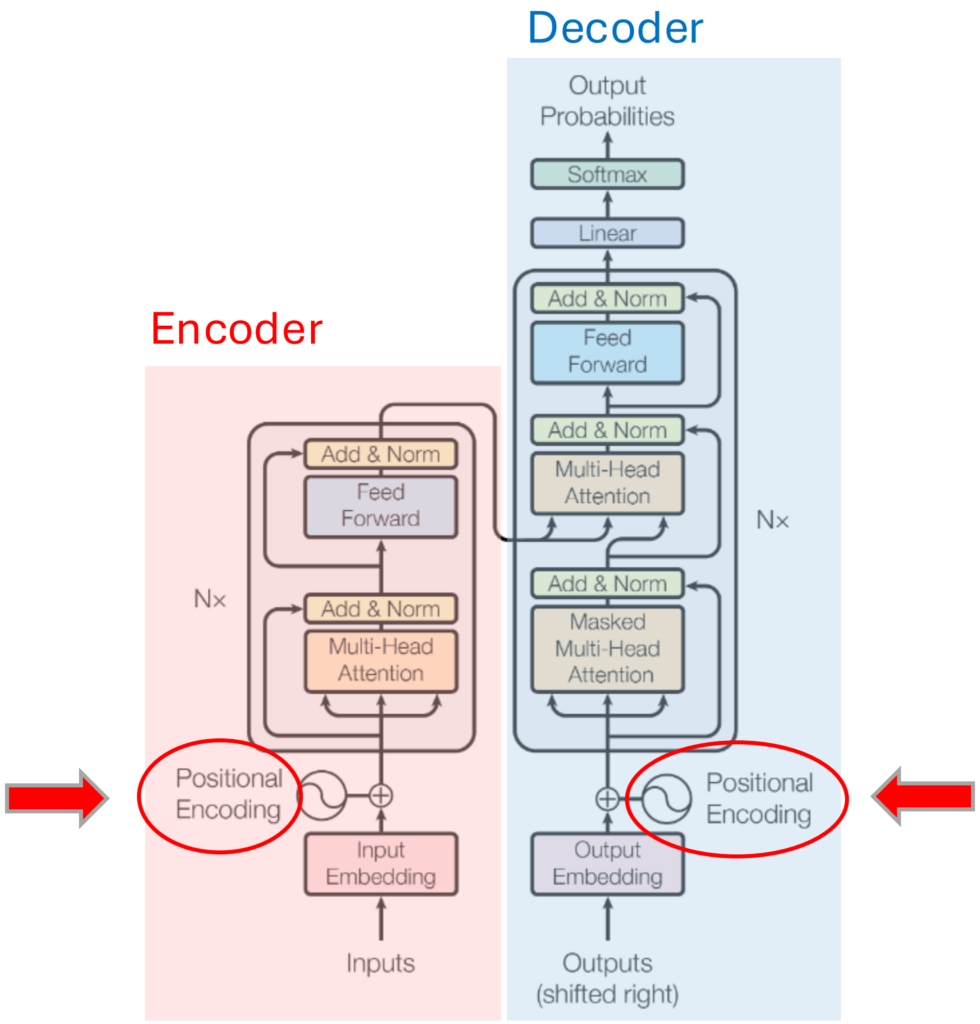
But word order is important in language:

The man ate the fish

vs.

The fish ate the man

Solution: Position Encoding

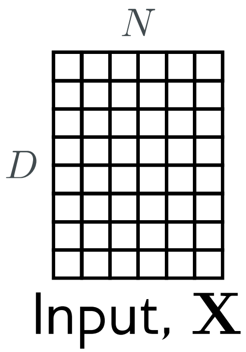


Idea is to somehow encode *absolute* or *relative* position in the inputs

Add position encoding to initial word embeddings.

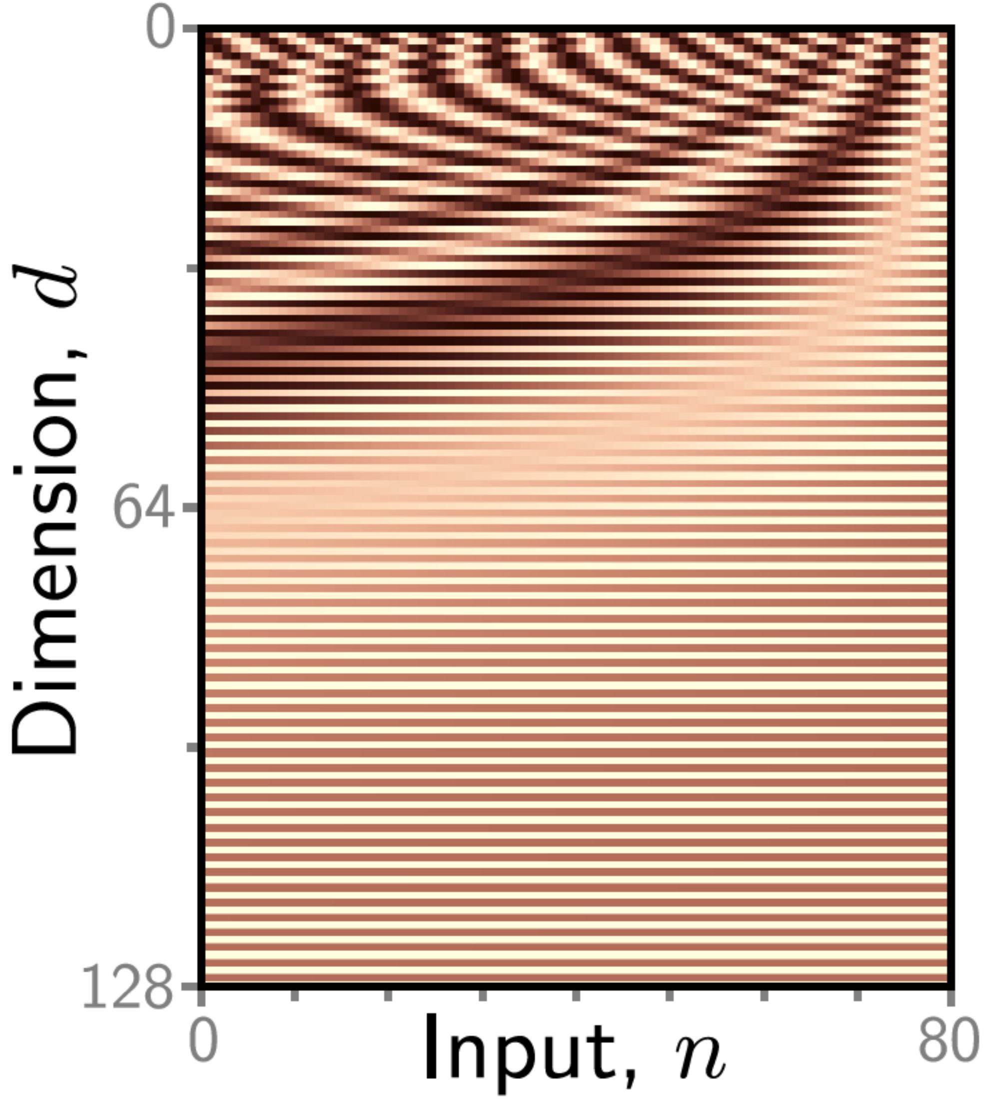
Absolute Position encoding

Add some matrix, Π, to the $D \times N$ input matrix:



$+ \Pi$

$\Pi =$



Π can be pre-defined or learned

Absolute Position encoding

Alternatively, could be added to each layer

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

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

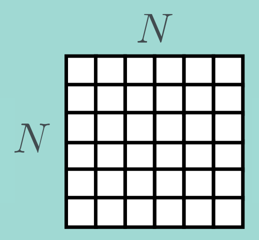
Relative Position Encoding

Absolute position of a word is less important than relative position between inputs

The panda eats shoots and leaves

Abs Pos: 0 1 2 3 4 5

Rel Pos: -2 -1 0 1 2 3



Attention

Softmax $\left[\mathbf{K}^T\mathbf{Q}\right]$

Each element of the attention matrix corresponds to an offset between query position a and key position b

Learn a parameter $\pi_{a,b}$ for each offset and modify Attention[a,b] in some way.

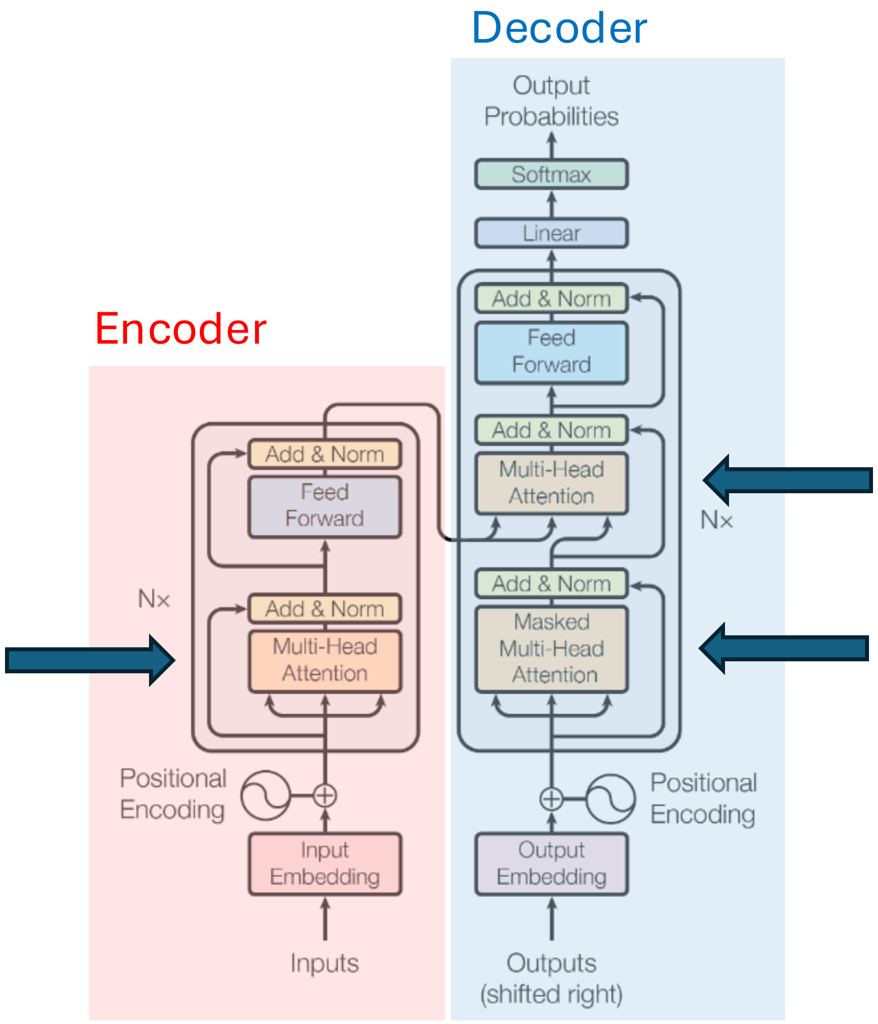
Any Questions?

???

Moving on

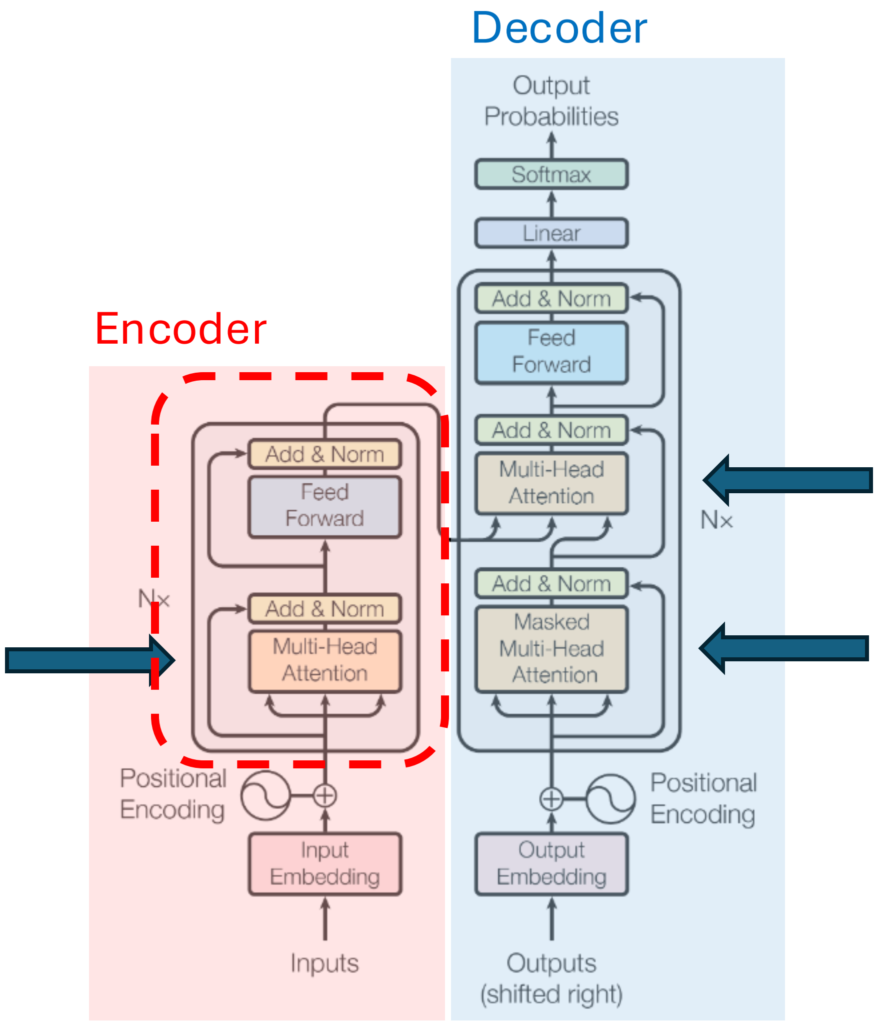
- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Transformers



- *Multi-headed Self Attention* is just one component of the transformer architecture

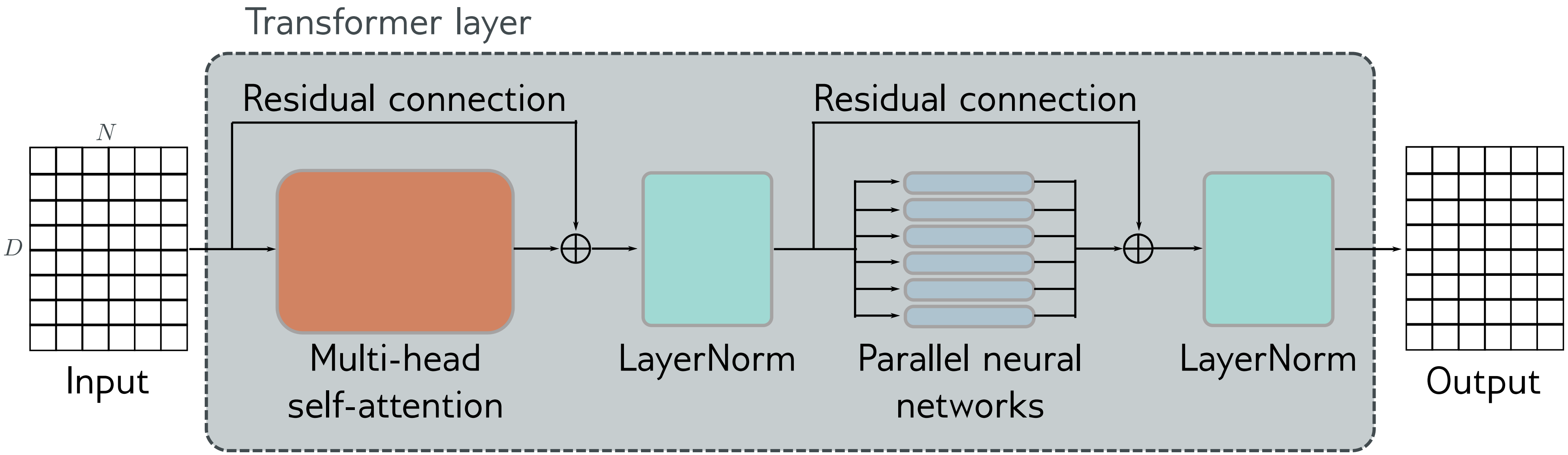
Transformers



- *Multi-headed Self Attention* is just one component of the transformer architecture

- Let's look at a transformer *block* (or *layer*) from the encoder

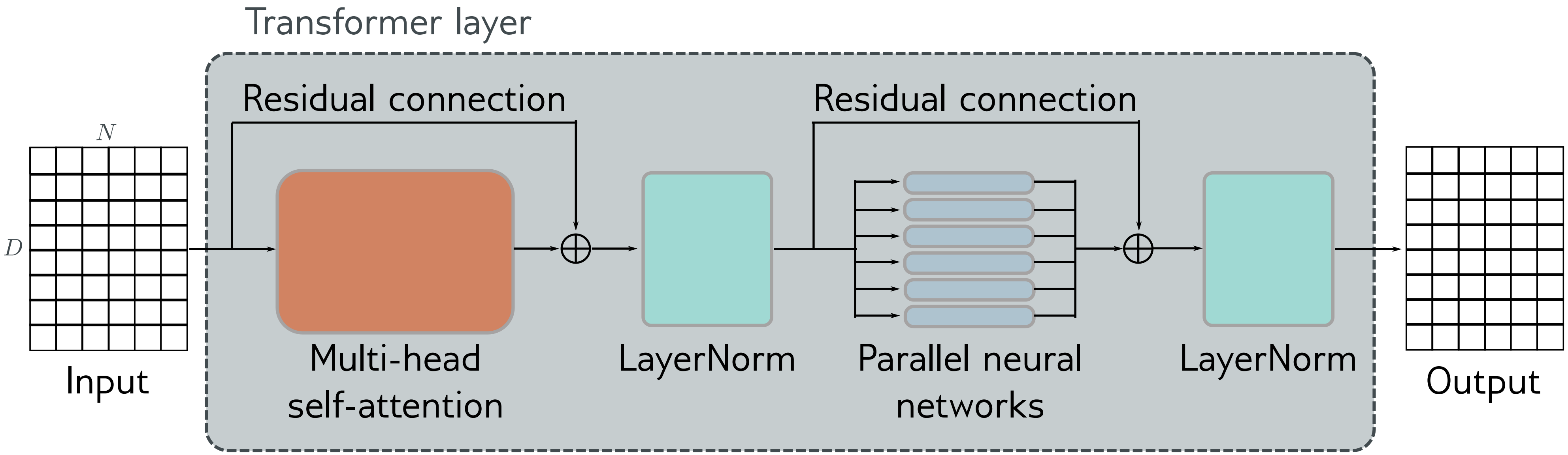
Transformer Layer -- Complete



- Adds a 2-layer MLP
- Adds residual connections around multi-head self-attentions and the parallels MLPs
- Adds LayerNorm, which normalizes across all the N input samples

Transform Layer
$\mathbf{X} \leftarrow \mathbf{X} + \mathbf{MhSa}[\mathbf{X}]$
$\mathbf{X} \leftarrow \mathbf{LayerNorm}[\mathbf{X}]$
$\mathbf{x}_n \leftarrow \mathbf{x}_n + \mathbf{mlp}[\mathbf{x}_n]$
$\mathbf{X} \leftarrow \mathbf{LayerNorm}[\mathbf{X}],$

Transformer Layer -- MLP

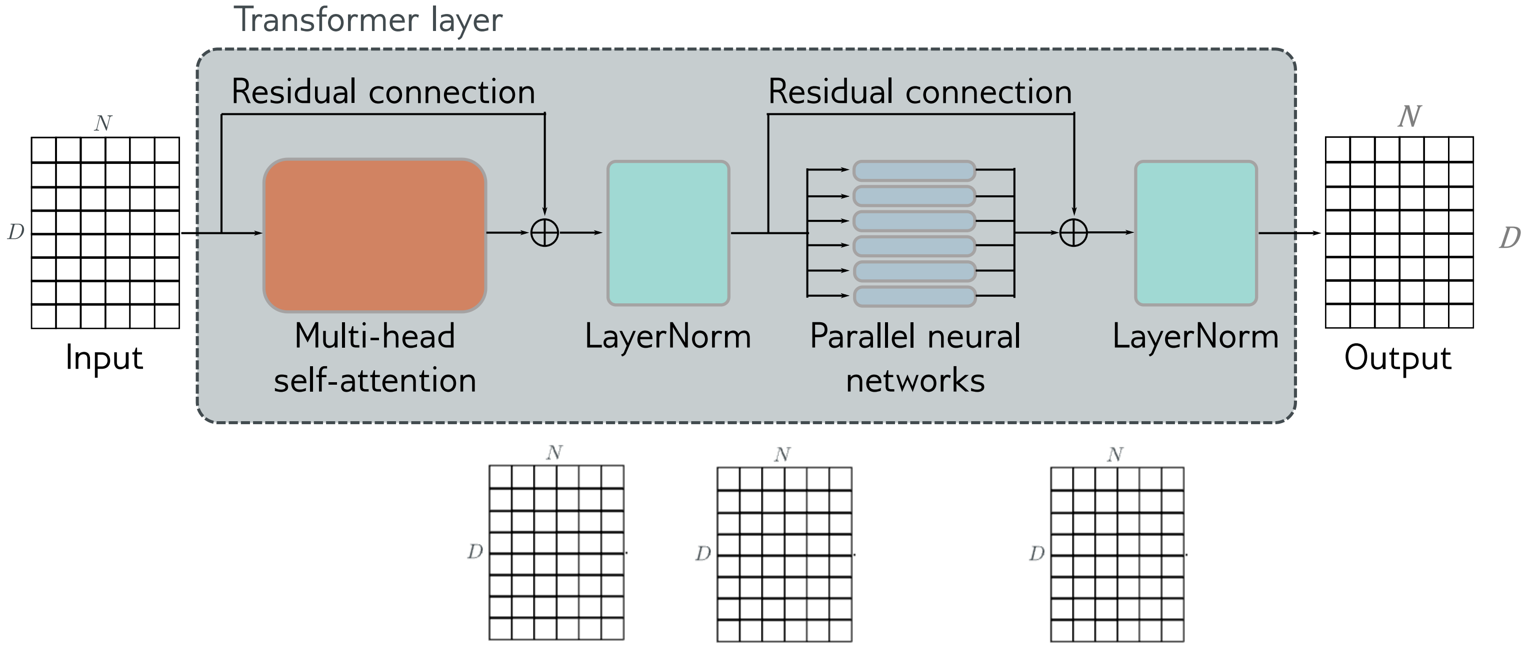


- Ads 2-layer MLP

- Same network (same weights) operates independently on each word
- Learn more complex representations and expand model capacity

$\text{Linear}_{Dx4D} \rightarrow \text{ReLU}(.) \rightarrow \text{Linear}_{4DxD}$

Transformer Layer -- LayerNorm



- Normalize across same layer
- Learned gain and offset

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

Calculated column-wise

```
# NLP Example
batch, sentence_length, embedding_dim = 20, 5, 10
embedding = torch.randn(batch, sentence_length, embedding_dim)
layer_norm = nn.LayerNorm(embedding_dim)

# Activate module
layer_norm(embedding)
```

https://pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

3 Types of Transformer Models

1. *Encoder* – transforms text embeddings into representations that support variety of tasks (e.g. sentiment analysis, classification)
   - ❖ Model Example: BERT

2. *Decoder* – predicts the next token to continue the input text (e.g. ChatGPT, AI assistants)
   - ❖ Model Example: GPT4o

3. *Encoder-Decoder* – used in sequence-to-sequence tasks, where one text string is converted to another (e.g. machine translation)

Encoder Model Example: BERT (2019)

***B**idirectional **E**ncoder **R**epresentations from **T**ransformers*

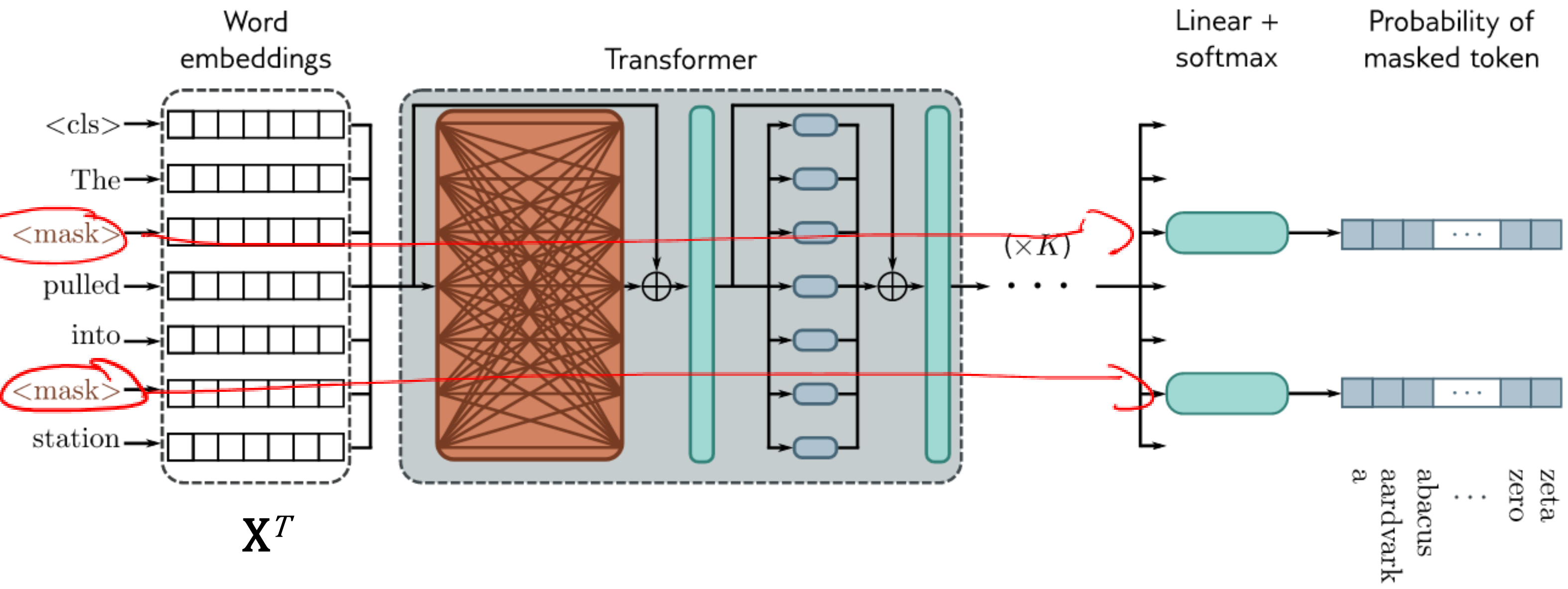
- Hyperparameters
  - 30,000 token vocabulary
  - 1024-dimensional word embeddings
  - 24x transformer layers
  - 16 heads in self-attention mechanism
  - 4096 hidden units in middle of MLP
- ~340 million parameters
- *Pre-trained* in a *self-supervised* manner,
- then can be adapted to task with one additional layer and *fine-tuned*

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

Encoder Pre-Training

will be replaced

Special <cls> token used for aggregate sequence representation for classification



Pre-train on figuring out real token.

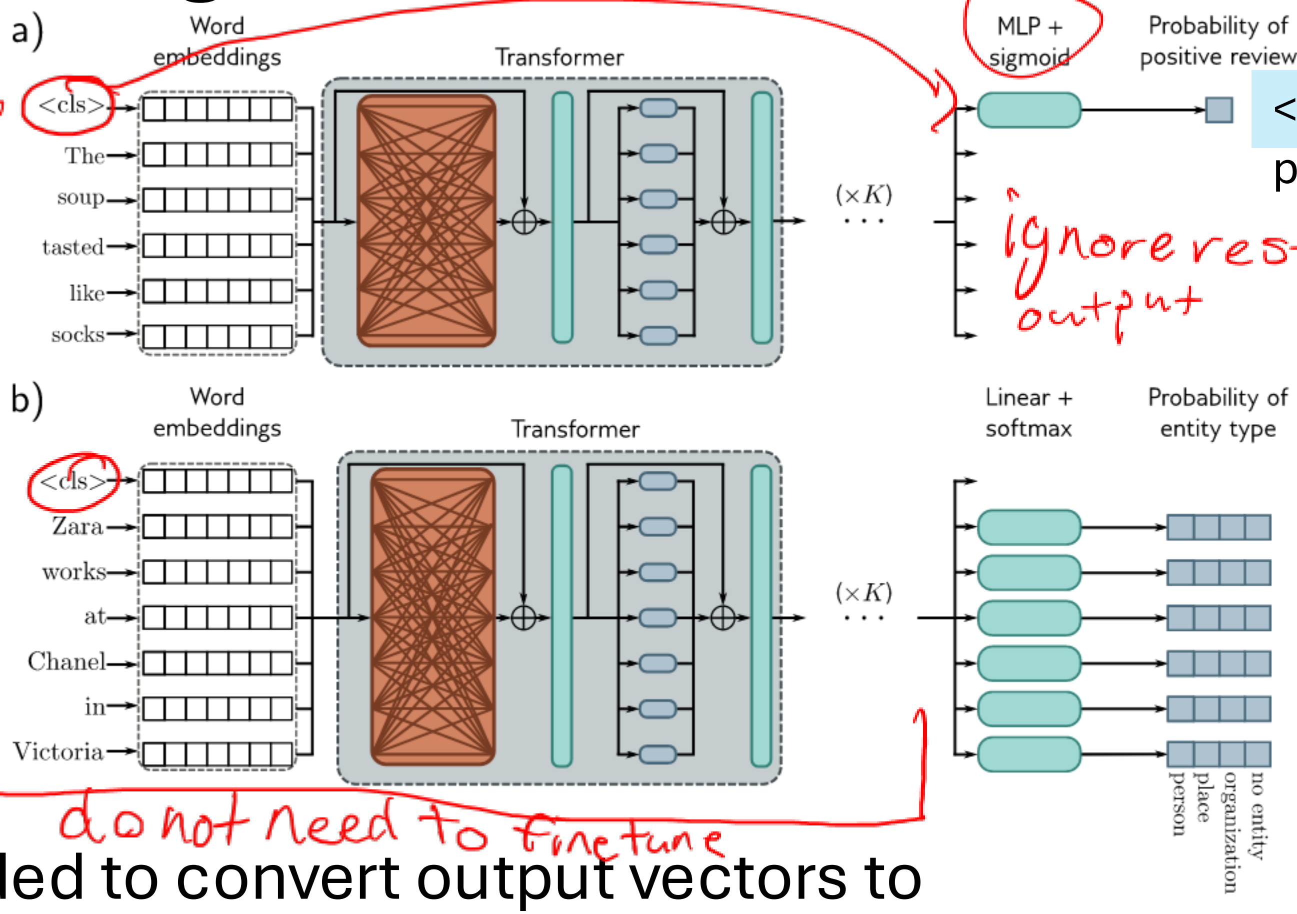
- A small percentage of input embedding replaced with a generic <mask> token
- Predict missing token from output embeddings
- Added linear layer and softmax to generate probabilities over vocabulary
- Trained on BooksCorpus (800M words) and English Wikipedia (2.5B words)

Encoder Fine-Tuning

small model to output → sigmoid or softmax

Sentiment Analysis

Named Entity Recognition (NER)



<cls> token position

- Extra layer(s) appended to convert output vectors to desired output format
- 3rd Example: Text span prediction -- predict start and end location of answer to a question in passage of Wikipedia, see https://rajpurkar.github.io/SQuAD-explorer/

Decoder Model Example: GPT3 (2020)

*<u>**G**</u>enerative <u>**P**</u>re-trained <u>**T**</u>ransformer*

- One purpose: *generate the next token in a sequence*
- By constructing an autoregressive model

T. B. Brown *et al.*, "Language Models are Few-Shot Learners." arXiv, Jul. 22, 2020. doi: 10.48550/arXiv.2005.14165.

Decoder Model Example: GPT3 (2020)

Generative Pre-trained Transformer

- One purpose: *generate the next token in a sequence*
- By constructing an autoregressive model
- Factors the probability of the sentence:

$$
\begin{aligned}
\Pr(\mathit{Learning\ deep\ learning\ is\ fun}) = \\
&\Pr(\mathit{Learning}) \times \Pr(\mathit{deep} \mid \mathit{learning}) \quad \times \\
&\Pr(\mathit{learning} \mid \mathit{Learning\ deep}) \times \\
&\Pr(\mathit{is} \mid \mathit{Learning\ deep\ learning}) \times \\
&\Pr(\mathit{fun} \mid \mathit{Learning\ deep\ learning\ is})
\end{aligned}
$$

T. B. Brown *et al.*, "Language Models are Few-Shot Learners." arXiv, Jul. 22, 2020. doi: 10.48550/arXiv.2005.14165.

Decoder Model Example: GPT3 (2020)

***G**enerative **P**re-trained **T**ransformer*

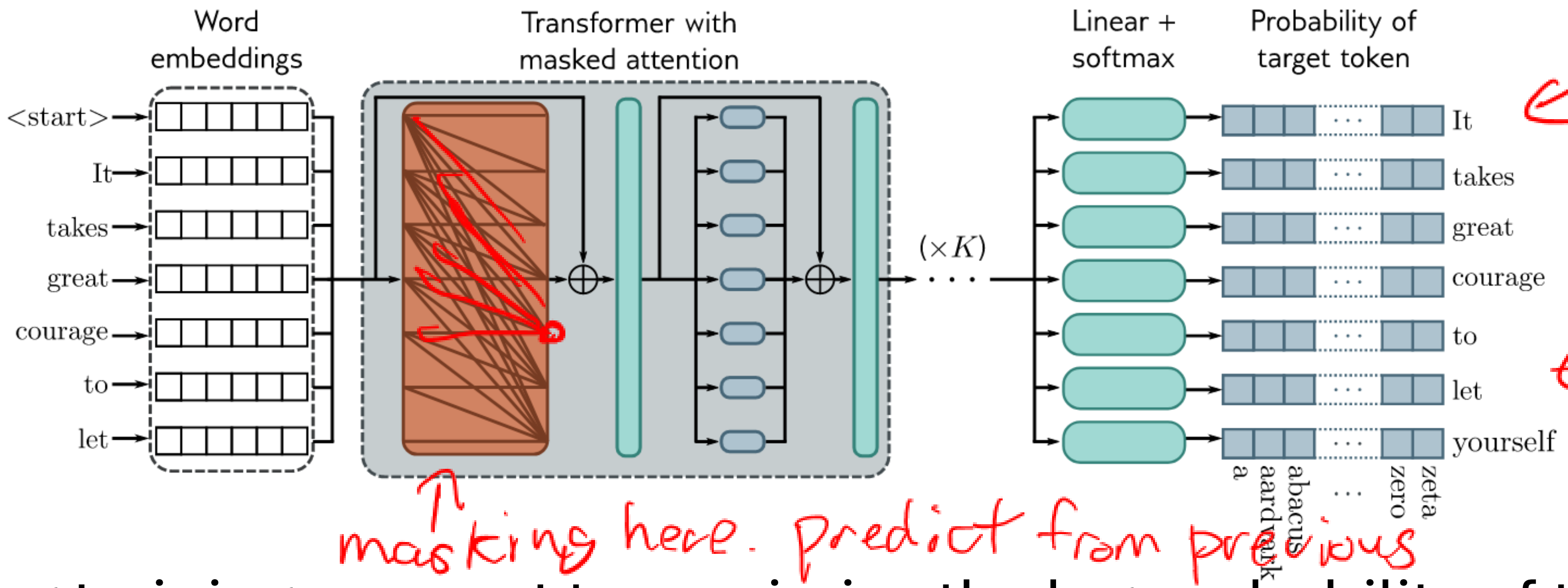
- One purpose: *generate the next token in a sequence*
- By constructing an autoregressive model
- Factors the probability of the sentence:
  $\mathrm{Pr}(Learning\ deep\ learning\ is\ fun) =$
  $\mathrm{Pr}(Learning) \times \mathrm{Pr}(deep \mid learning) \times$
  $\mathrm{Pr}(learning \mid Learning\ deep) \times$
  $\mathrm{Pr}(is \mid Learning\ deep\ learning) \times$
  $\mathrm{Pr}(fun \mid Learning\ deep\ learning\ is)$
- More formally: Autoregressive model

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

T. B. Brown *et al.*, "Language Models are Few-Shot Learners." arXiv, Jul. 22, 2020. doi: 10.48550/arXiv.2005.14165.

→ not same masking

Decoder: *Masked* Self-Attention



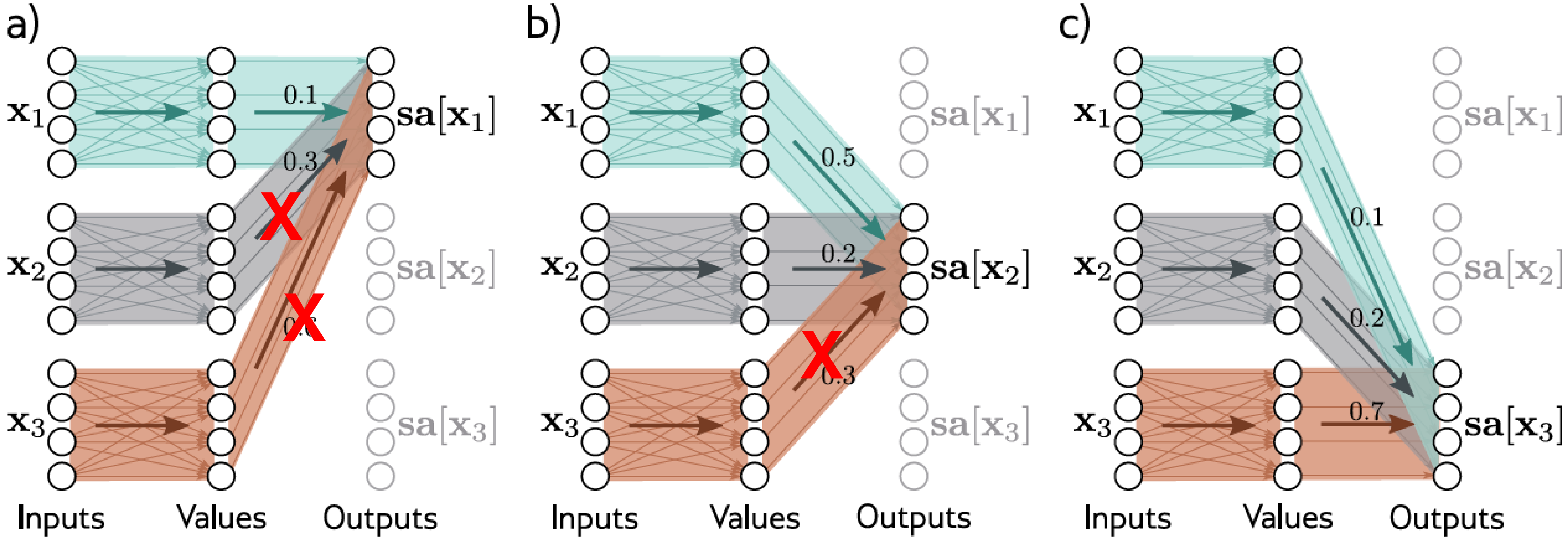
probability of 1st token

probabilities of next token conditionned on previous tokens.

masking here. predict from previous

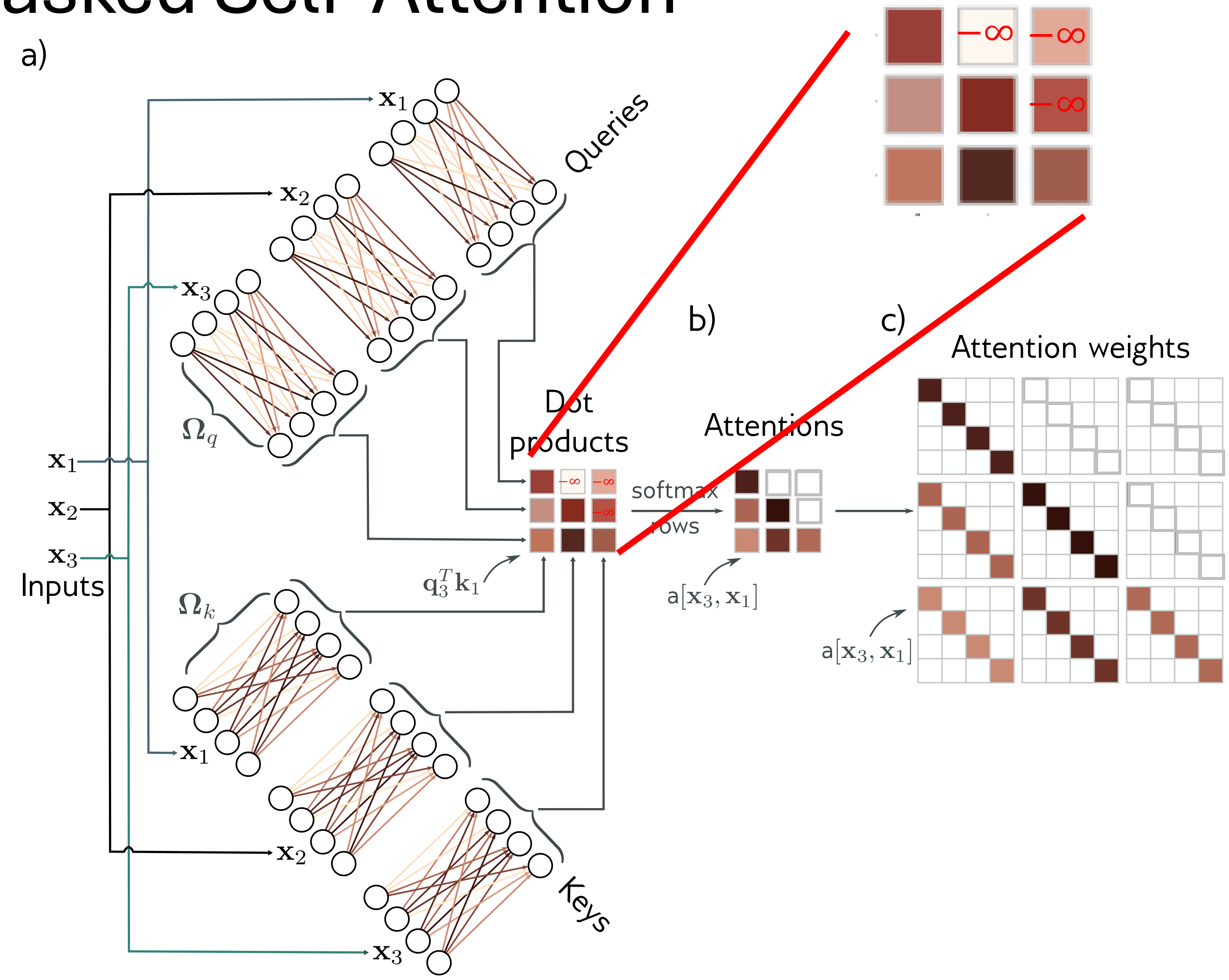
- During training we want to maximize the log probability of the input text under the autoregressive model.
- We want to make sure the model doesn't "cheat" during training by looking ahead at the next token.
- Hence, we mask the self attention weights corresponding to current and right context to *negative infinity*.

Masked Self-Attention

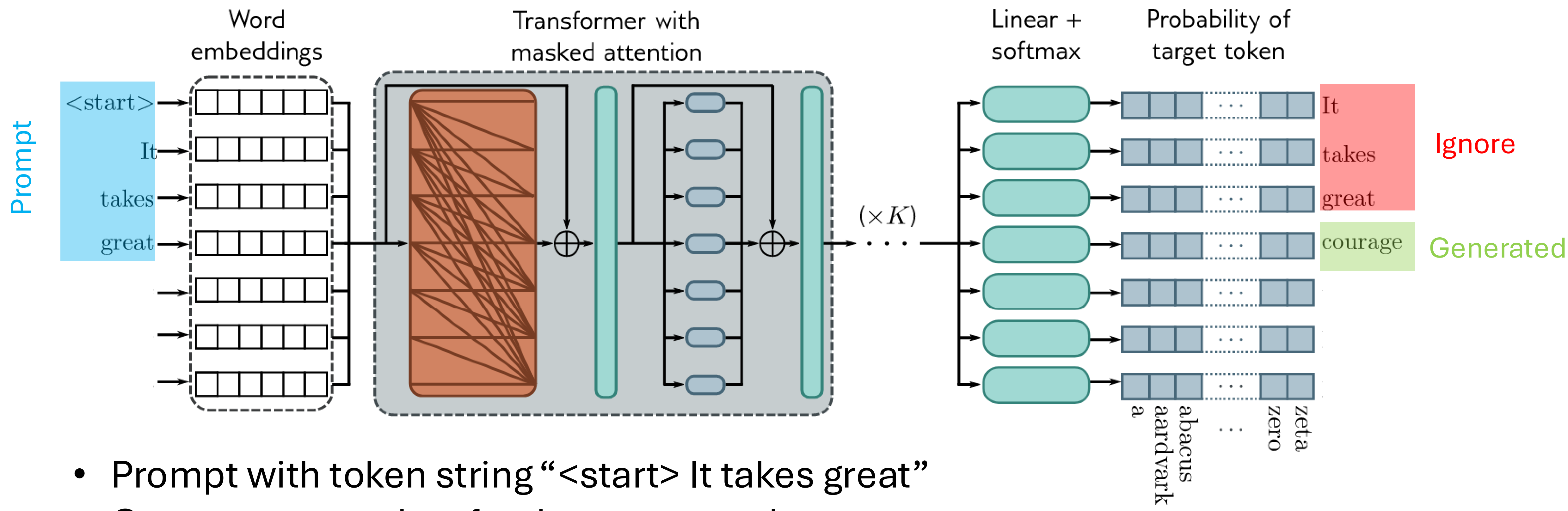


Mask right context self-attention weights to zero

Masked Self-Attention



Decoder: Text Generation (Generative AI)



- Prompt with token string "<start> It takes great"
- Generate next token for the sequence by
  - picking most likely token
  - sample from the probability distribution
    - alternative *top-k* sampling to avoid picking from the long tail
  - beam search – select the most likely sentence rather than greedily pick

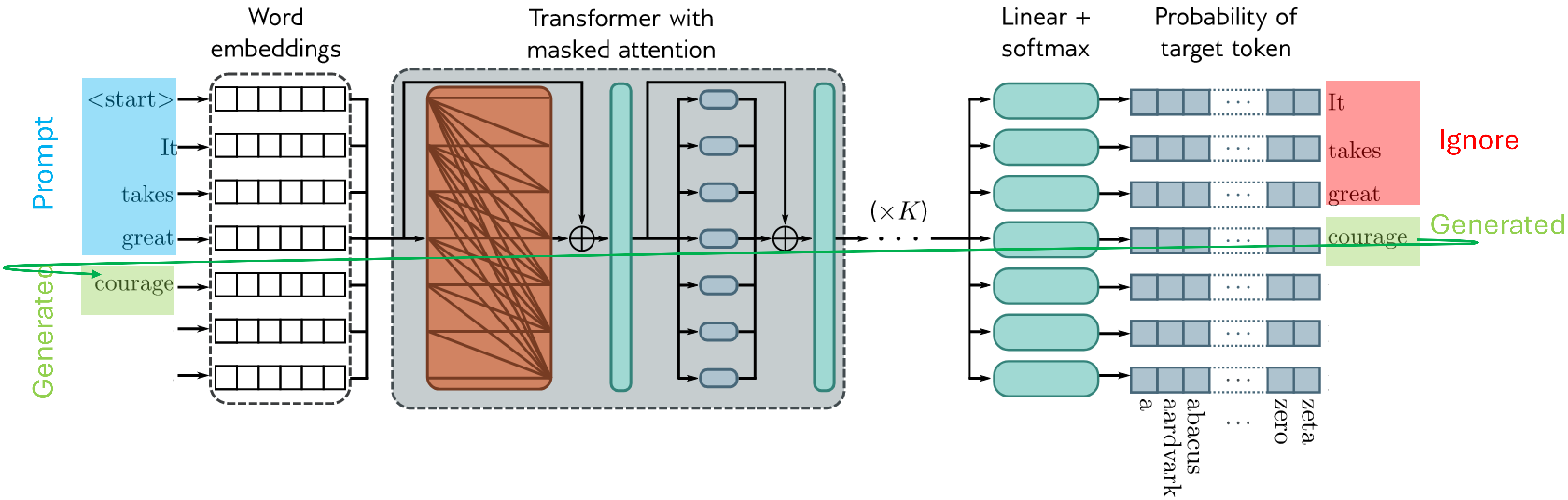
Dummy's Guide to LLM Sampling

- https://rentry.co/samplers

- Will talk about this more next time.

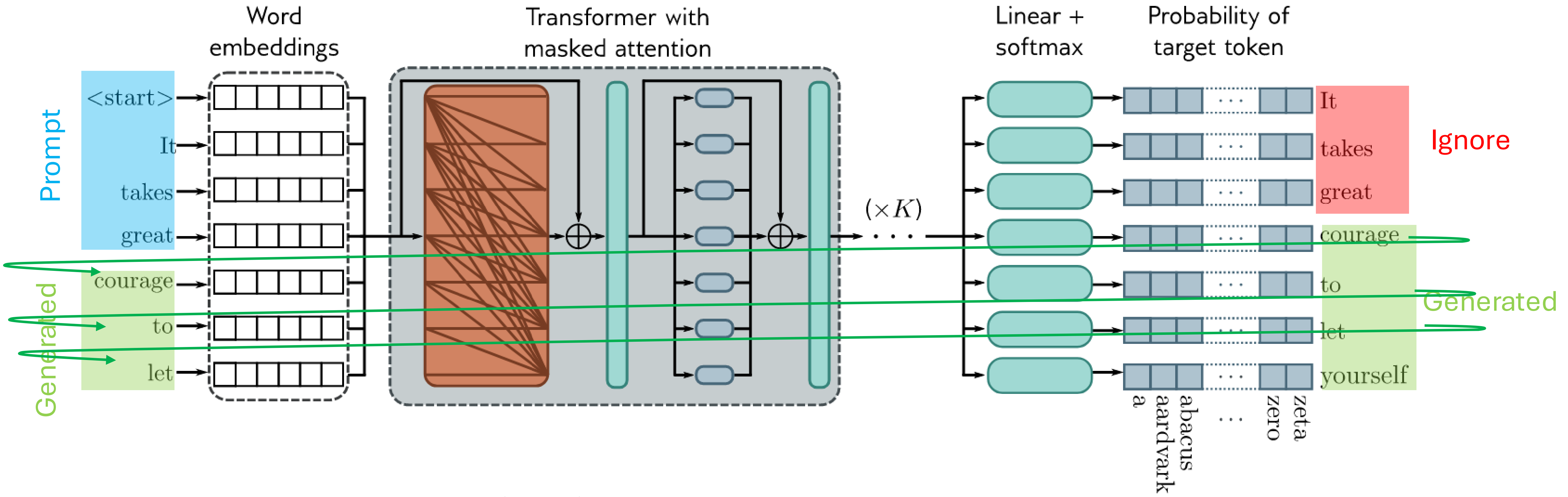
Dummy's Guide to Modern LLM Sampling

Decoder: Text Generation (Generative AI)



- Feed the output back into input

Decoder: Text Generation (Generative AI)



- Feed the output back into input

Technical Details

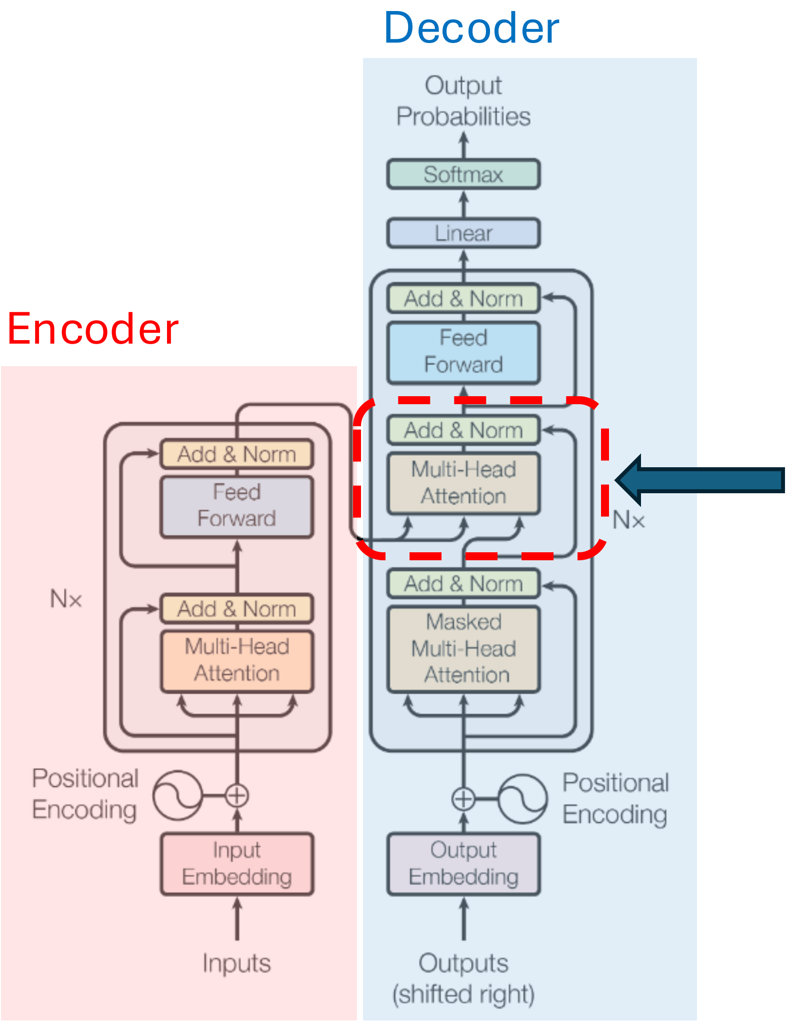
	BERT	GPT3
Model Architecture	Encoder	Decoder
Embedding Size	1024	12,288
Vocabulary	30K tokens	
Sequence Length		2048
# Heads	16	96
# Layers	24	96
Q,K,V dimensions	64	128
Training set size	3.3B tokens	300B+ tokens
# Parameters	340M	175B

Encoder-Decoder Model

- Used for *machine translation*, which is a *sequence-to-sequence* task

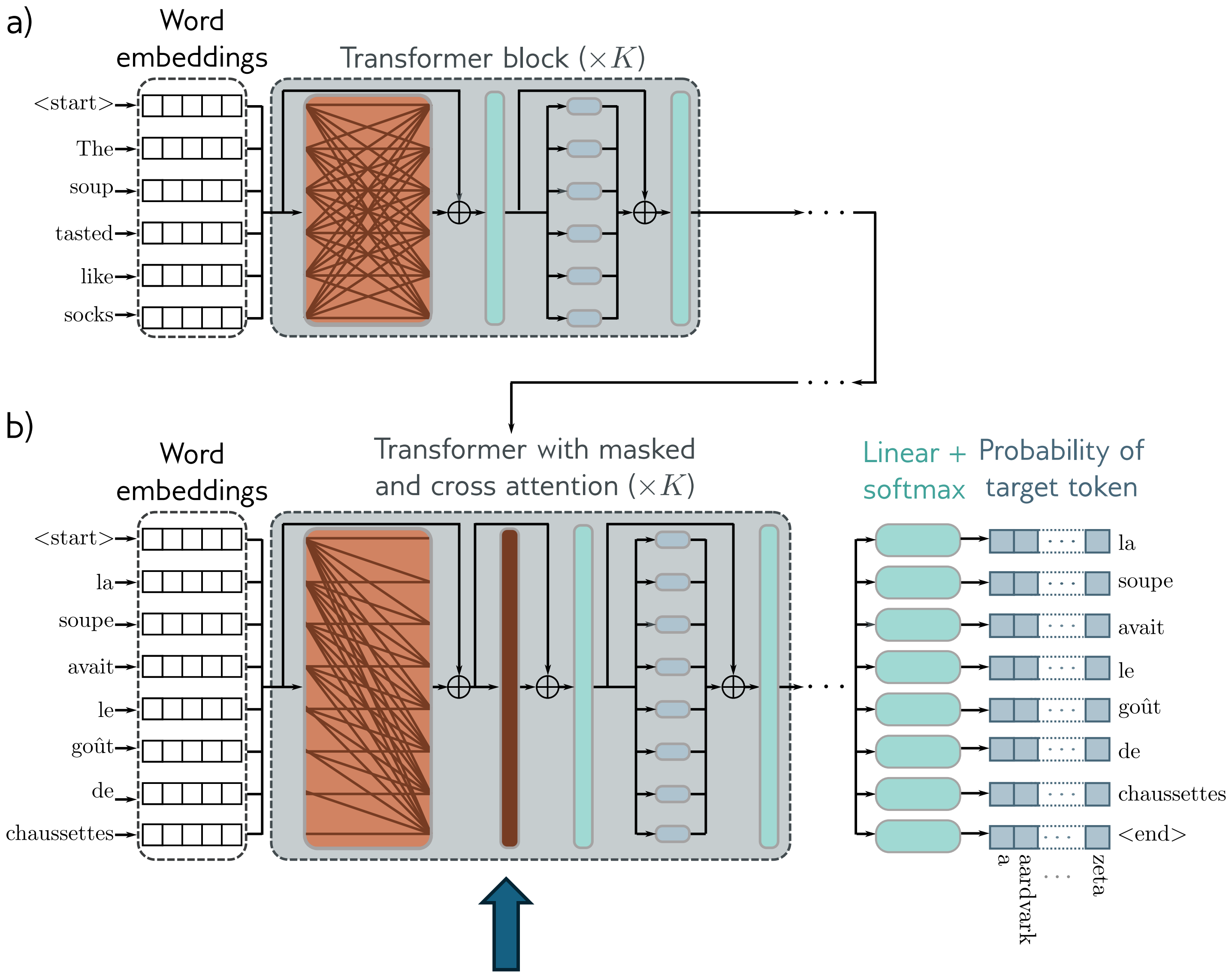
https://jalammar.github.io/illustrated-transformer/

Encoder Decoder Model



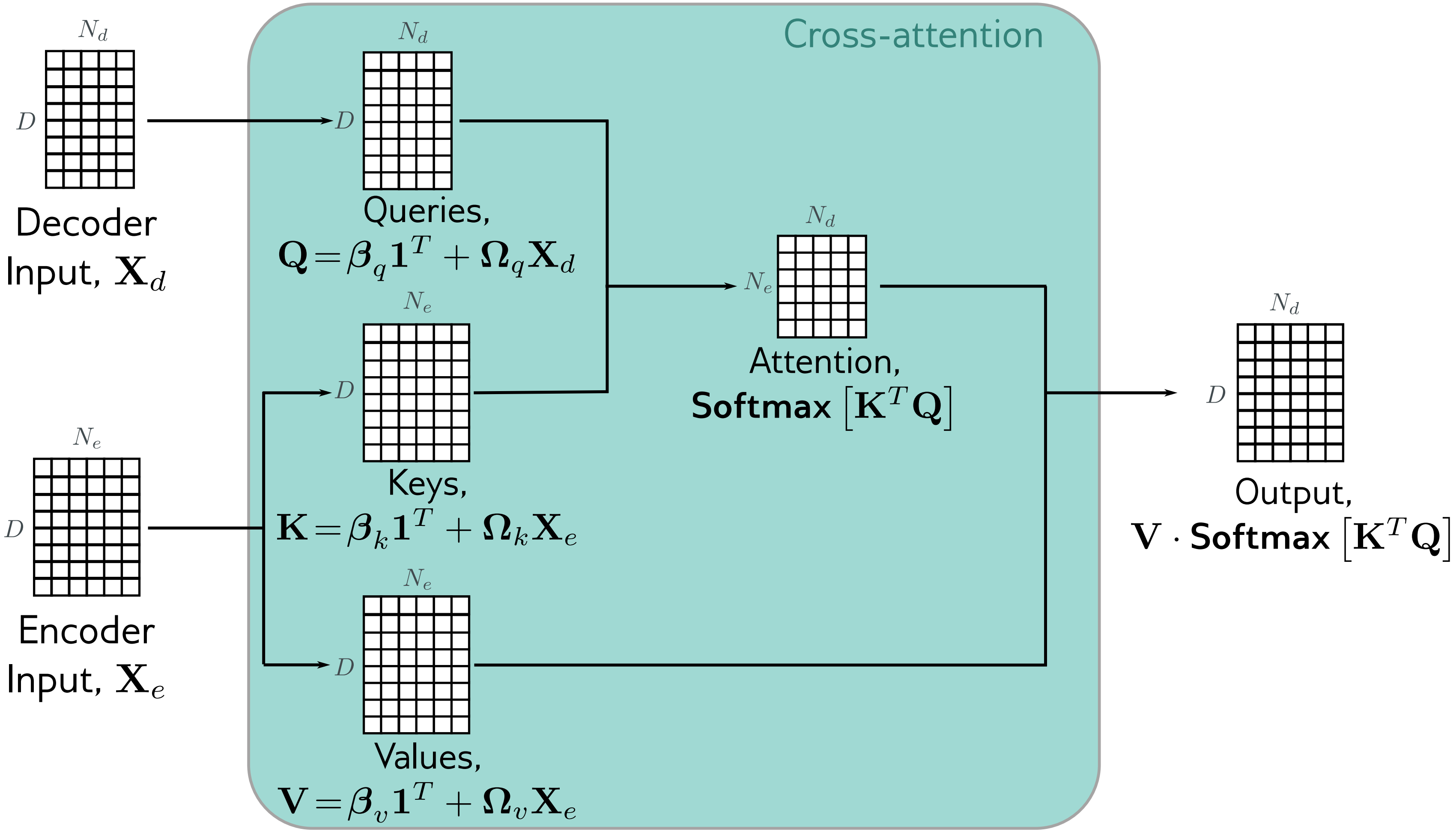
- The transformer layer in the decoder of the encoder-decoder model has an extra stage
- Attends to the input of the encoder with *cross attention* using Keys and Values from the output of the encoder
- Shown here on original diagram from "Attention is all you need" paper

Encoder Decoder Model



- Same view per UDL book

Cross-Attention



Keys and Values come from the last stage of the encoder

Any Questions?

???

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations