Transformers Part 2

Antoine Bosselut

EPFL

nlp lab

Tuesday 11th March

9am to 6:30pm in BC

for IC students in master or third year bachelor

>> SCAN

or go to clic.epfl.ch/icbd

CLIC EPFL Faculté informatique et communications EPIC EPFL PhDs of IC

Une association reconnue par l'EPFL
An association recognized by the EPFL

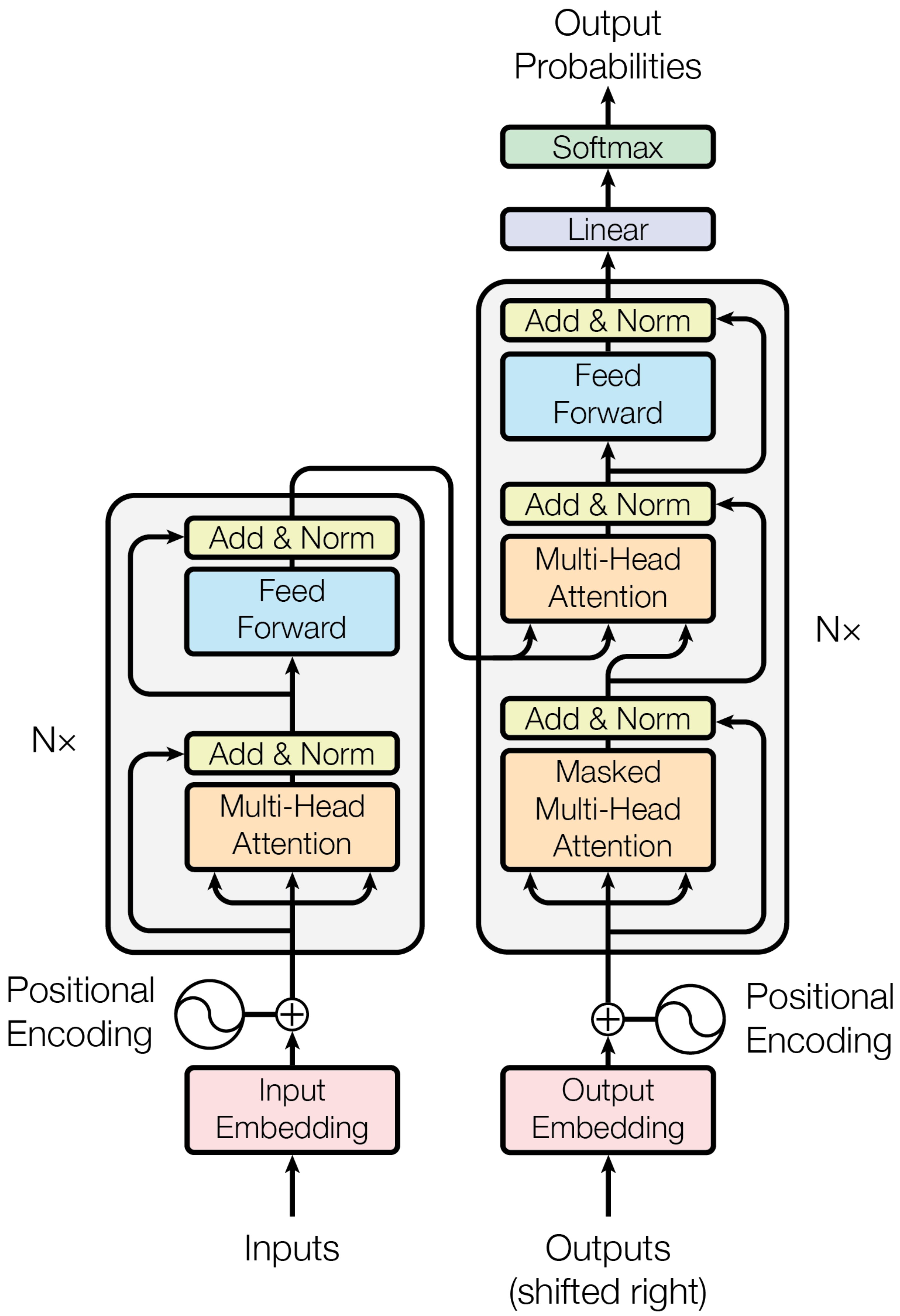
Today's Outline

- **Lecture**
  - **Recap:** Transformers
  - **Decoding**
  - **Supercharging Transformers:** Pretraining, GPT
- **Exercise Session**
  - **Review of Week 2**
  - **Week 3:** Sequence-to-sequence models, transformers + decoding

Some slides adapted from Mohit Iyyer, Emma Strubell

Full Transformer

- Made up of encoder and decoder, each with multiple cascaded transformer blocks
  - slightly different architecture in encoder and decoder transformer blocks
- Blocks generally made up **multi-headed attention** layers (self-attention) and **feedforward** layers
- No recurrent computations! **Encode sequences with self-attention**
  - Position embeddings provide sequence order information to transformer



(Vaswani et al., 2017)

Self-attention (in encoder)

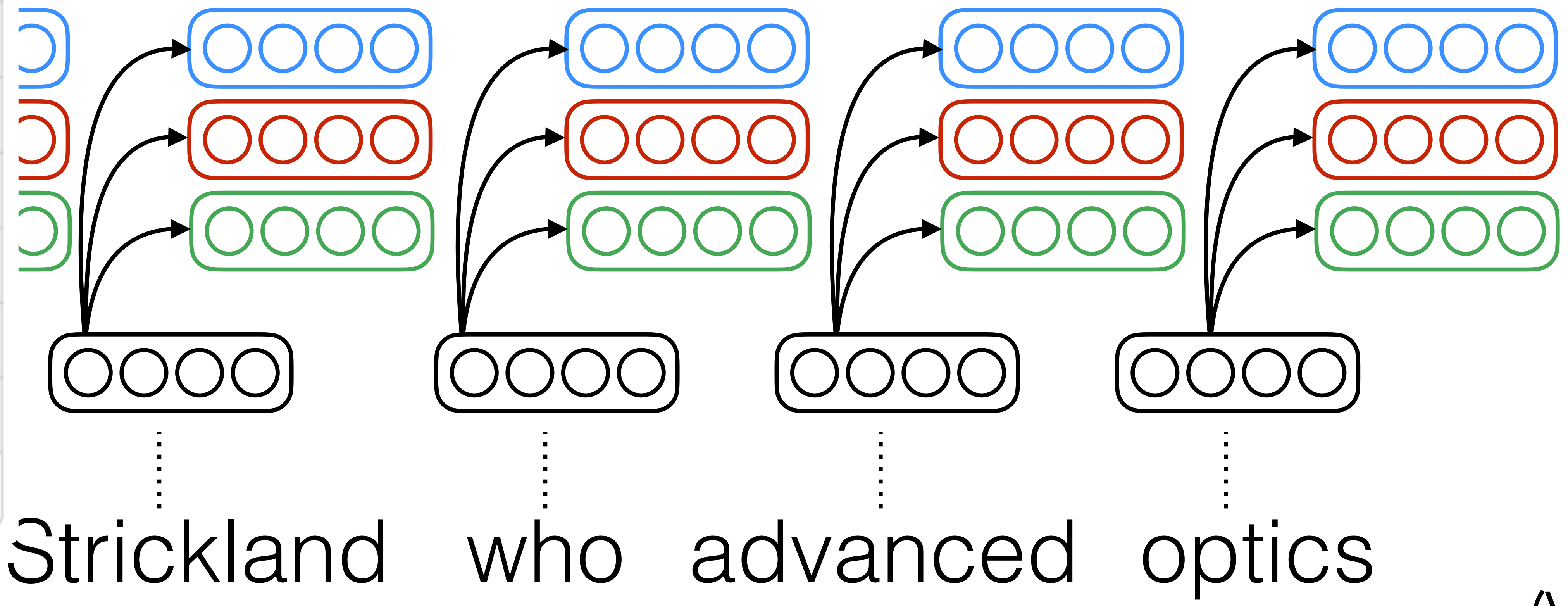
STEP 1: tokenizat° = split sentences in tokens

STEP 2: convert each token into a vector representat° using an embedding layer
(just like in word2vec / Glove) => embedding matrix, $X \in \mathbb{R}^{N \times d}$ ← embedding size
↖ # tokens in sentence

STEP 3: creating Q, K, V => $Q = XW^Q$, $K = XW^K$, $V = XW^V$ → actual info content of the word
→ what this word offers to others
→ what this word is looking for in other words

STEP 4: attent° _scores = $QK^T \in \mathbb{R}^{N \times N}$ => each value in matrix tells how much attent° one word should pay to another
Scale the scores to avoid large values from dominating after softmax: $(QK^T)/\sqrt{d}$
attent°_weights = softmax $(QK^T/\sqrt{d})$

Step	What Happens?
1. Tokenization	Split sentence into tokens
2. Word Embeddings	Convert tokens into vectors
3. Q, K, V Matrices	Project embeddings into query, key, and value vectors using learnable weights
4. Attention Scores	Compute QK^T to measure relationships between words
5. Softmax	Normalize scores into probabilities
6. Weighted Sum	Multiply attention weights by V to get output
7. Multi-Head Attention	Repeat the process in parallel, concatenate outputs
8. Add & Normalize	Residual connection and layer normalization
9. Feedforward Network	Pass through a fully connected layer



(Vaswani et al., 2017)

Self-attention (in encoder)

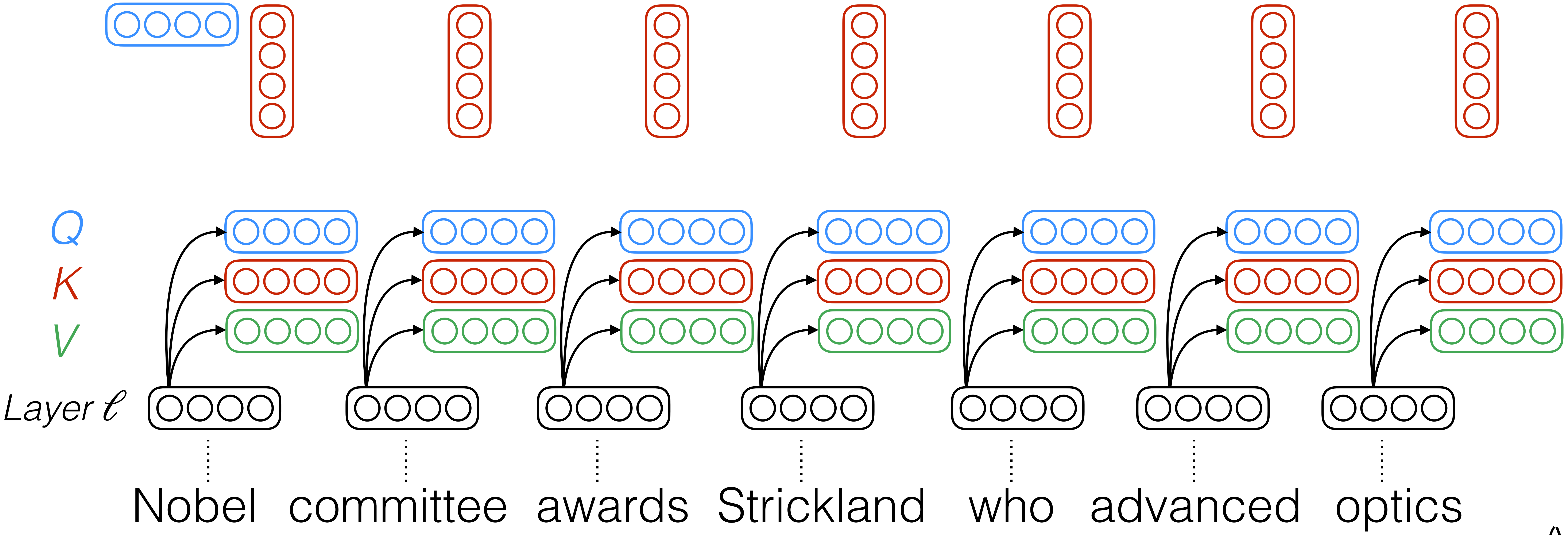
$= W^Q x_t$ $\quad$ $= W^K X$

$$\mathbf{a_t} = \frac{(\mathbf{W}^Q Q_t)(\mathbf{W}^K K)}{\sqrt{d}}$$

how is this equivalent to $A = \frac{QK^T}{\sqrt{d}}$

Keys *K* & values *V* are the same at every time step: Projected token representations

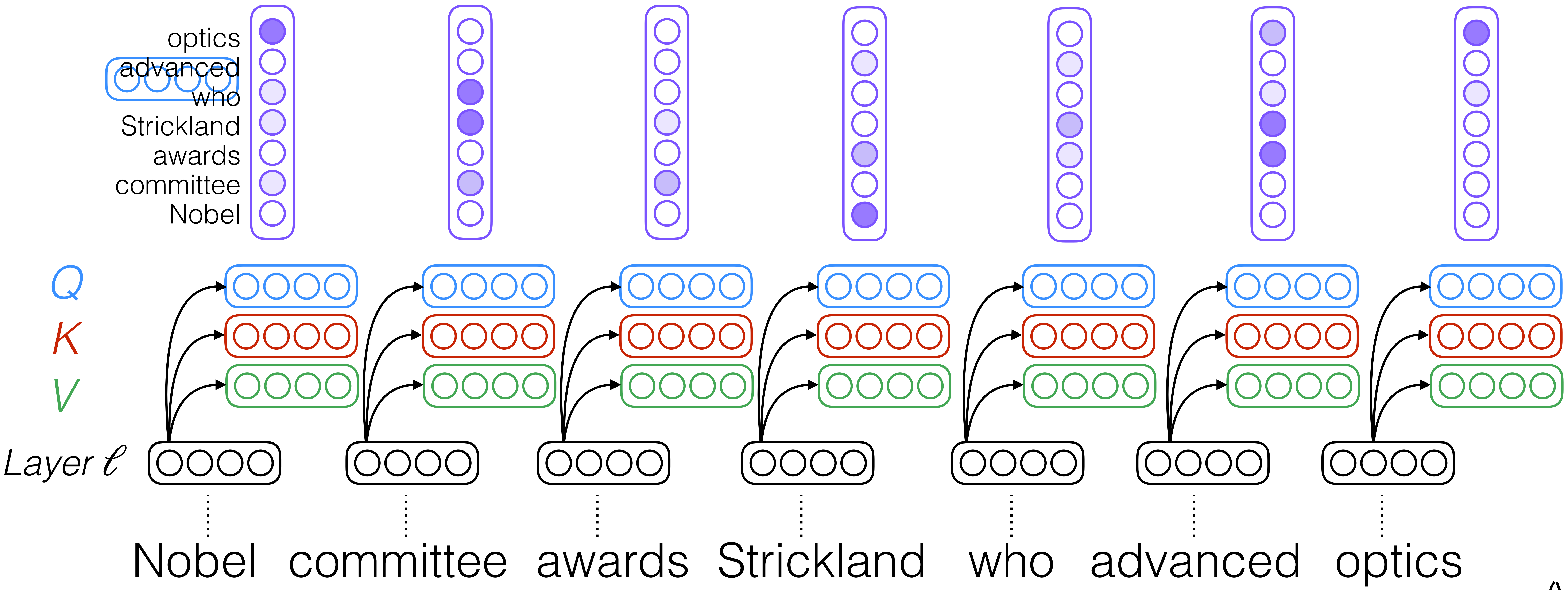
Query Q_t changes at every time step since the current token serves as the query



(Vaswani et al., 2017)

Self-attention (in encoder)

$$\mathbf{a_t} = \frac{(\mathbf{W}^Q Q_t)(\mathbf{W}^K K)}{\sqrt{d}} \qquad \alpha_t = \textbf{softmax}(\mathbf{a_t})$$

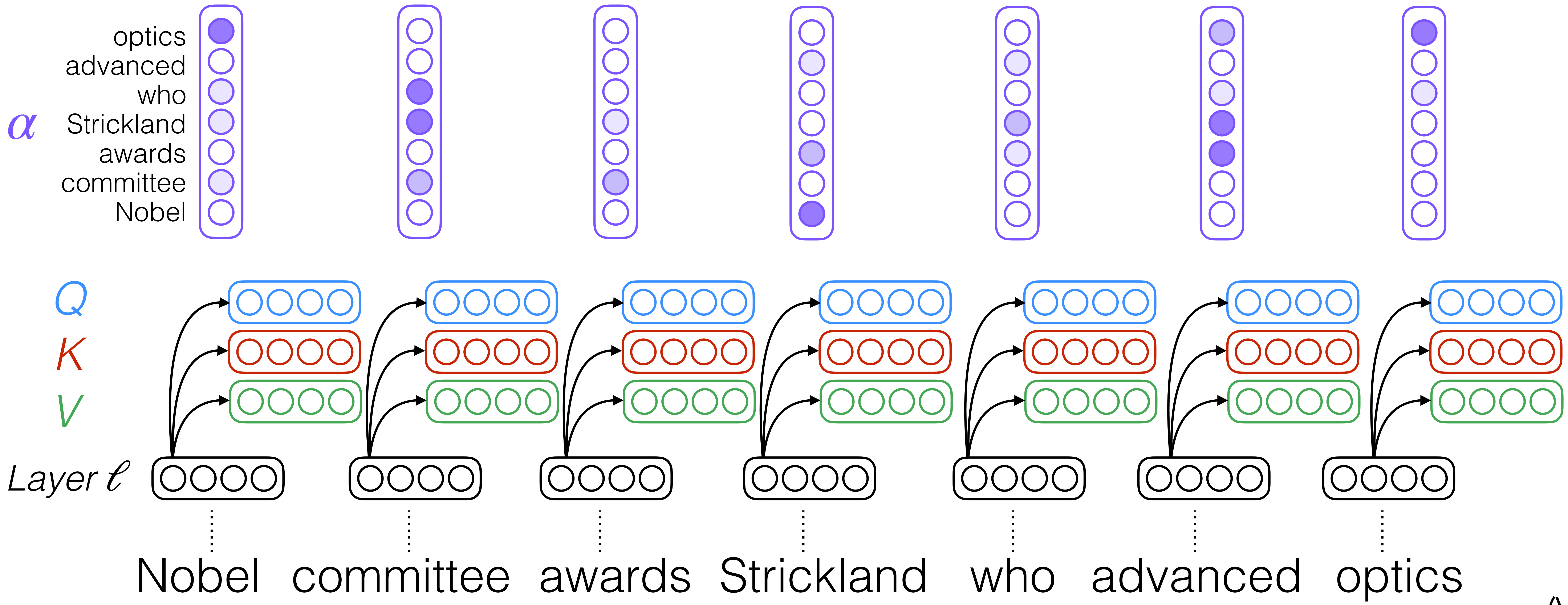


(Vaswani et al., 2017)

Self-attention (in encoder)

$$\mathbf{a_t} = \frac{(\mathbf{W}^Q \boxed{Q_t})(\mathbf{W}^K \boxed{K})}{\sqrt{d}}$$

$$\boxed{\alpha_t} = \mathbf{softmax}(\mathbf{a_t})$$



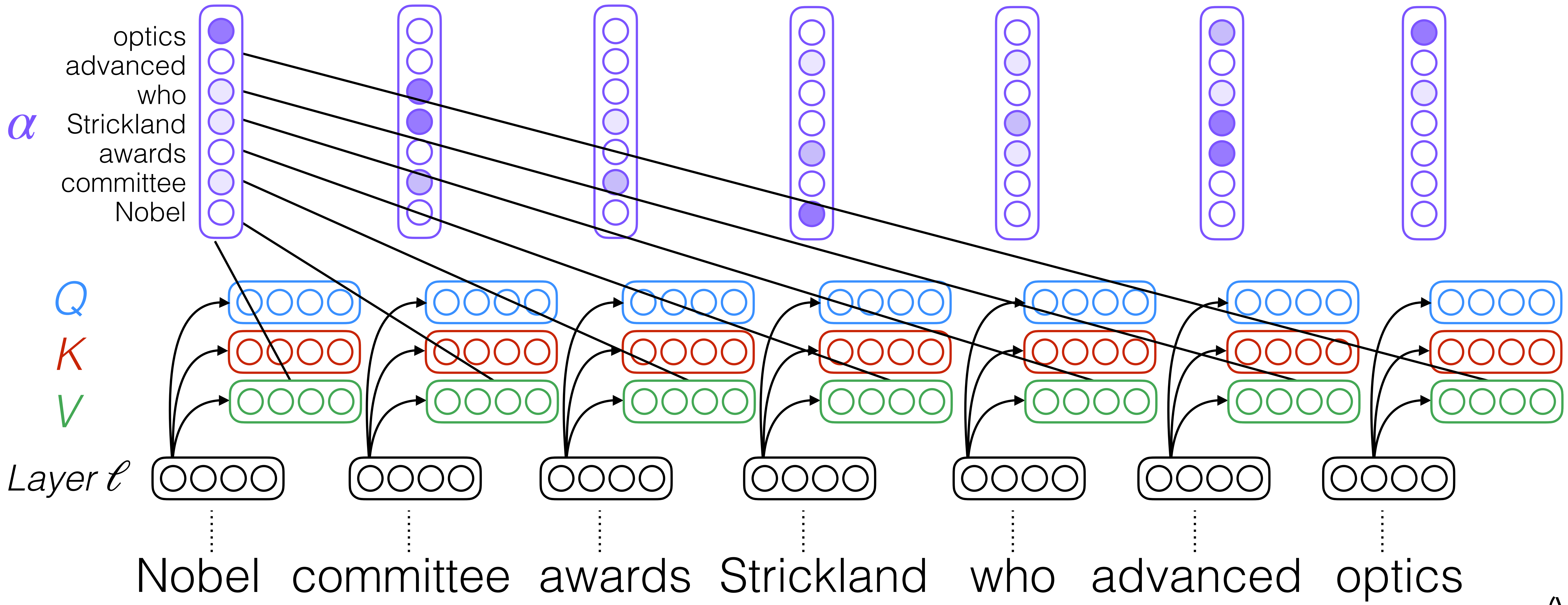
(Vaswani et al., 2017)

Self-attention (in encoder)

$$\mathbf{a_t} = \frac{(\mathbf{W}^Q Q_t)(\mathbf{W}^K K)}{\sqrt{d}}$$

$$\alpha_t = \mathbf{softmax}(\mathbf{a_t})$$

$$M_t = W^O \alpha_t (V \mathbf{W}^V)$$

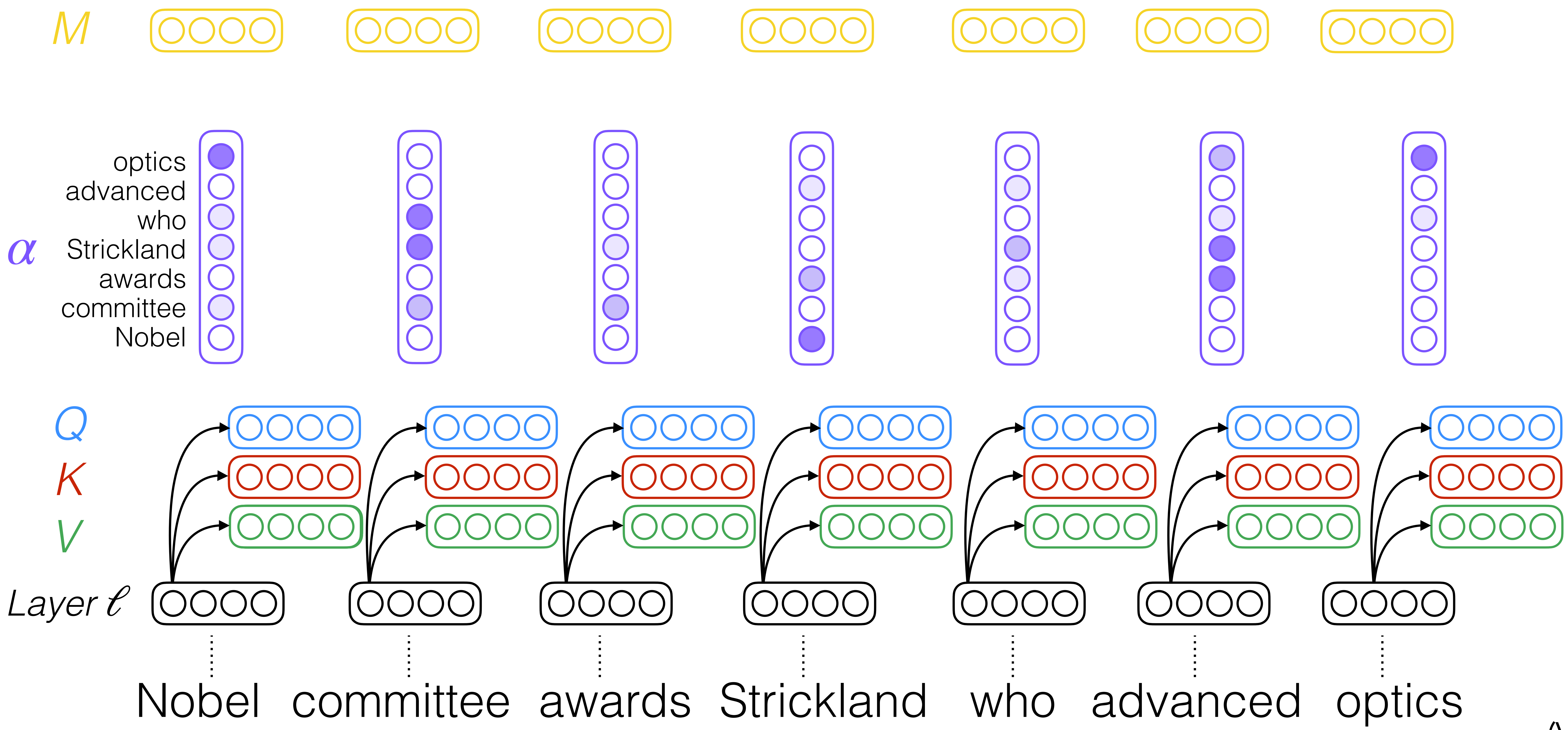


(Vaswani et al., 2017)

Self-attention (in encoder)

?!

$$\boxed{M_t} = W^O \boxed{\alpha_t} (\boxed{V}\mathbf{W}^V)$$

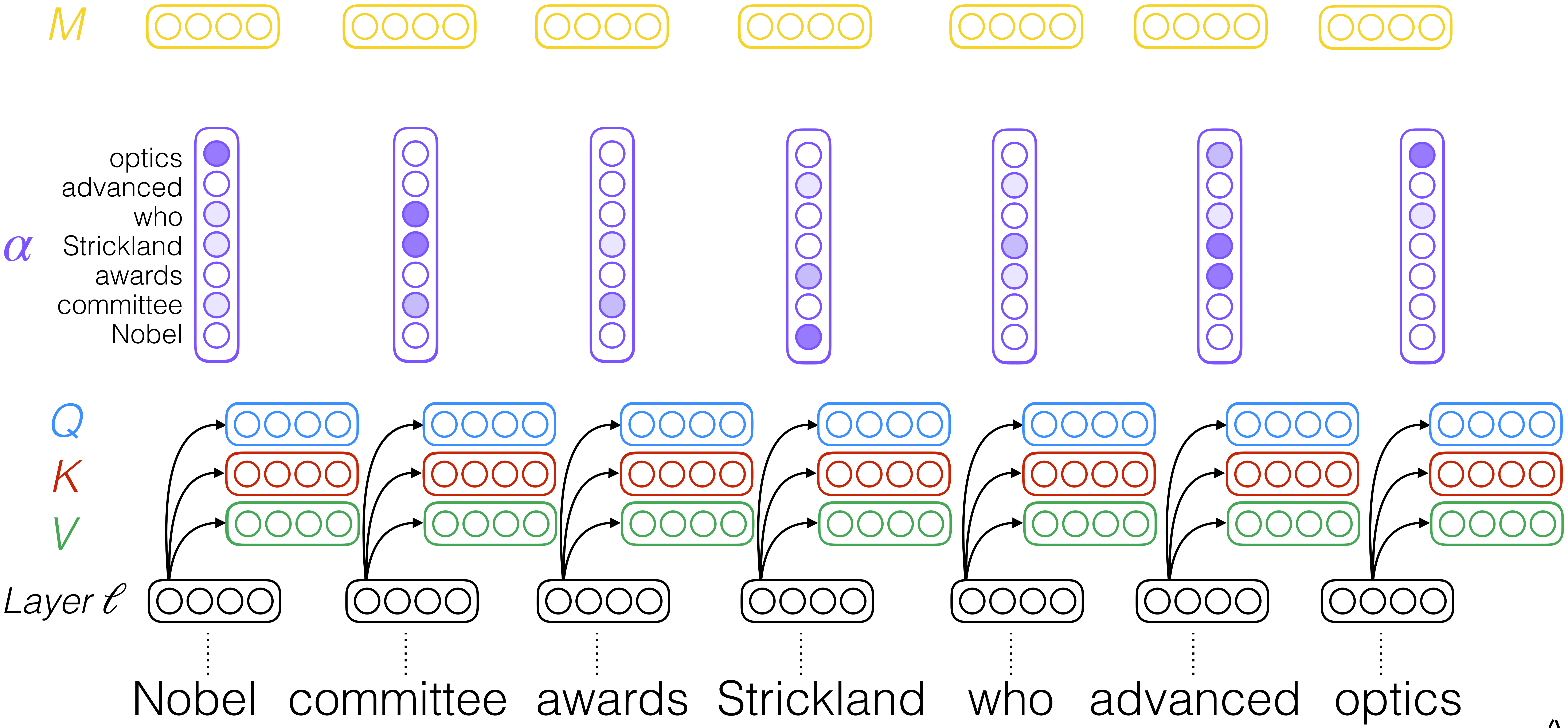


(Vaswani et al., 2017)

Self-attention (in encoder)

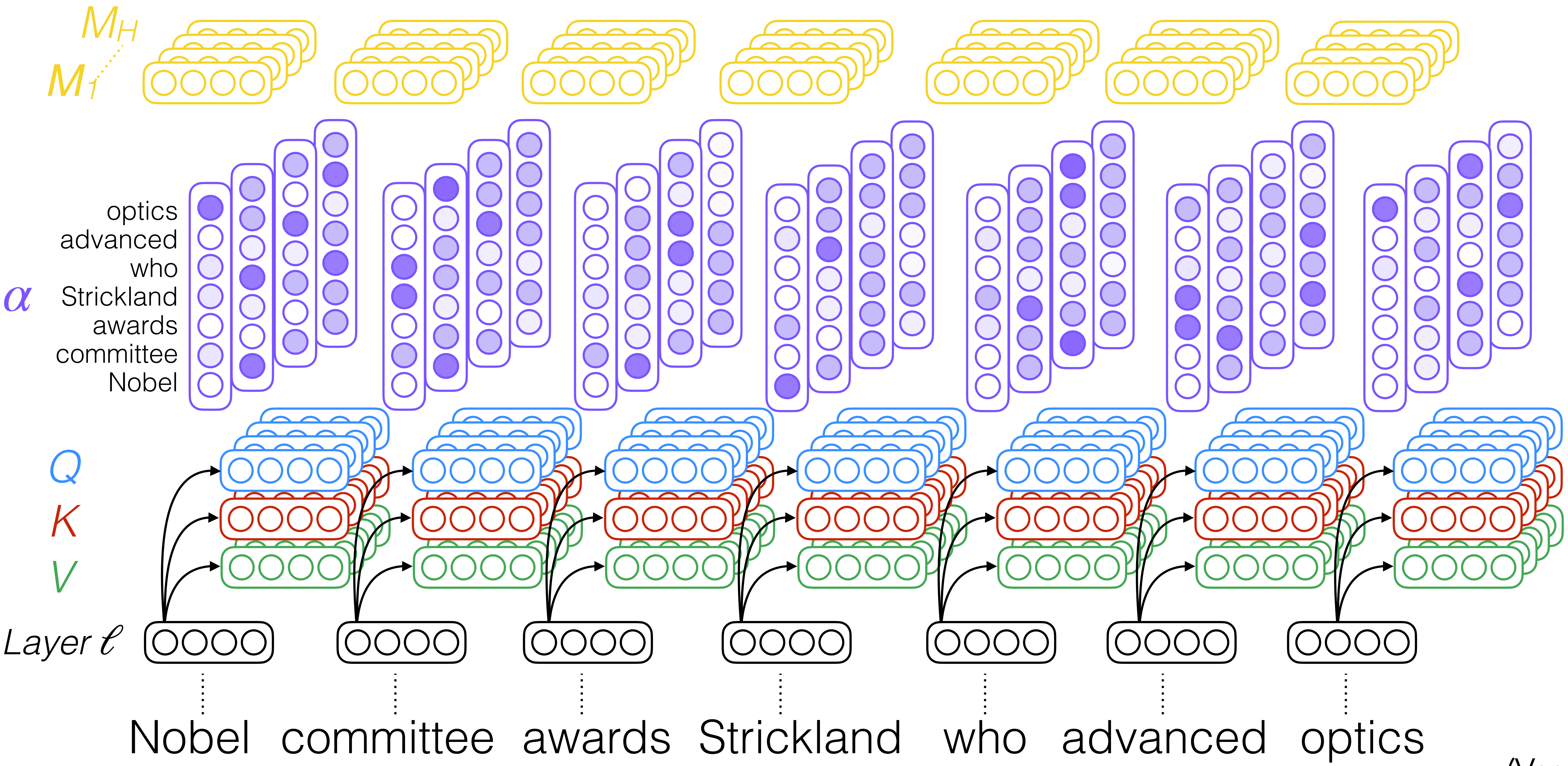
$$M_t = W^O \alpha_t (V \mathbf{W}^V)$$

how is this equal to
output = αV



(Vaswani et al., 2017)

$$\mathbf{a}_{h,t} = \frac{(\mathbf{W}_h^Q Q_t)(\mathbf{W}_h^K K)}{\sqrt{d/H}} \qquad \alpha_{h,t} = \textbf{softmax}(\mathbf{a}_{h,t}) \qquad M_{h,t} = \alpha_{h,t}(V\mathbf{W}_h^V)$$



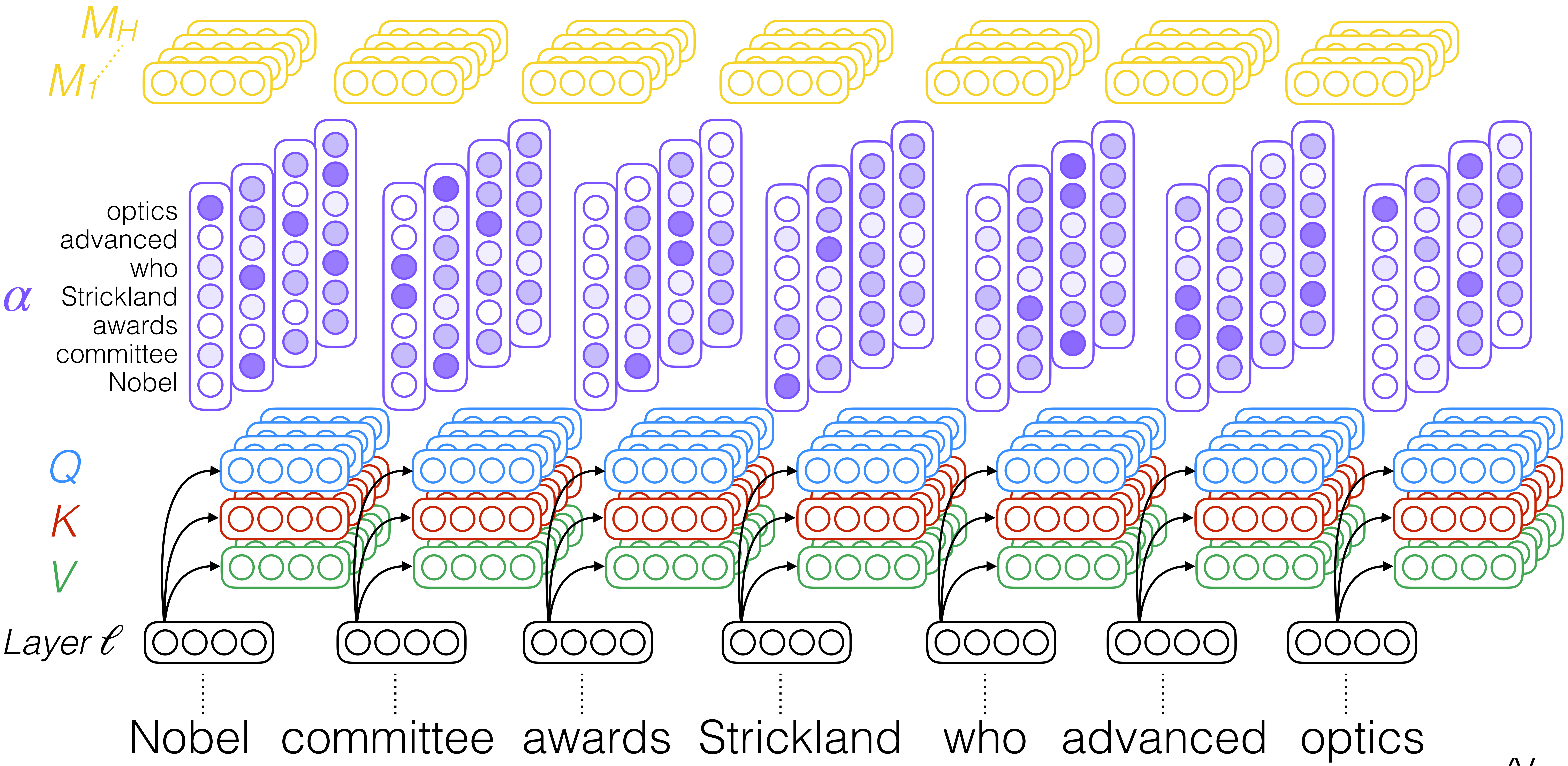
(Vaswani et al., 2017)

$$\mathbf{a}_{h,t} = \frac{(\mathbf{W}_h^Q Q_t)(\mathbf{W}_h^K K)}{\sqrt{d/H}}$$

$$\alpha_{h,t} = \textbf{softmax}(\mathbf{a}_{h,t})$$

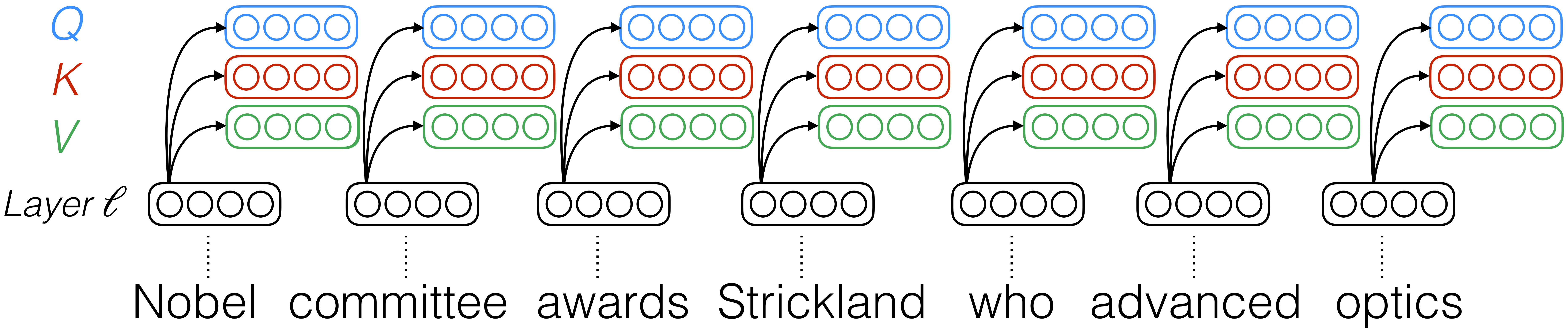
$$M_{h,t} = \alpha_{h,t}(V\mathbf{W}_h^V)$$

$$M_t = W^O[M_{1,t}; \ldots ; M_{H,t}]$$



(Vaswani et al., 2017)

Single Headed Attention:

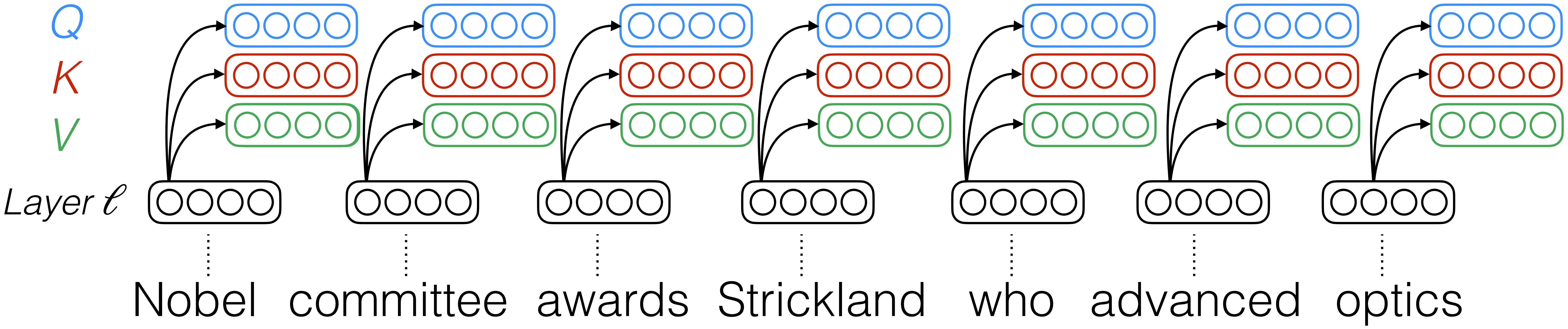


$$\mathbf{a_t} = \frac{(\mathbf{W}^Q Q_t)(\mathbf{W}^K K)}{\sqrt{d}}$$

$$\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d \times d}$$

(Vaswani et al., 2017)

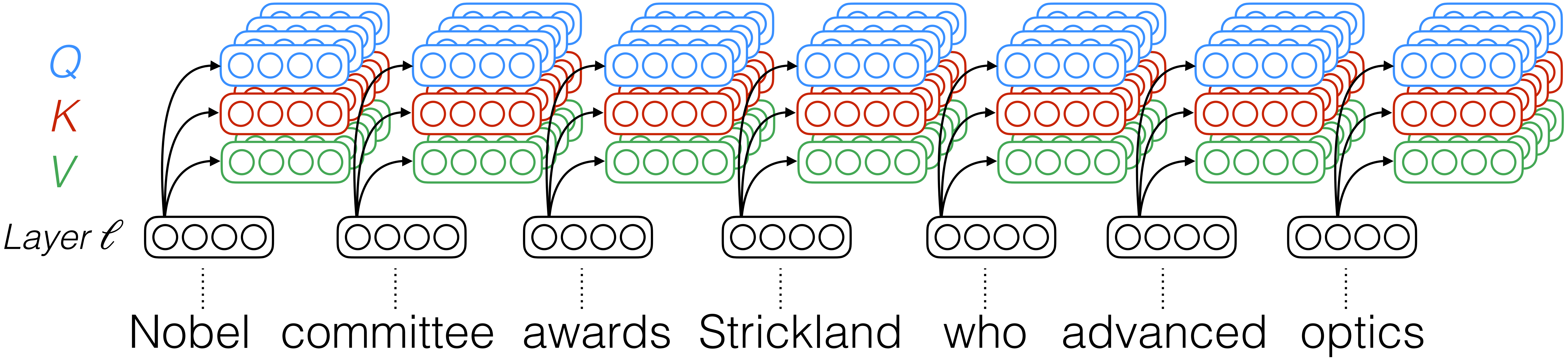
Single Headed Attention:



$$\mathbf{a}_t = \frac{(\mathbf{W}^Q Q_t)(\mathbf{W}^K K)}{\sqrt{d}}$$

$$\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d \times d}$$

Multi-headed Headed Attention:

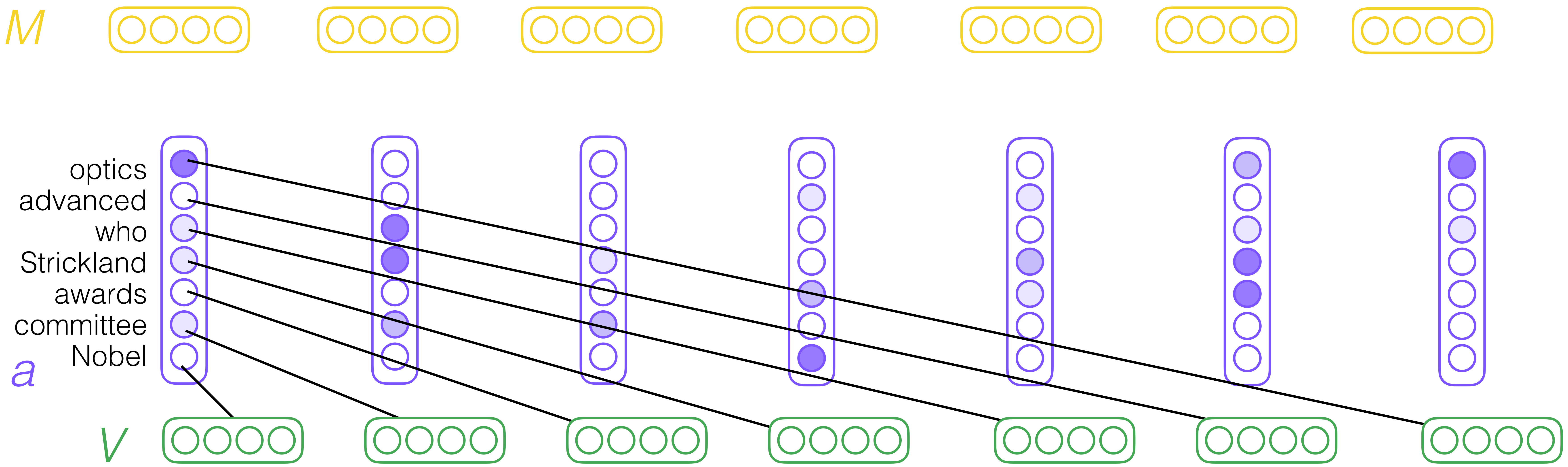


$$\mathbf{a}_{h,t} = \frac{(\mathbf{W}_h^Q Q_t)(\mathbf{W}_h^K K)}{\sqrt{d/H}}$$

$$\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{d \times d/H}$$

(Vaswani et al., 2017)

Single Headed Attention:

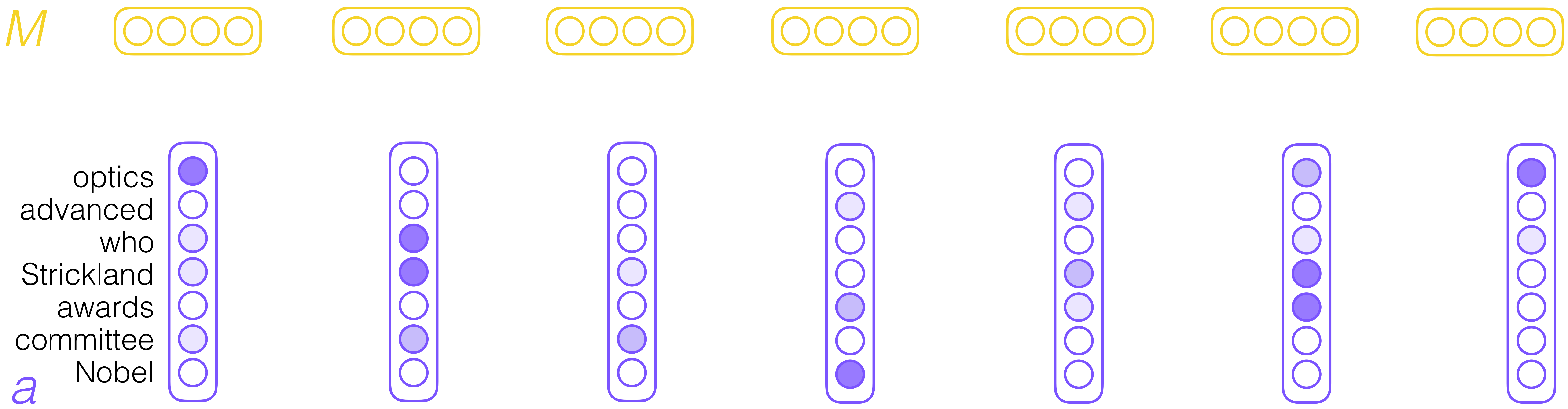


$$\boxed{M_t} = W^O \boxed{\alpha_t}(\boxed{V}\mathbf{W}^V)$$

$$\mathbf{W}^V, \mathbf{W}^O \in \mathbb{R}^{d \times d}$$

(Vaswani et al., 2017)

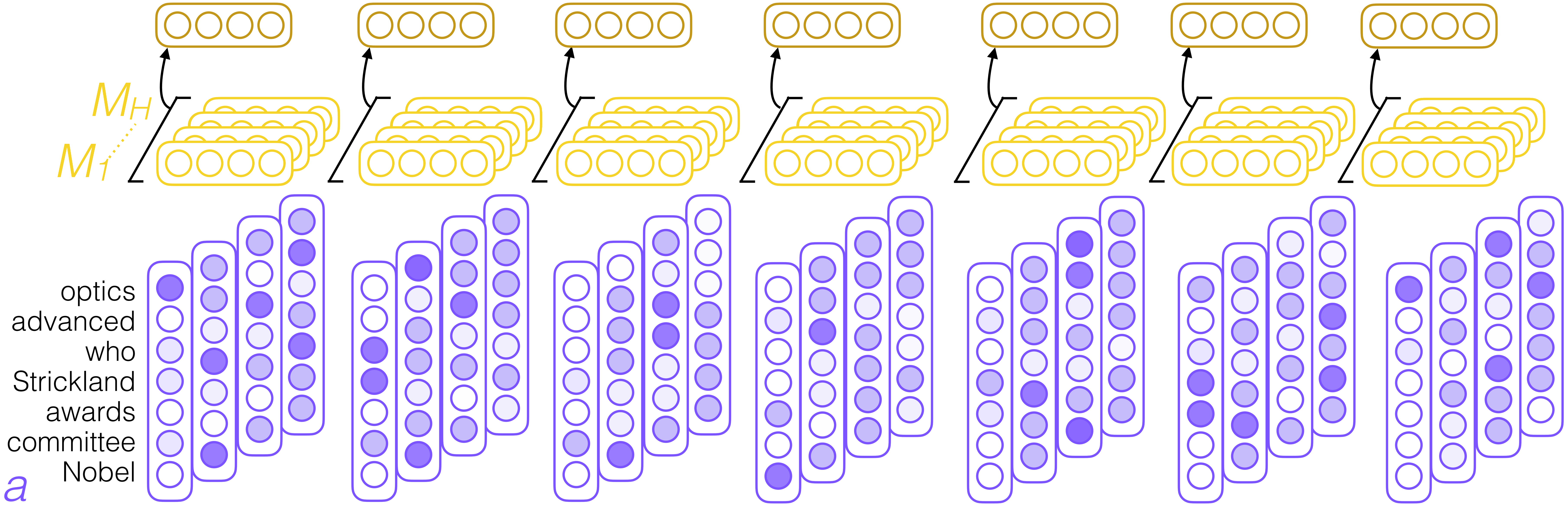
Single Headed Attention:



$$M_t = W^O \alpha_t (V \mathbf{W}^V)$$

$$\mathbf{W}^V, \mathbf{W}^O \in \mathbb{R}^{d \times d}$$

Multi-headed Headed Attention:



$$M_{h,t} = \alpha_{h,t} (V \mathbf{W}_h^V)$$

$$M_t = W^O [M_{1,t}; \ldots; M_{H,t}]$$

$$\mathbf{W}_h^V \in \mathbb{R}^{d \times d/H}$$

$$\mathbf{W}^O \in \mathbb{R}^{d \times d}$$

(Vaswani et al., 2017)

Question

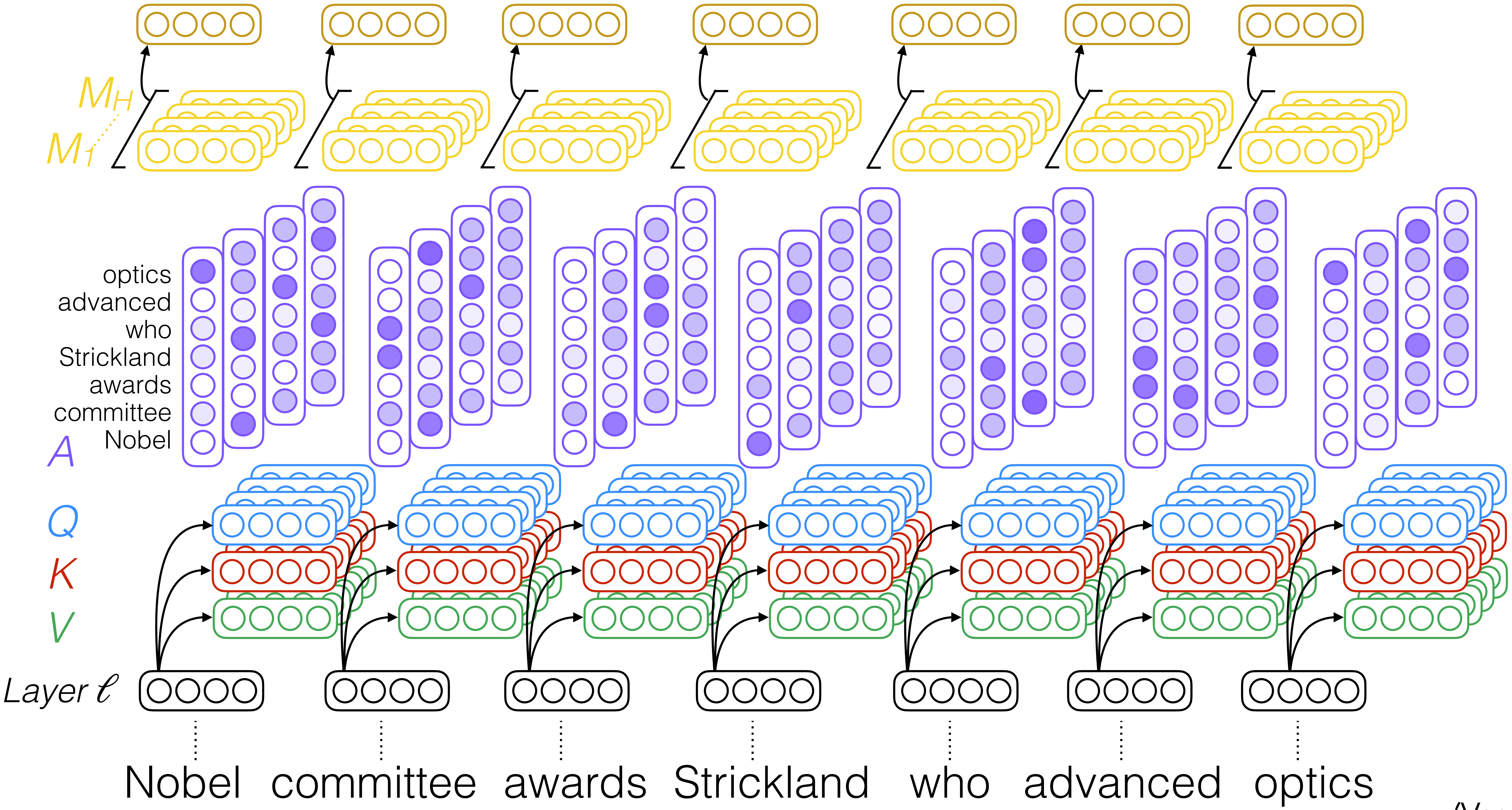
What are the learnable parameters in these matrices?

$$\mathbf{a}_{h,t} = \frac{(\mathbf{W}_h^Q Q_t)(\mathbf{W}_h^K K)}{\sqrt{d/H}}$$

$$\alpha_{h,t} = \textbf{softmax}(\mathbf{a}_{h,t})$$

$$M_{h,t} = \alpha_{h,t}(V\mathbf{W}_h^V)$$

$$M_t = W^O[M_{1,t}; \ldots; M_{H,t}]$$

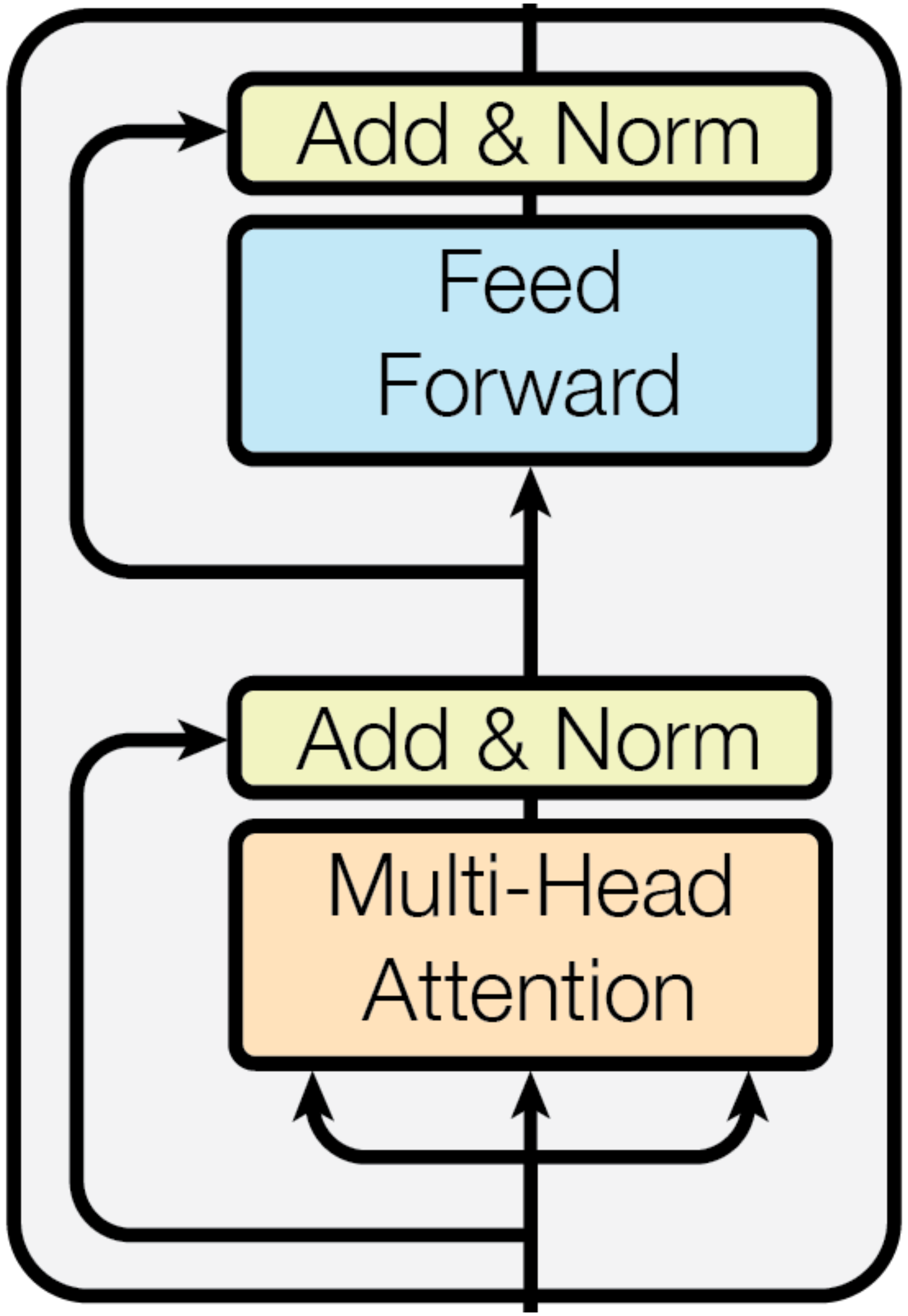


(Vaswani et al., 2017)

Transformer Block

- Multi-headed attention is the main innovation of the transformer model!
- Each block also composed of:
  - a layer normalisations
  - a feedforward network
  - residual connections
- Feedforward network also composed of trainable parameters

$$y = \mathrm{gelu}(W_2\,\mathrm{gelu}(W_1 x + b_1) + b_2)$$



Vaswani et al., 2017

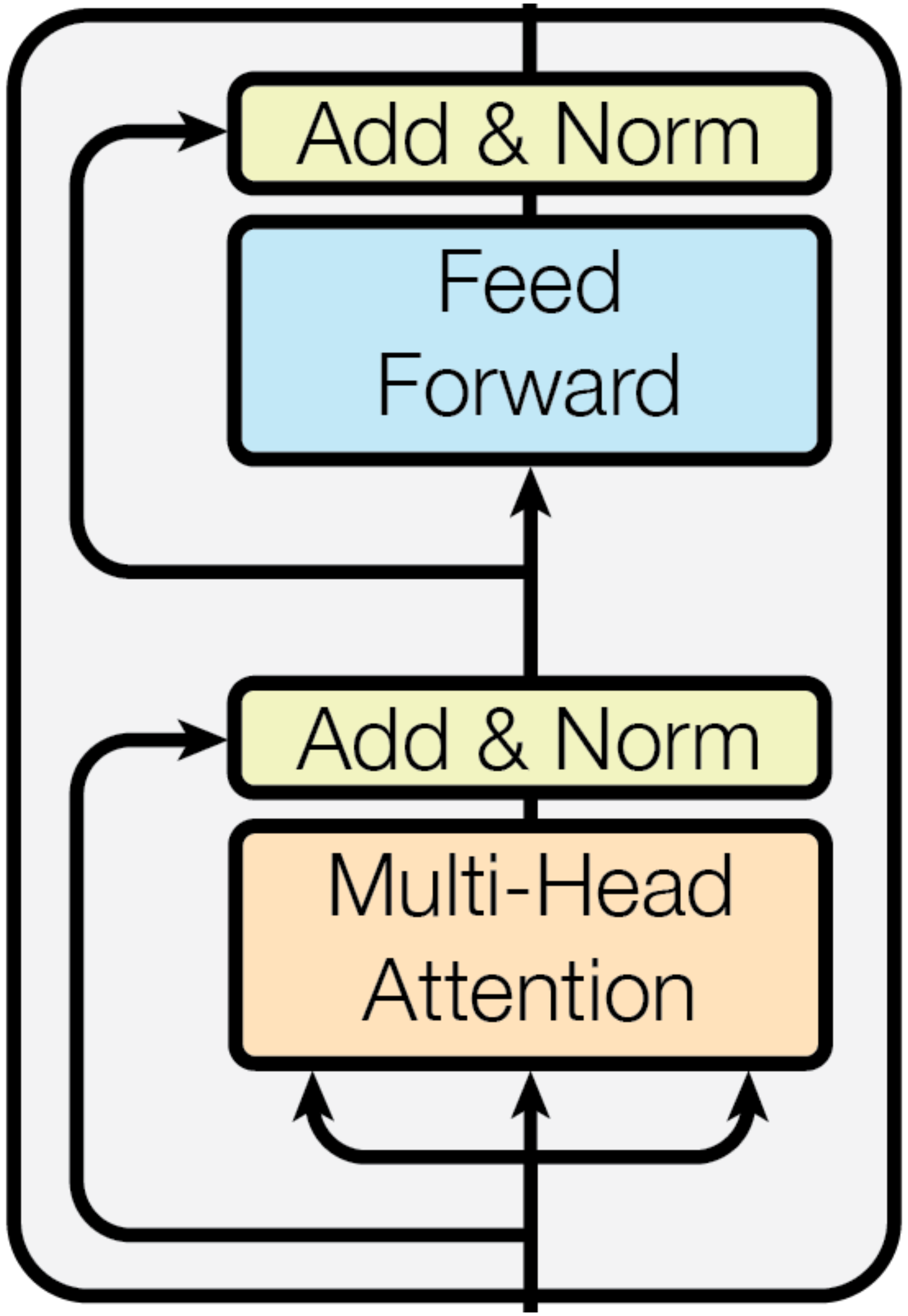
Question

What are the learnable parameters in these matrices?

Transformer Block

- Multi-headed attention is the main innovation of the transformer model!
- Each block also composed of:
  - a layer normalisations
  - a feedforward network
  - residual connections
- Feedforward network also composed of trainable parameters

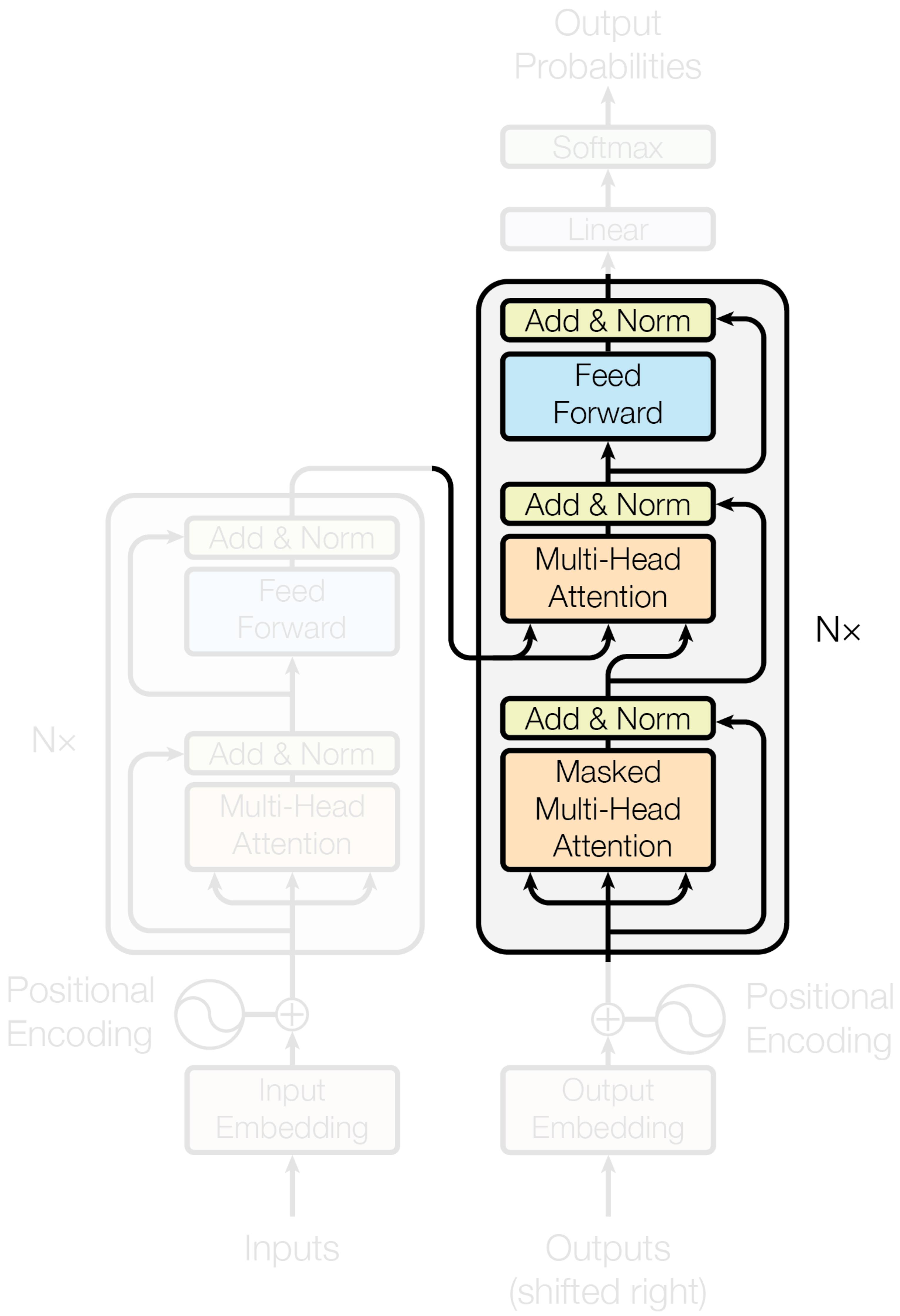
$$y = \text{relu}(W_2\, \text{relu}(W_1 x + b_1) + b_2)$$



Vaswani et al., 2017

Full Transformer

- Transformer decoder (right) similar to encoder
  - First layer of block is **masked** multi-headed attention
  - Second layer is multi-headed attention over *final-layer* encoder outputs **(cross-attention)**
  - Third layer is feed-forward network
- **Different parameters for masked self-attention, cross-attention, and feedforward networks**



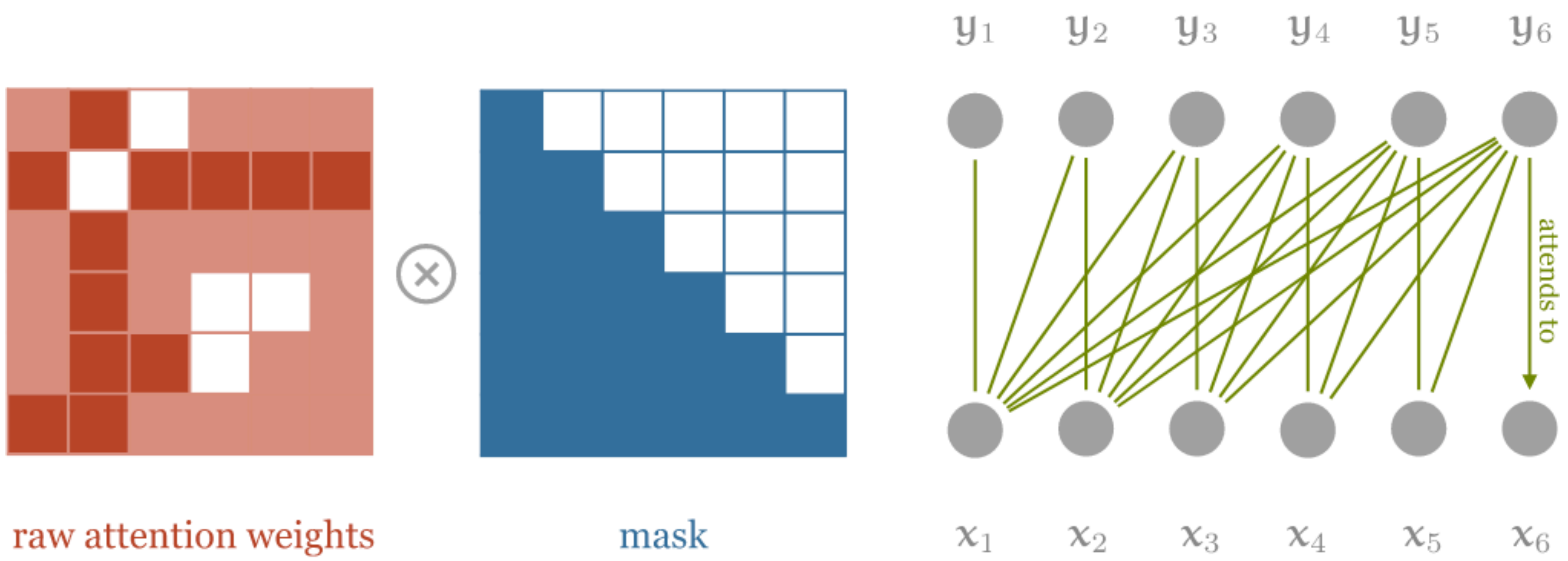
Vaswani et al., 2017

Masked Multi-headed Attention

- Self-attention can attend to any token in the sequence

Mask the attention scores of future tokens so their attention = 0

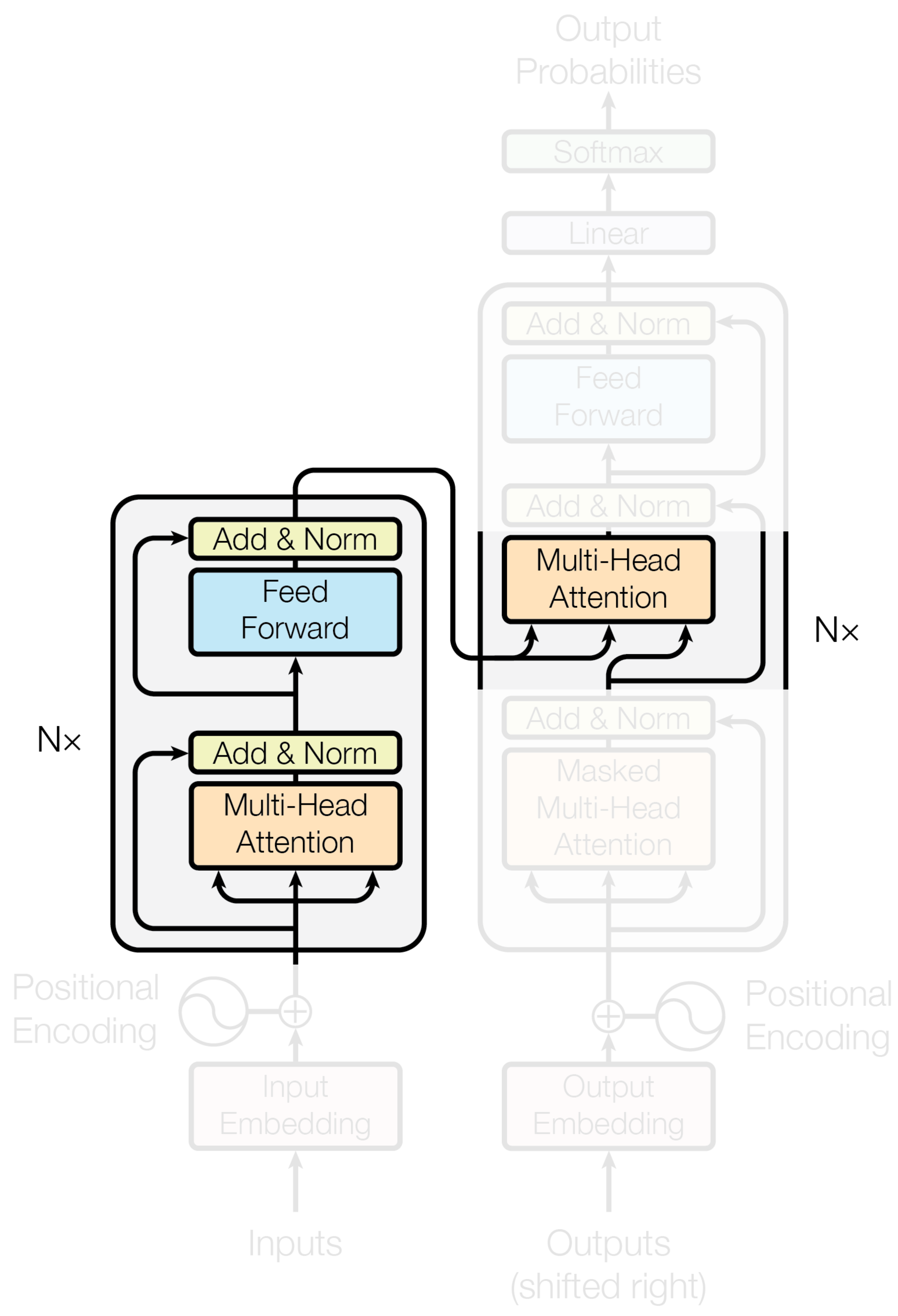
- For the decoder, **you don't want tokens to attend to future tokens**
  - Decoder used to generate text (i.e., machine translation)



$$a_{st} = \frac{(\mathbf{W}^Q q)^T (\mathbf{W}^K K)}{\sqrt{d}} \quad \Rightarrow \quad a_{st} := a_{st} - \infty \ ; \ s < t \quad \Rightarrow \quad \alpha_{st} = \frac{e^{a_{st}}}{\sum_j e^{a_{sj}}} = 0$$

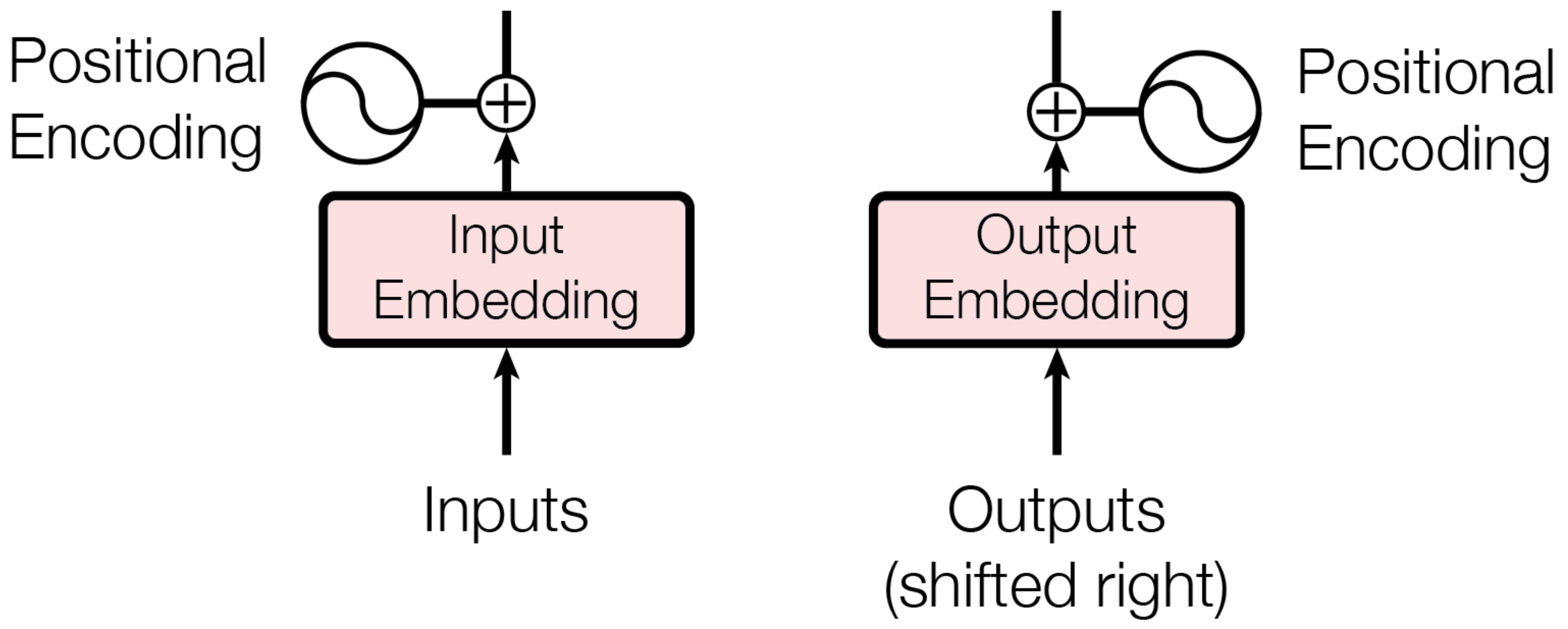
Cross-attention

- **Cross attention** is the same classical attention as in the RNN encoder-decoder model
- Keys and values are output of **final** encoder block
- Query to the attention function is output of the masked multi-headed attention in the decoder
  - A representation from the decoder is used to **attend** to the encoder outputs



Vaswani et al., 2017

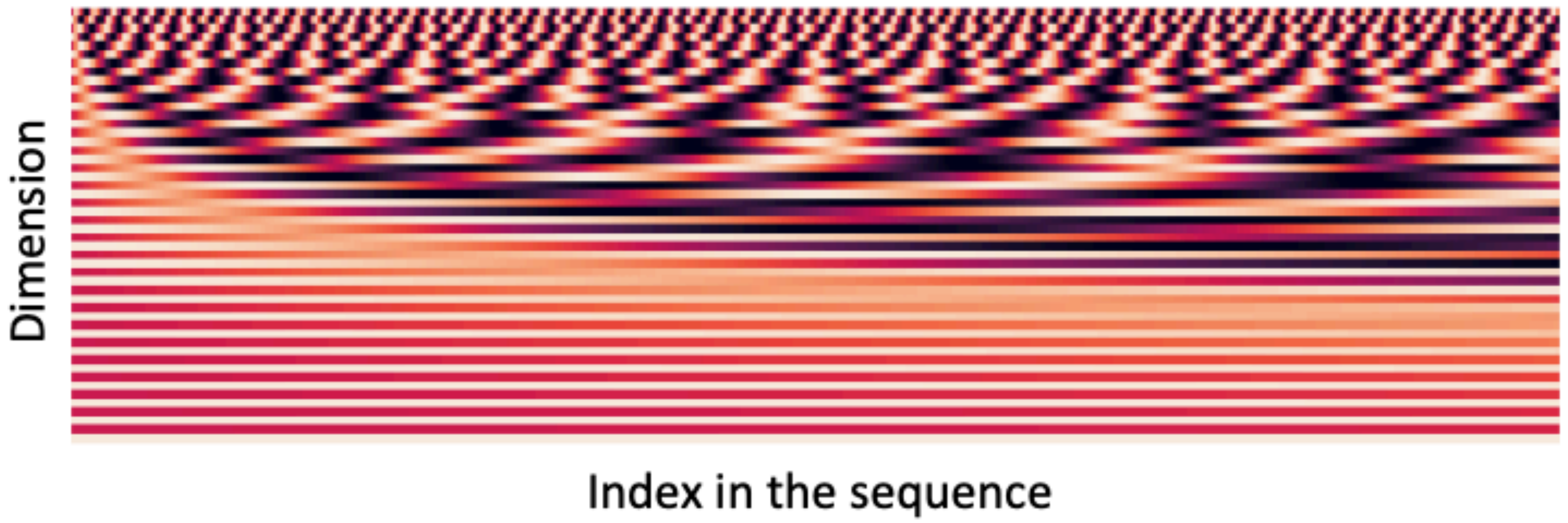
Position Embeddings



- Early position embeddings encoded a sinusoid function that was offset by a phase shift proportional to sequence position

- **In practice, easiest is to learn position embeddings from scratch**

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$



Vaswani et al., 2017

Recap

- **Self-attention:** compute representations of tokens as mixtures of surrounding tokens
- Modern **Transformers** use self-attention as fundamental building block
- Decoder blocks mask attention on future tokens to not leak information
- Require position embeddings to capture sequence order

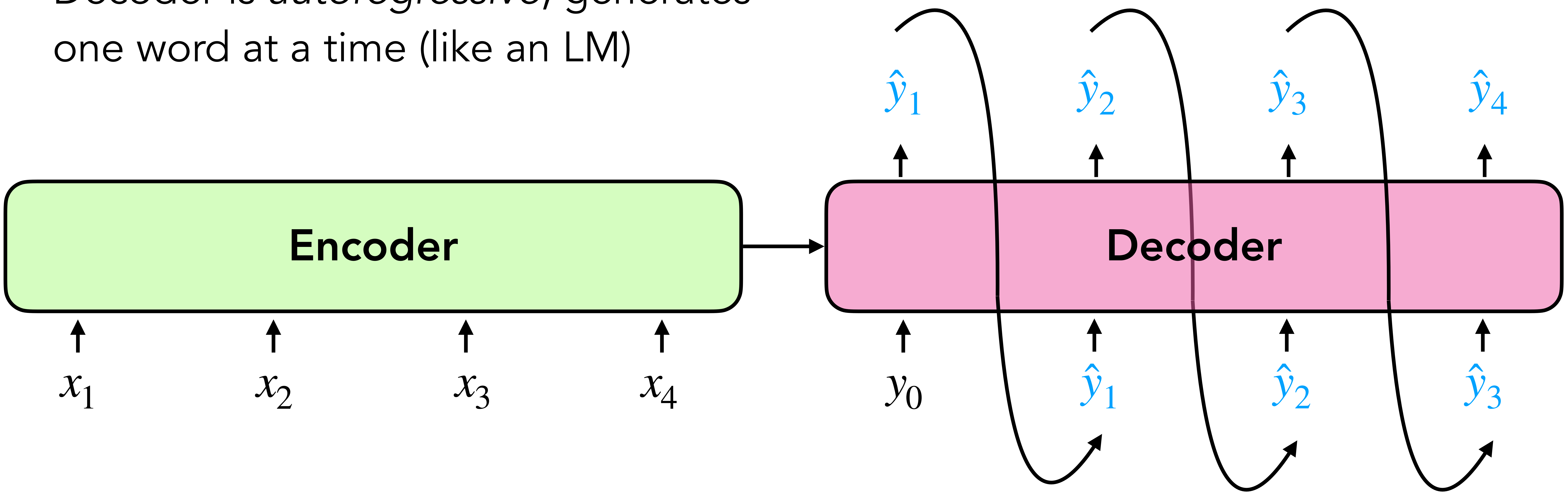
Decoding from Neural Models

Antoine Bosselut

Encoder-Decoder Models

- Encode a sequence fully with one model (**encoder**) and use its representation to seed a second model that decodes another sequence (**decoder**)

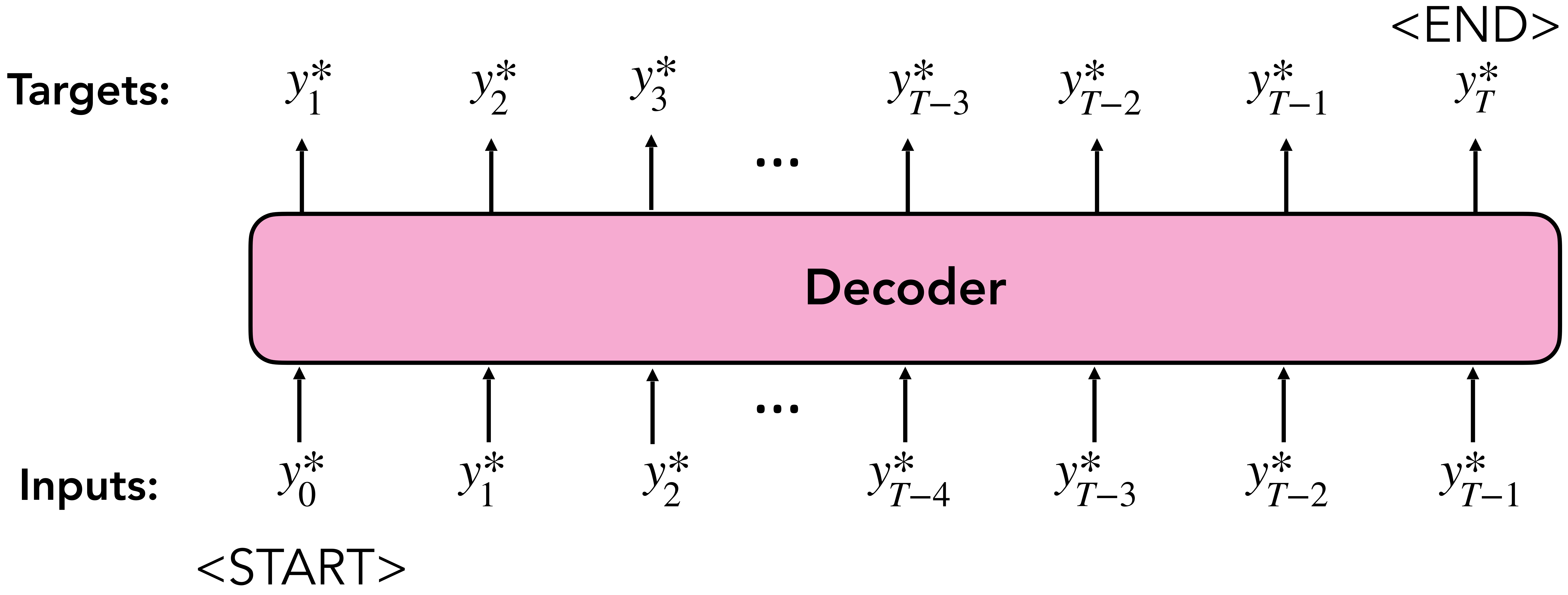
- Decoder is *autoregressive*, generates one word at a time (like an LM)



Training a Decoder

- Minimize the negative log probability of the gold* sequences in your dataset

$$\mathscr{L} = -\sum_{t=1}^{T} \log P\left(y_t^* \mid \{y_s^*\}_{s<t}\right)$$



Decoding: Main Idea

- At each time step *t*, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$:

$$S = f\left(\{y_{<t}\}\right)$$

$f(\,.\,)$ is your decoder

- Then, we compute a probability distribution P over these scores (with a softmax):

$$P\left(y_t = w \,\middle|\, \{y_{<t}\}\right) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Decoding algorithm defines a function to select a token from this distribution:

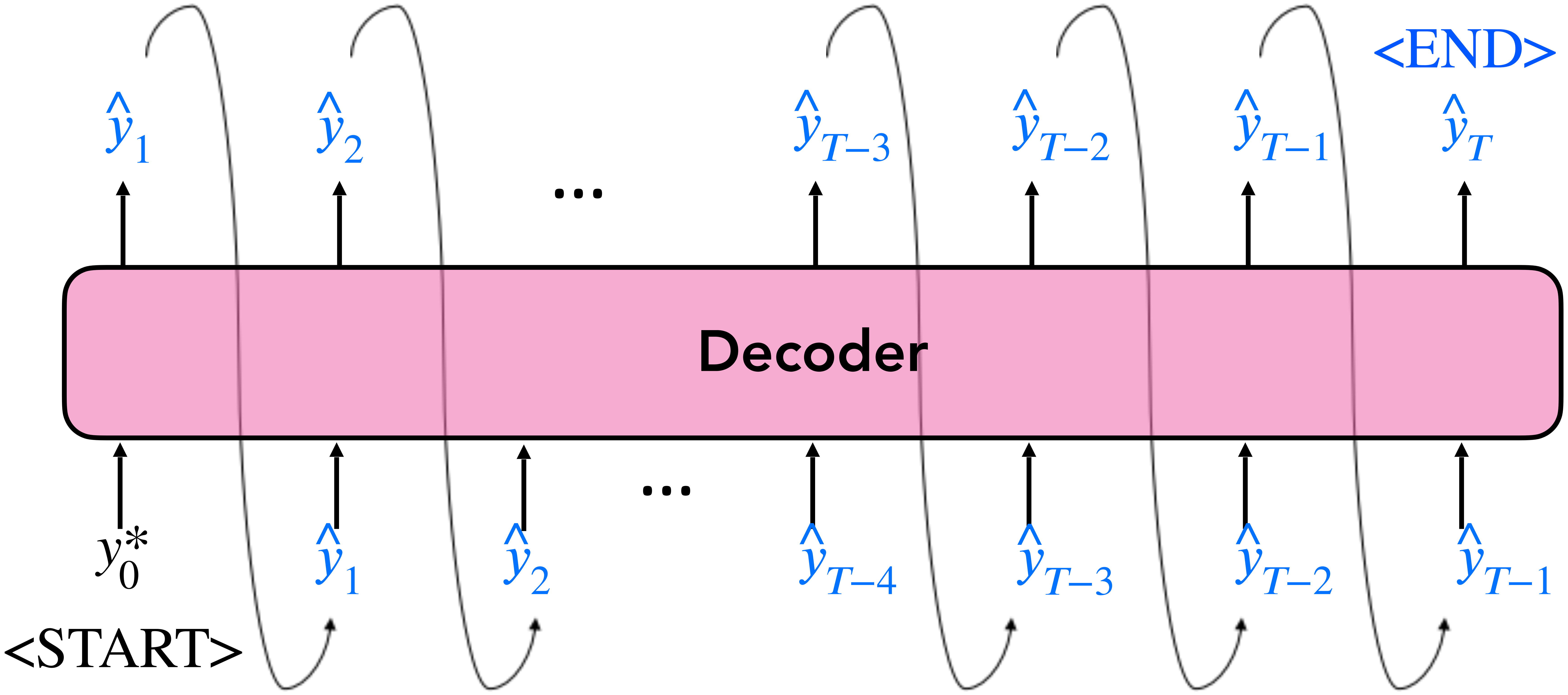
$$\hat{y}_t = g\left(P\left(y_t \mid \hat{y}_{<t}\right)\right)$$

$g(\,.\,)$ is your decoding algorithm

Decoding: Main Idea

- Decoding algorithm defines a function to select a token from this distribution

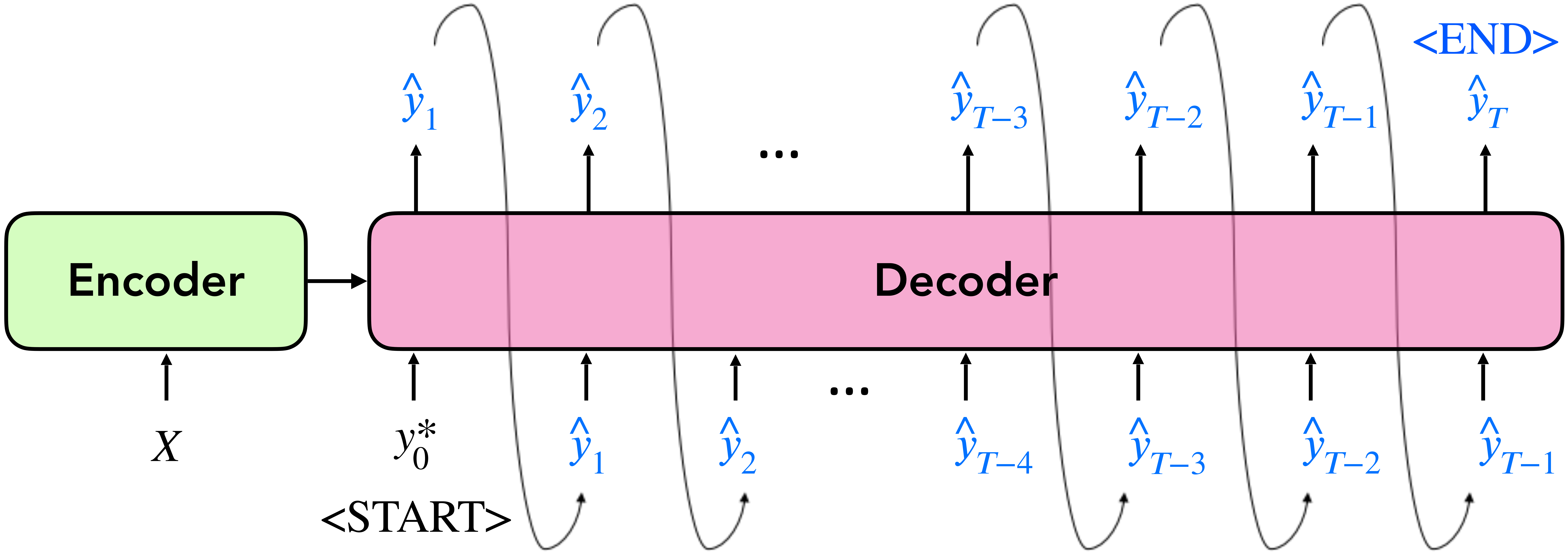
$$\hat{y}_t = g\left(P\left(y_t \mid \hat{y}_{<t}\right)\right)$$



Optional: Encoder Input

- Decoding algorithm defines a function to select a token from this distribution

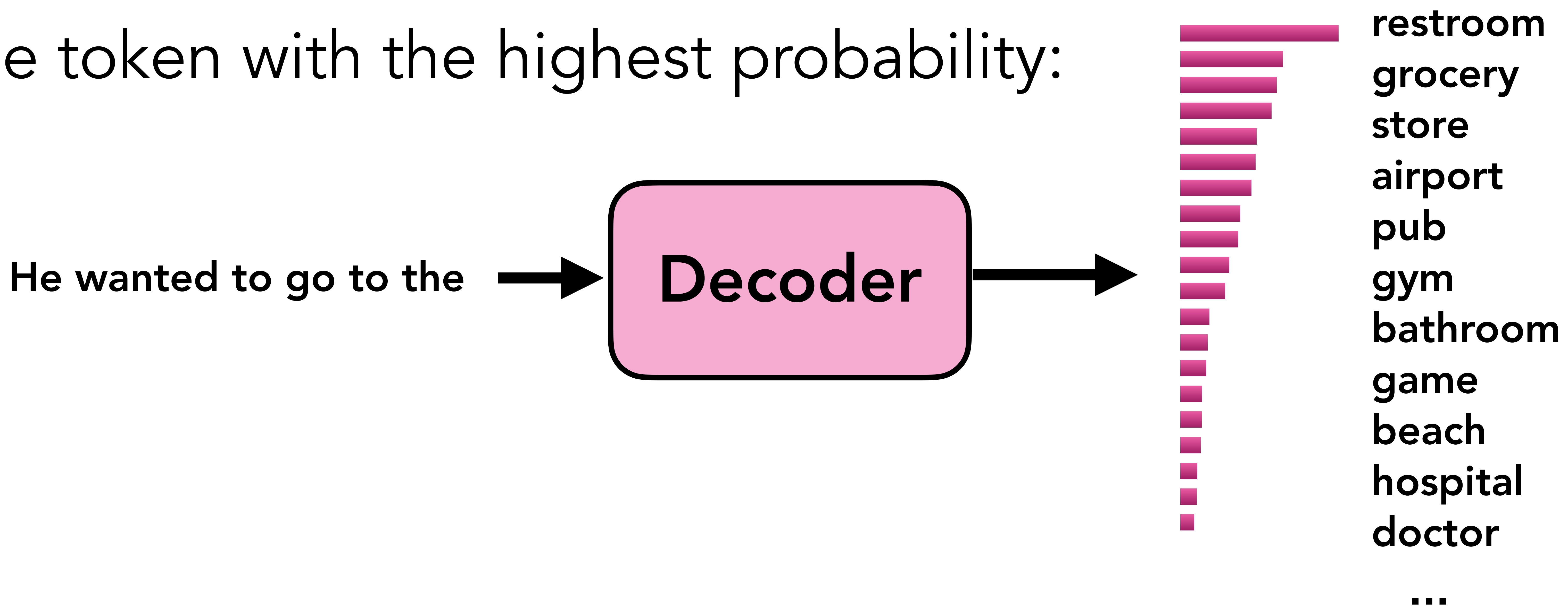
$$\hat{y}_t = g\left(P\left(y_t \mid X, \hat{y}_{<t}\right)\right)$$



Greedy methods: Argmax Decoding

$$\hat{y}_t = \underset{w \in V}{\textbf{argmax}}\ P(y_t = w \mid \{y\}_{<t})$$

- g = select the token with the highest probability:



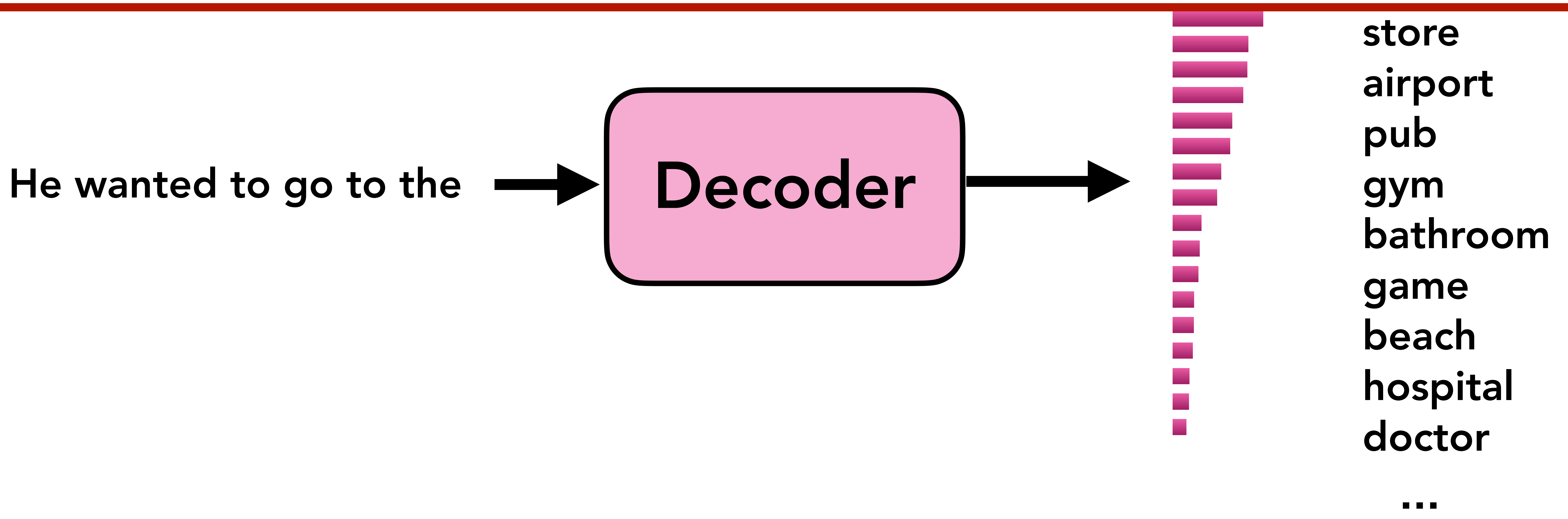
Greedy methods: Argmax Decoding

$$\hat{y}_t = \textbf{argmax}\ P(y_t = w \mid \{y\}_{<t})$$

Select **highest scoring** token

What's a potential problem with argmax decoding?

- g = selec



Issues with argmax decoding

- In argmax decoding, we cannot revise prior decisions
    - *les pauvres sont démunis (the poor don't have any money)*
    - → *the* ____
    - → *the poor* ____
    - → *the poor are* ____

- Potential leads to sequences that are
    - **Ungrammatical**
    - **Unnatural**
    - **Nonsensical**
    - **Incorrect**

Beam Search

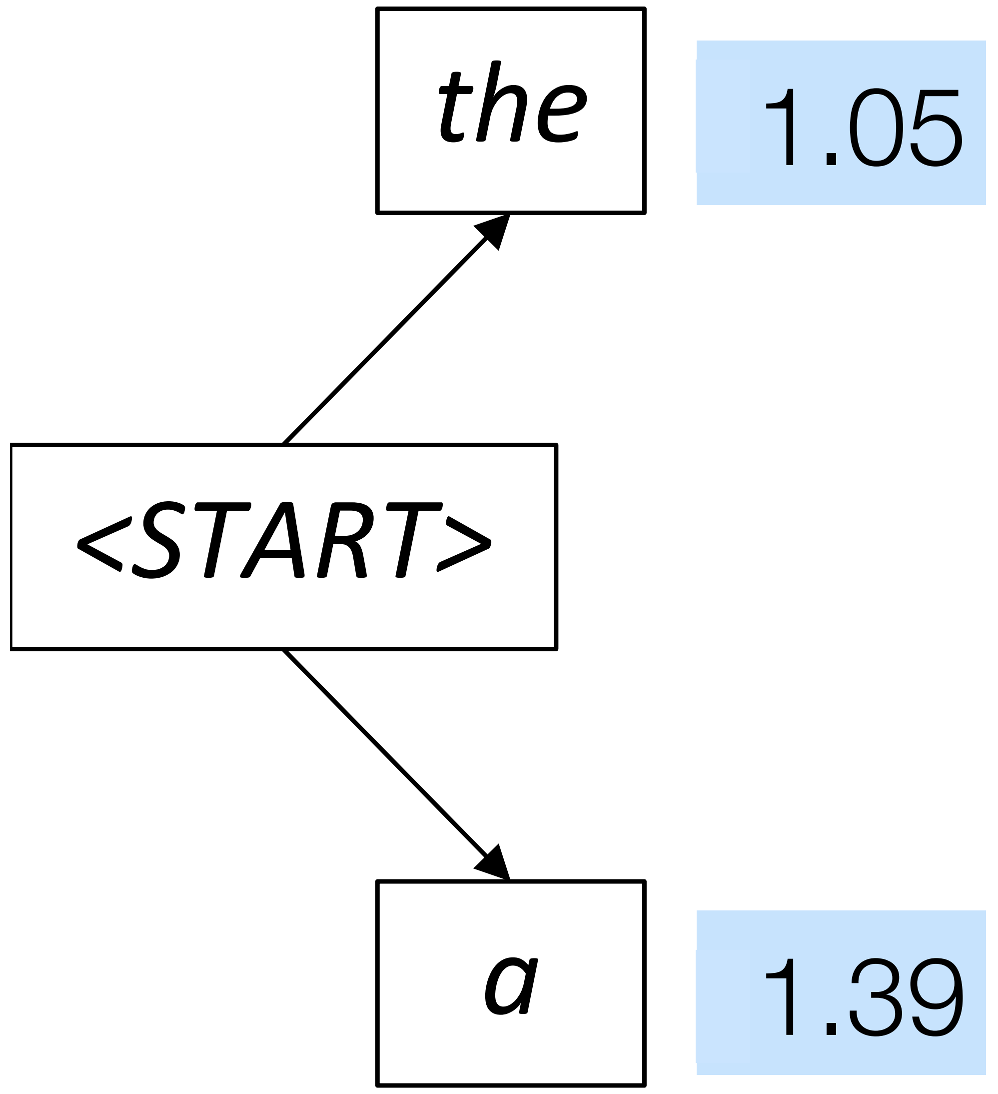
- *les pauvres sont démunis (the poor don't have any money)*
- *→ the ____*
- *→ the poor ____*
- *→ the poor are ____*

- **Beam Search**: Explore several different hypotheses instead of just one
  - Track of the *b* highest scoring sequences at each decoder step instead of just one
  - Score at each step: $-\sum_{t=1}^{j} \log P(\hat{y}_t \mid \hat{y}_1, \ldots, \hat{y}_{t-1}, X)$
  - *b* is called the **beam size**

Beam Search

Beam size = 2

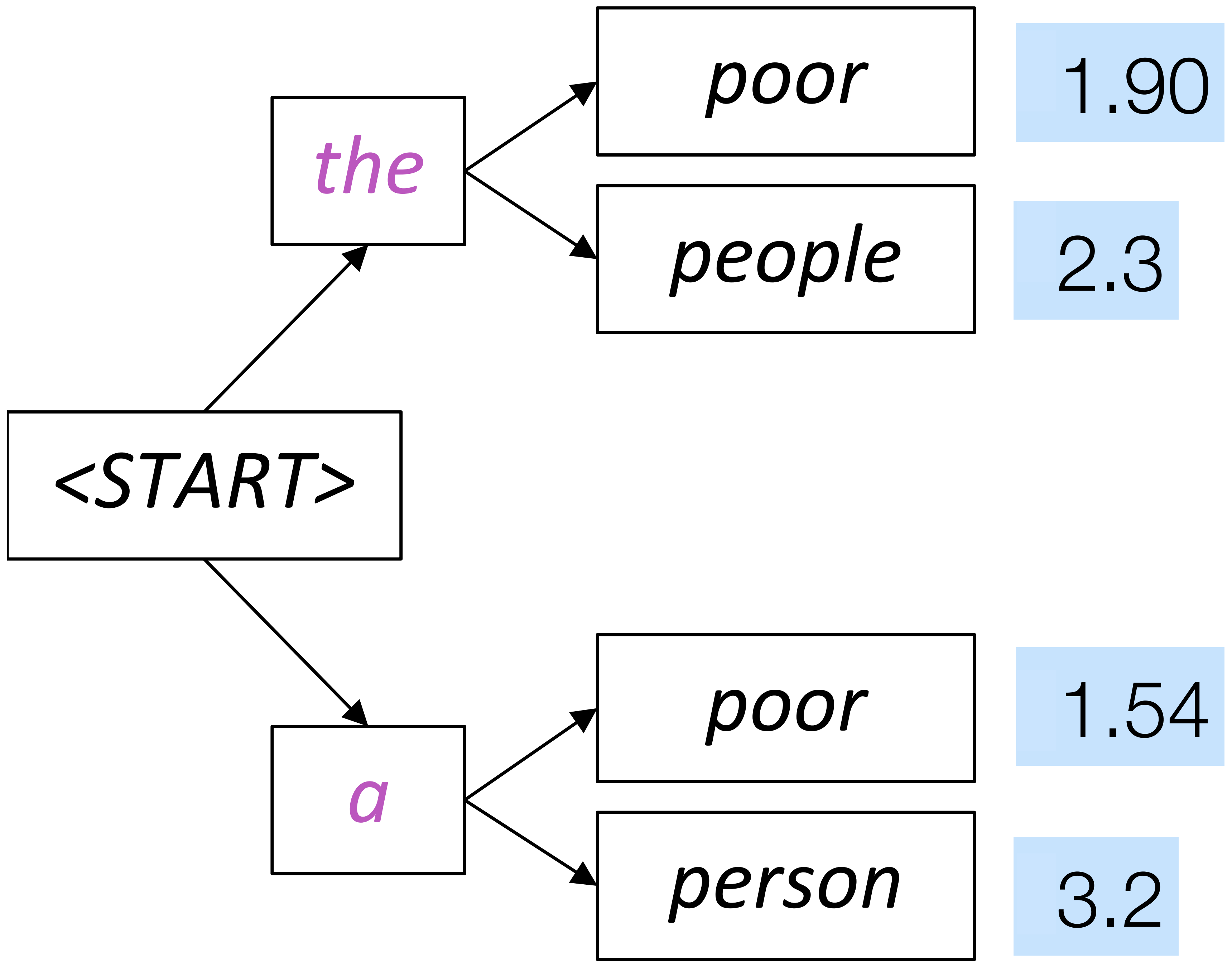
$-\log P(\hat{y}_1 \mid y_0)$



Beam Search

Beam size = 2

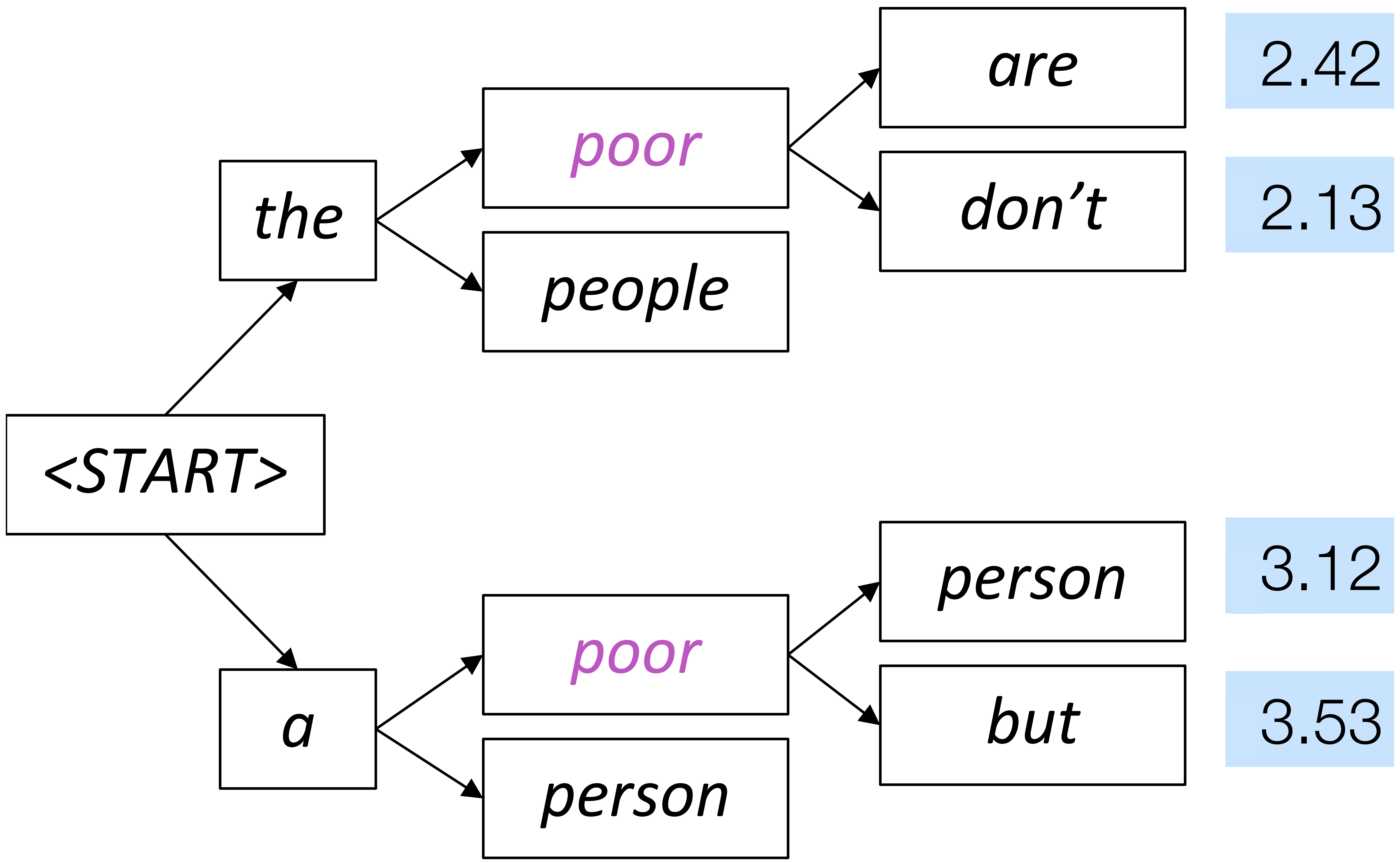
$$-\sum_{t=1}^{2} \log P(\hat{y}_t \mid \hat{y}_0, \ldots, \hat{y}_{t-1})$$



Beam Search

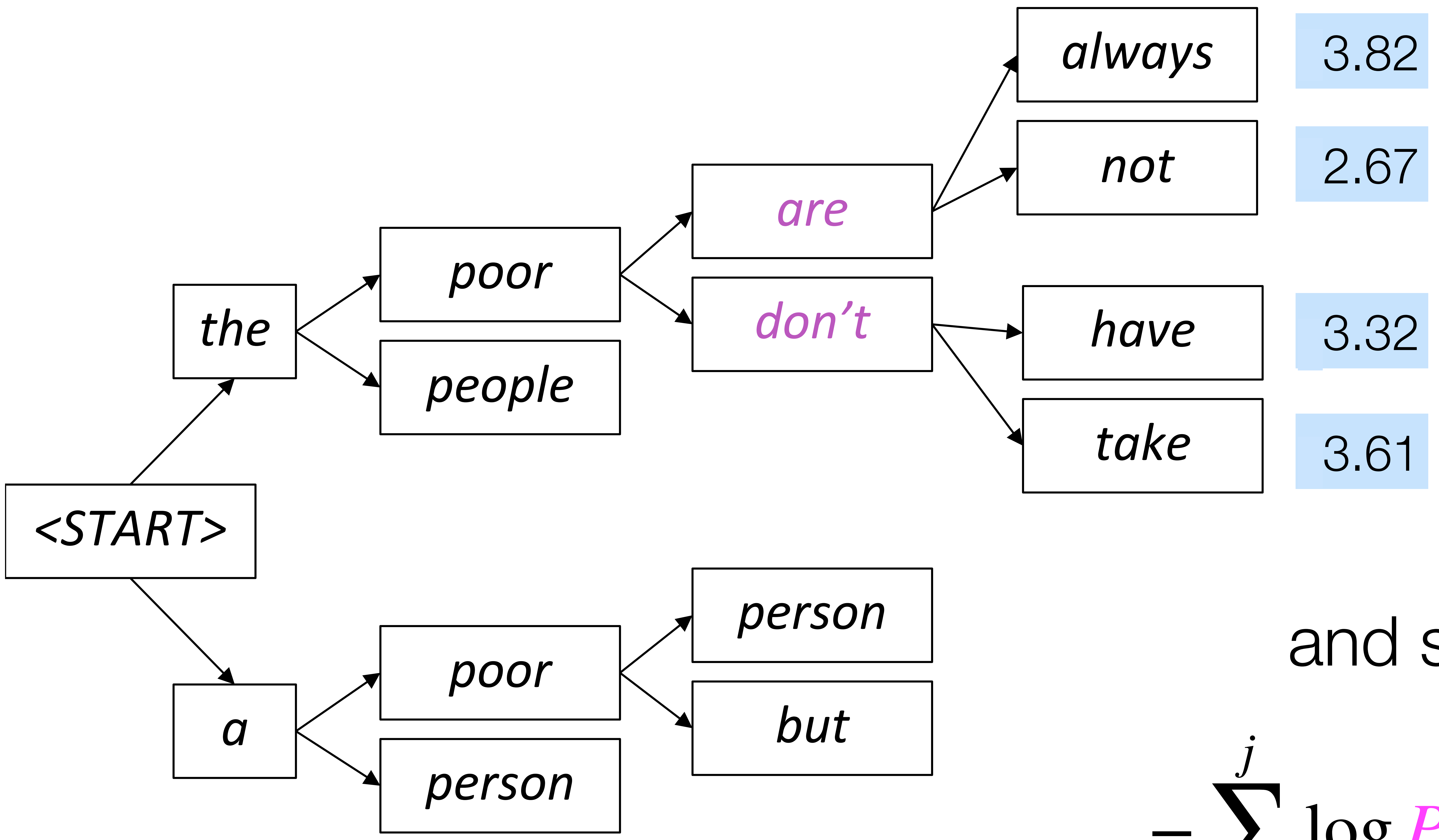
Beam size = 2

$$-\sum_{t=1}^{3} \log P(\hat{y}_t \mid y_0, \hat{y}_1, \ldots, \hat{y}_{t-1})$$



Beam Search

Beam size = 2

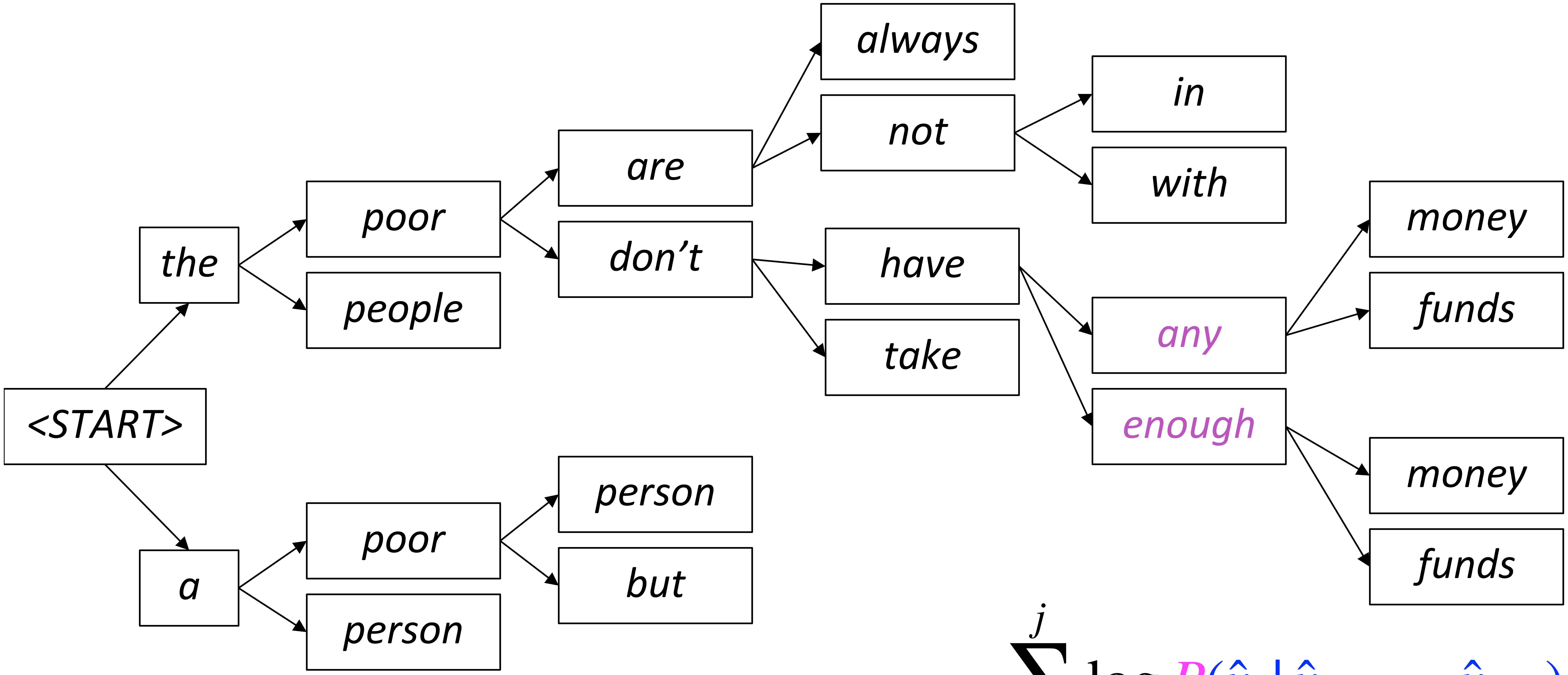


and so on…

$$-\sum_{t=1}^{j} \log P(\hat{y}_t \mid \hat{y}_1, \ldots, \hat{y}_{t-1})$$

Beam Search

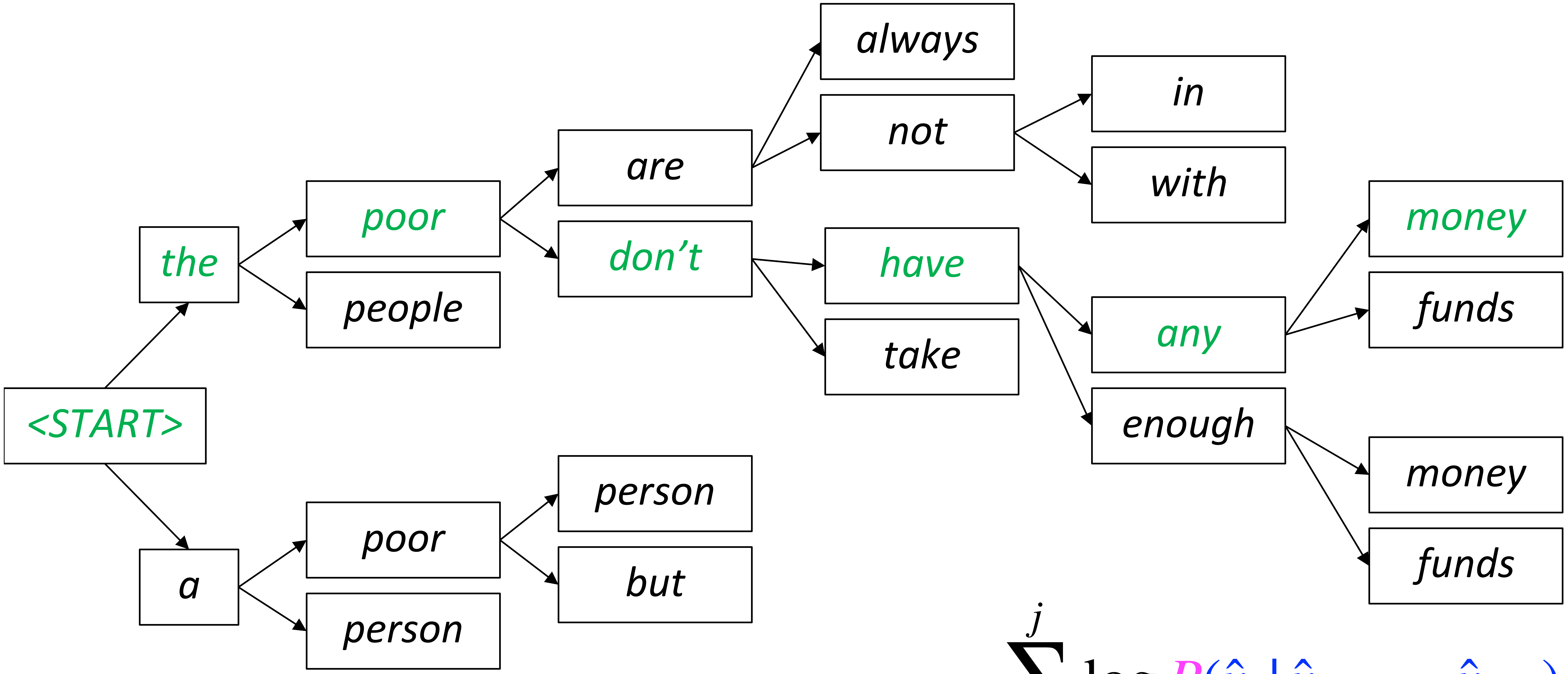
Beam size = 2



$$-\sum_{t=1}^{j} \log P(\hat{y}_t \mid \hat{y}_1, \ldots, \hat{y}_{t-1})$$

Beam Search

Beam size = 2



$$-\sum_{t=1}^{j} \log P(\hat{y}_t \,|\, \hat{y}_1, \ldots, \hat{y}_{t-1})$$

Beam Search

- Different hypotheses may produce <END> token at different time steps
  - When a hypothesis produces <END>, stop expanding it and place it aside
- Continue beam search until:
  - All b beams (hypotheses) produce <END> OR
  - Hit max decoding limit T
- Select top hypotheses using the *normalized* likelihood score

$$-\frac{1}{T}\sum_{t=1}^{T}\log P(\hat{y}_t \mid \hat{y}_1, \ldots, \hat{y}_{t-1}, X)$$

  - Otherwise shorter hypotheses have higher scores

What do you think might happen if we increase the beam size?

Effect of beam size

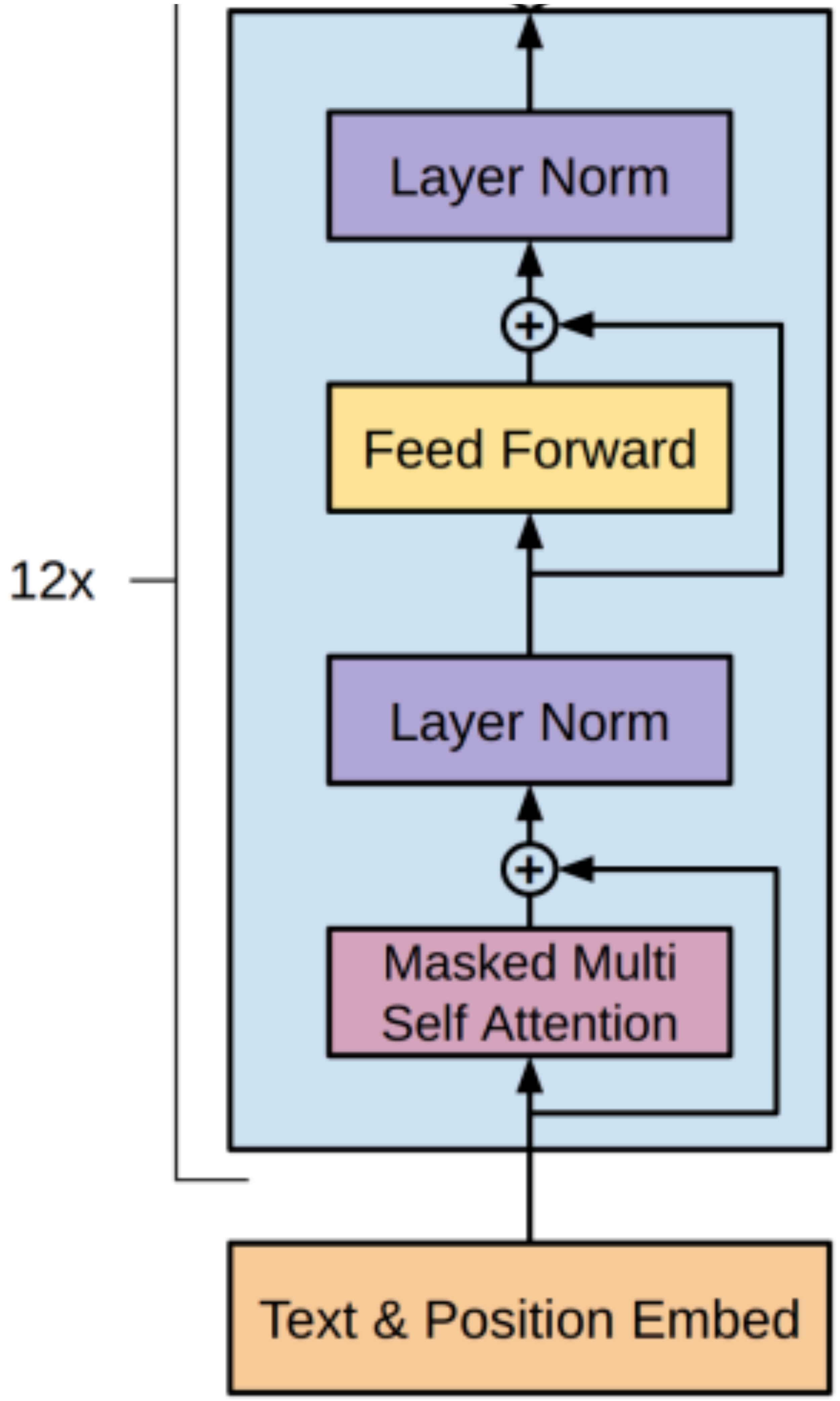
- Small beam size *b* has similar issues as argmax decoding
    - **Outputs that are ungrammatical, unnatural, nonsensical, incorrect**
    - b=1 is the same as argmax decoding
- Larger beam size *b* reduces some of these problems
    - Potentially much more computationally expensive
    - Outputs tend to get shorter and more generic

GPT: Generative Pretrained Transformer

Antoine Bosselut

GPT: Generative Pretrained Transformer

- Called a *decoder* transformer
- **But**, actual GPT block mixes design of encoder and decoder from original transformer
- Uses masked multi-headed self-attention (decoder)
  - Can't see future
- No cross-attention; only computes a self-attention over its history in each block (encoder)



Training GPT

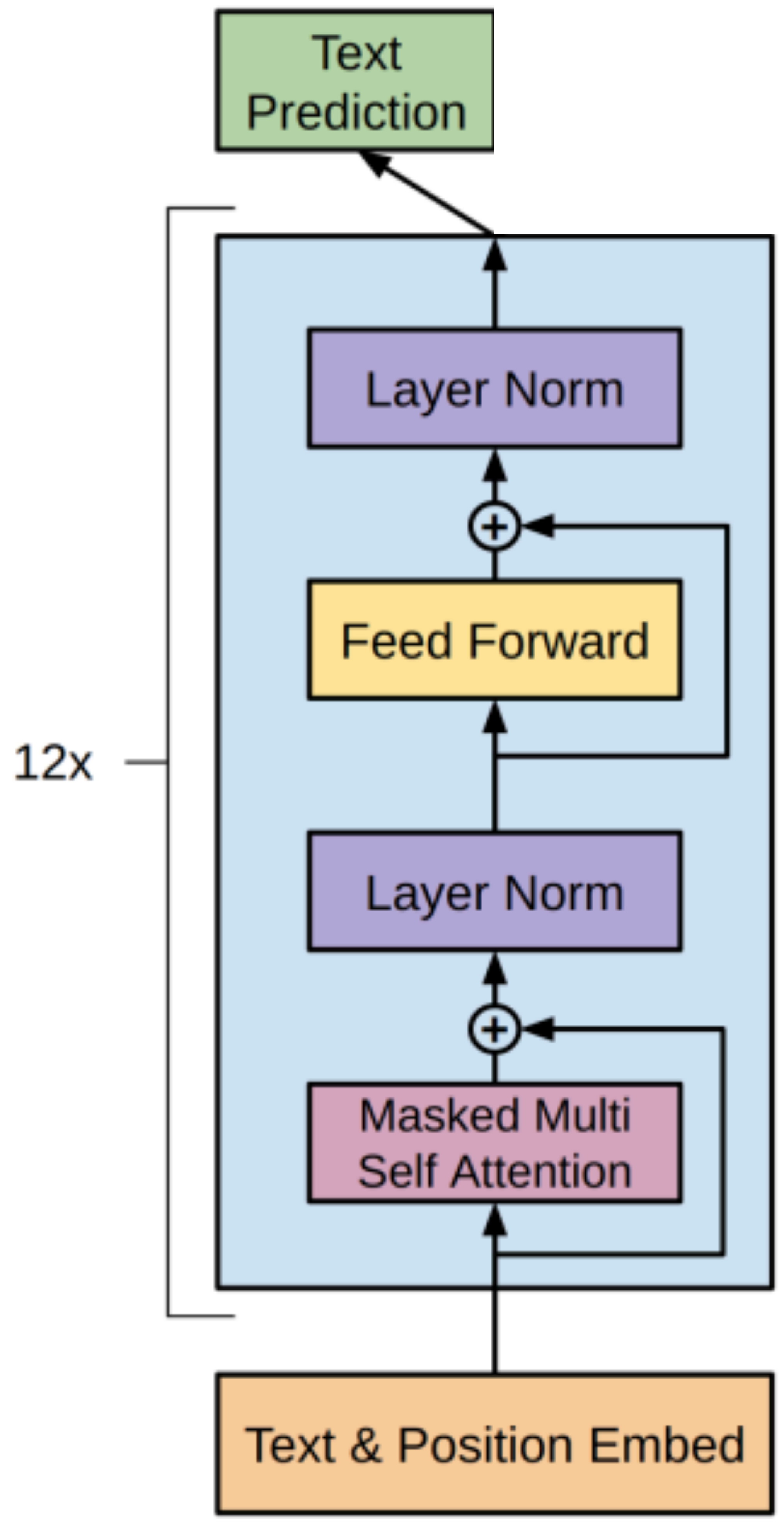
- Pretrained on TorontoBooks corpus: 7000 unpublished books (~13 GB)

- Corpus segmented broken up into windows of 512 tokens

  - can model long-range context during pretraining

- **Pretraining task:** next word prediction (i.e., language modelling)

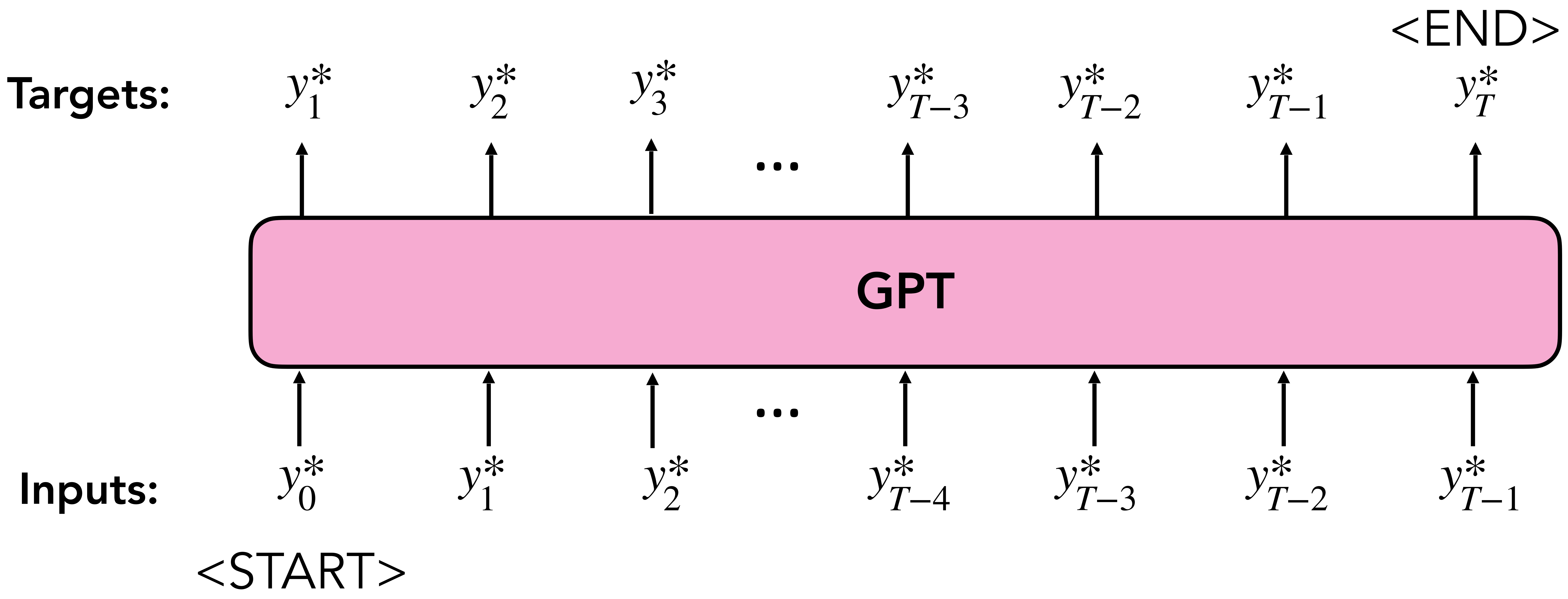
$$\mathscr{L} = -\sum_{t=1}^{T} \log P\left(y_t^* \mid \{y_s^*\}_{s<t}\right)$$



Pretraining

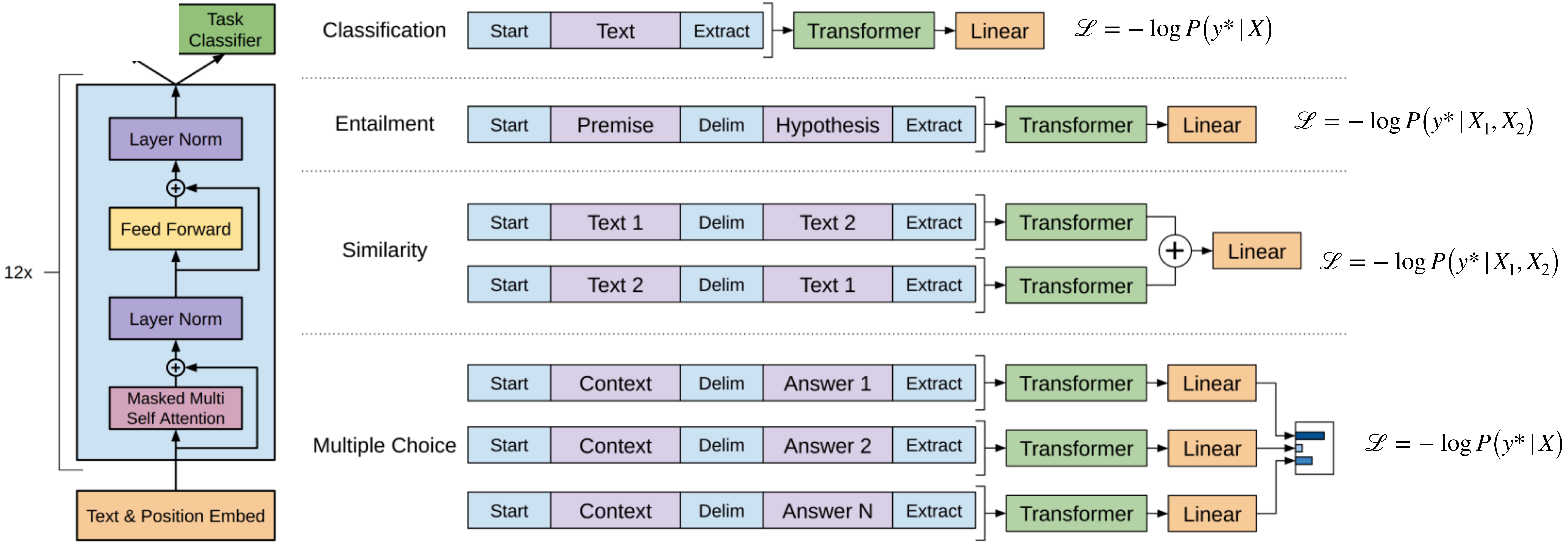
- Minimize the negative log probability of the gold* sequences in your dataset

$$\mathscr{L} = -\sum_{t=1}^{T} \log P\left(y_t^* \mid \{y_s^*\}_{s<t}\right)$$



Fine-tuning

- After pre-training, model can be fine-tuned by training on individual datasets
- Pretrained model used as initialisation for training on individual tasks



Massive Improvements (back then)

Dataset	Task	SOTA	Ours
SNLI	Textual entailment	89.3	89.9
MNLI matched	Textual entailment	80.6	82.1
MNLI mismatched	Textual entailment	80.1	81.4
SciTail	Textual entailment	83.3	88.3
QNLI	Textual entailment	82.3	88.1
RTE	Textual entailment	61.7	56.0
STS-B	Semantic similarity	81.0	82.0
QQP	Semantic similarity	66.1	70.3
MRPC	Semantic similarity	86.0	82.3
RACE	Reading comprehension	53.3	59.0
ROCStories	Commonsense reasoning	77.6	86.5
COPA	Commonsense reasoning	71.2	78.6
SST-2	Sentiment analysis	93.2	91.3
CoLA	Linguistic acceptability	35.0	45.4
GLUE	Multi task benchmark	68.9	72.8

https://openai.com/research/language-unsupervised

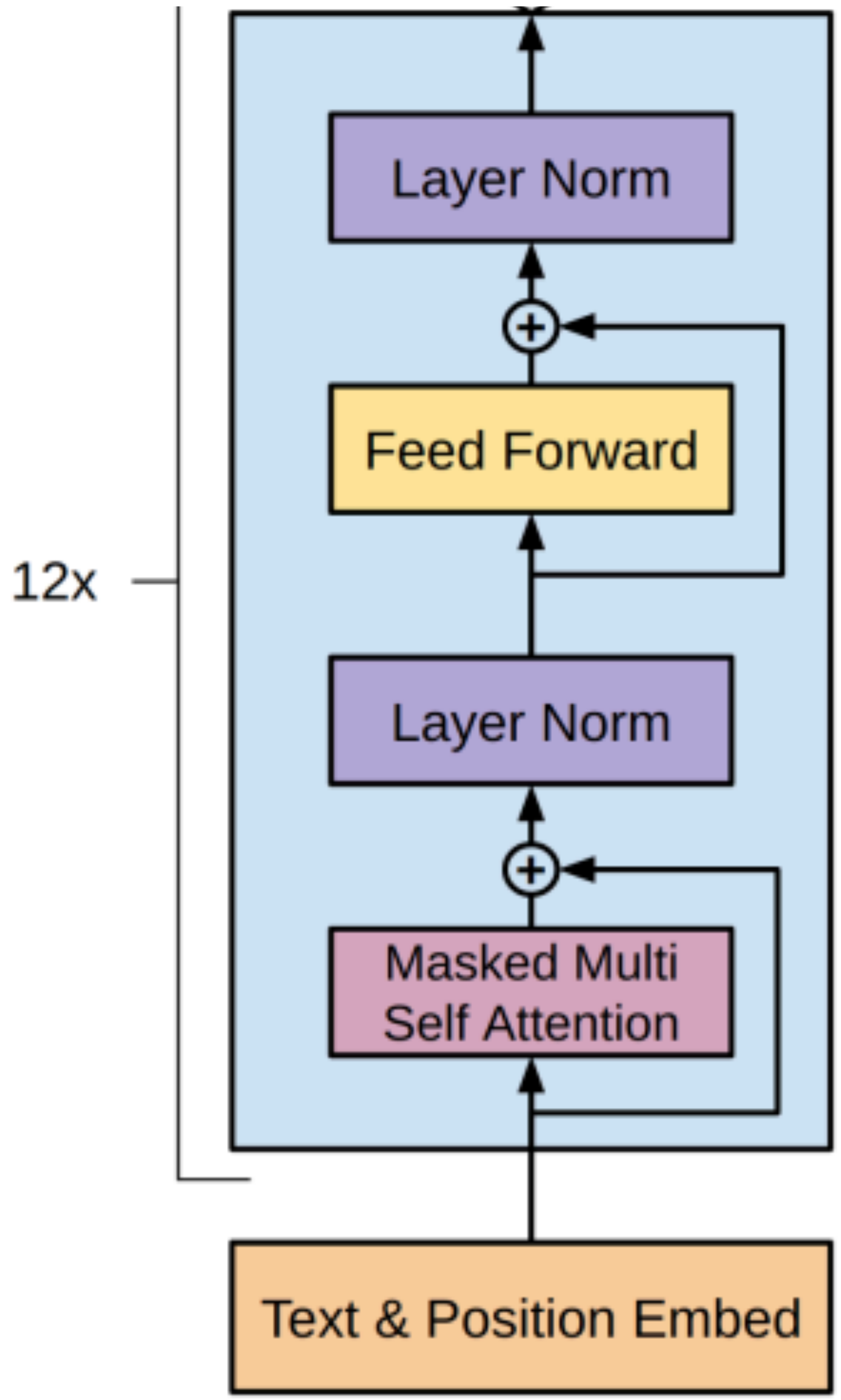
Paradigm Shift

- **Fundamental change:** No longer only using pretrained "embeddings"
  - Word2vec, GloVe, fastText, etc.
- **Using completely pretrained models**

GPT2

- Roughly the same architecture as GPT
- Trained on 40 GB of data: OpenWebText
  - Articles linked to from reddit threads
- Four model sizes:

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600



GPT2

Dataset	Metric	Our result	Previous record	Human
Winograd Schema Challenge	accuracy (+)	70.70%	63.7%	92%+
LAMBADA	accuracy (+)	63.24%	59.23%	95%+
LAMBADA	perplexity (–)	8.6	99	~1–2
Children's Book Test Common Nouns (validation accuracy)	accuracy (+)	93.30%	85.7%	96%
Children's Book Test Named Entities (validation accuracy)	accuracy (+)	89.05%	82.3%	92%
Penn Tree Bank	perplexity (–)	35.76	46.54	unknown
WikiText-2	perplexity (–)	18.34	39.14	unknown
enwik8	bits per character (–)	0.93	0.99	unknown
text8	bits per character (–)	0.98	1.08	unknown
WikiText-103	perplexity (–)	17.48	18.3	unknown

https://openai.com/research/better-language-models

GPT2

SYSTEM PROMPT (HUMAN-WRITTEN)	*Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.*
MODEL COMPLETION (MACHINE-WRITTEN, SECOND TRY)	The 19-year-old singer was caught on camera being escorted out of the store by security guards. The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back. Scroll down for video Shoplifting: Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today (pictured) The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back The singer was also wearing a pair of black-rimmed glasses, a black jacket, black jeans and black sandals. She was carrying a pair of black and white striped gloves and a small black bag.

Increase in Scale

- GPT (Jun 2018): 117 million parameters, trained on 13GB of data (~1 billion tokens)
- GPT2 (Feb 2019): 1.5 billion parameters, trained on 40GB of data
- GPT3 (July 2020): 175 billion parameters, ~500GB data (~300 billion tokens)
- GPT4 (March 2023): ???? parameters, ????? data

More to come in Week 7!

References

- Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *ArXiv, abs/1706.03762*.
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.