Sequence Modeling, Attention & Transformers

Phillip Rust

Original Slides: Desmond Elliott

This Lecture

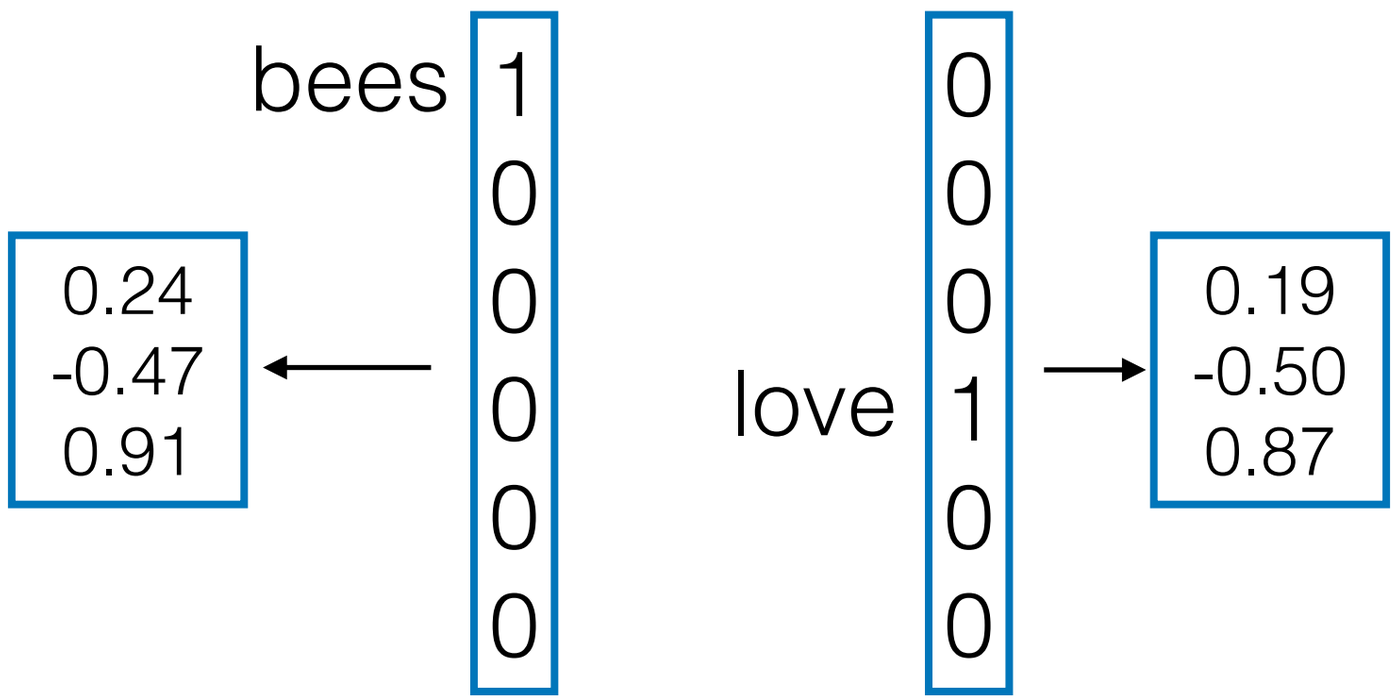
- Recap type-based word embeddings
- Subword representation learning
- Type-based → contextual representations
- From RNNs to Attention-based Models
- Transformer
- Briefly: Transfer Learning with Transformers

Recap: Representing Text

- One-hot vector over a vocabulary of V word types $w \in \mathcal{R}^{|V|}$
- Continuous dense vector with E dimensions (word embedding)

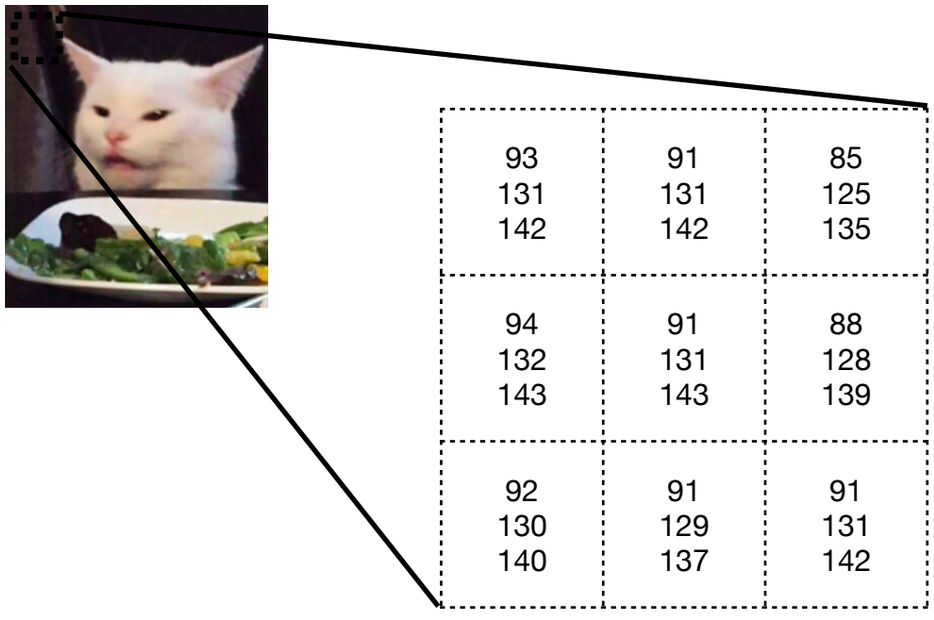
$$x \in \mathcal{R}^{|E|}$$

- One-hot -to- word embedding is a lookup function

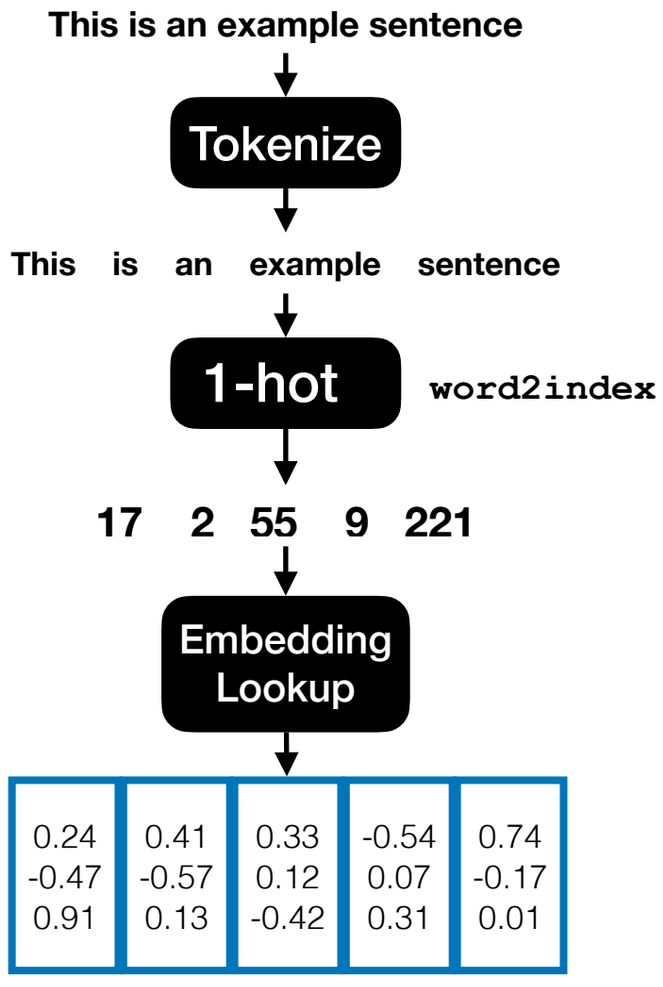


Need for Word Embeddings

Pixel values are continuous values



Words are discrete units



Flavours of Embeddings

- Type-based
  - **Based on the type**
    - Skip-gram
    - CBOW
    - GLoVE
    - FastText
- Contextual
  - **Based on the context** in which the **type** is used.
    - Skip-thought
    - ELMo
    - BERT
    - XLNet

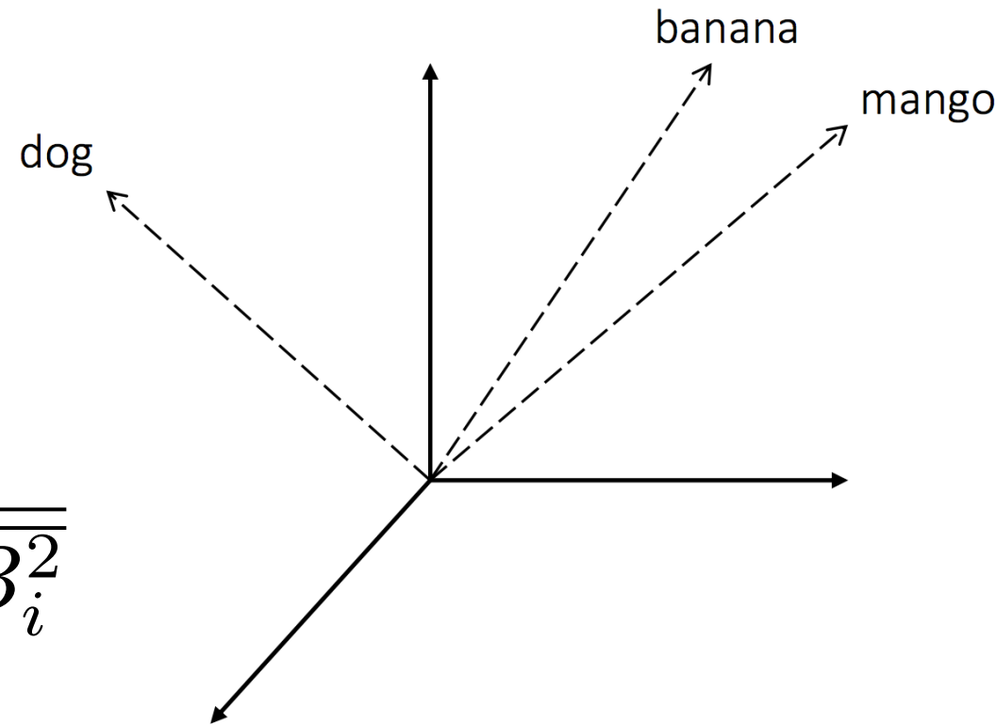
Type-based Representations

Why learn Embeddings?

- Overcome sparsity of one-hot vectors

- Cosine similarity will tell us the **angle formed between vectors A and B,** where A and B are representations of the different terms

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



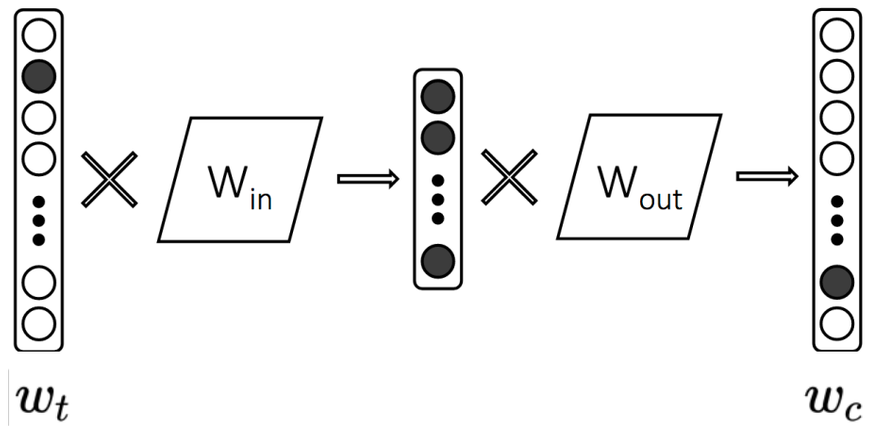
Type-based Skip-gram Embeddings

- Skip-gram embeddings are learned in the context of **predicting the surrounding words** $w_{i-|C|}$, $w_{i+|C|}$,given the target word w_t

$$\mathcal{L} = \sum_{i=1}^{T} \sum_{c \in \mathcal{C}_t} \log \mathrm{p}(w_c | w_t)$$

Over all surrounding windows in the corpus

Estimate probability of predicting each word in the surrounding window



$$\log \mathrm{p}(w_c | w_t) := \mathrm{softmax}(W_{out} W_{in} w_t)$$

- Softmax doesn't scale so you can use noise-contrastive estimation

$$\sum_{t=1}^{T} \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right] \qquad s(w_t, w_c) = \mathbf{u}_{w_t}^{\top} \mathbf{v}_{w_c}$$

Mikolov et al. 2013; Figure Mitra and Craswell 2019

Sub-word Embeddings: Byte-Pair Encoding

- NLP is usually an **open vocabulary** problem so we need techniques to deal with representation of unseen words.

Given a `CORPUS` of words, and integer `merge_operations`

while `merge_operations` > `0`:

1. `symbols` := split `words` in `CORPUS`, given the current `vocabulary`
2. `pairs` := Count all pairs of `symbols` in the corpus
3. Replace the most frequent pair in `pairs (A,B)` with a new symbol `(AB)`
4. Update current `vocabulary` of symbols

`merge_operations--`

```
corpus = {'l o w </w>' : 5,
          'l o w e r </w>' : 2,
          'n e w est</w>':6,,
          'w i d est</w>':3}}}
```

```
merge_ops = 10
```

```
('e', 's')
('es', 't')
```

See Also:

WordPiece
SentencePiece

Sennrich et al. 2016

Type-based FastText Embeddings

- The main shortcoming of type-based embeddings is **how should we deal with out-of-vocabulary tokens**?

- FastText proposes to split words into all sequences of 3—6 characters and learn embeddings $\mathbf{z}_g$ for each sequence

subwords(**where**)

<wh	**<whe**	**<wher**	**<where**
whe	**wher**	**where**	**where>**
her	**here**	**here>**	
ere	**ere>**		
re>			

FastText

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c$$

Type-based

$$s(w_t, w_c) = \mathbf{u}_{w_t}^\top \mathbf{v}_{w_c}$$

Bojanowski et al. 2016

Contextual Embeddings

Contextualized Word Meaning

"Y*ou shall know a word by the company it keeps*"
[Firth, 1957]

My **house** has two rooms and a balcony.

Does your **house** has a bathtub?

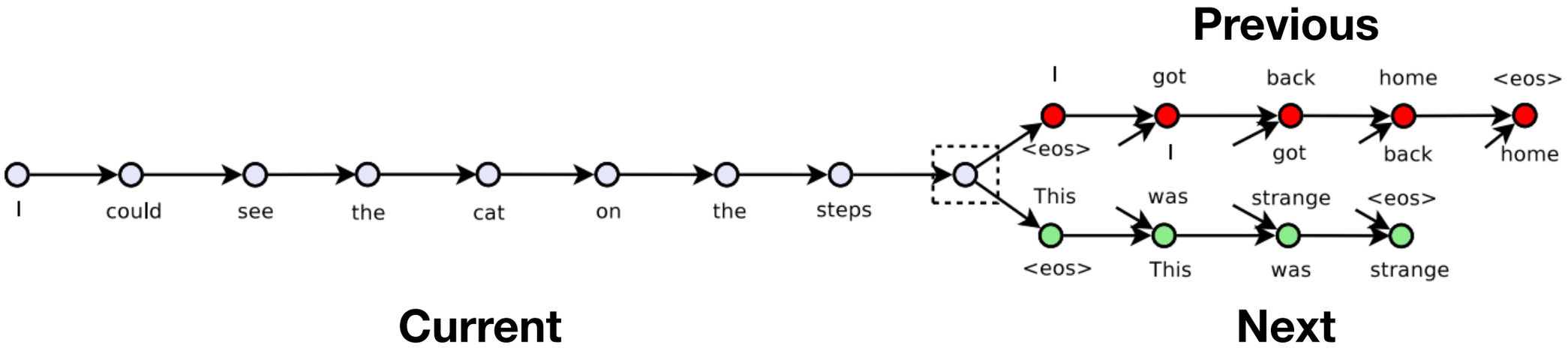
Where will we **house** the students?

This **house** has a beautiful view of the coast.

The shell of a snail is also its **house**.

Contextual Skip-thought Embeddings

- Learn representations in the encoder by learning to predict the tokens in the next and previous sentence.



- Trained on 984,846,357 words in 11,000 books
- Objective: $\sum_t \log P\left(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i\right) + \sum_t \log P\left(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i\right)$

Kiros et al. (NeurIPS 2015)

Contextual Embeddings from Language Models

- Represent each token using multi-layer bidirectional LSTMs.

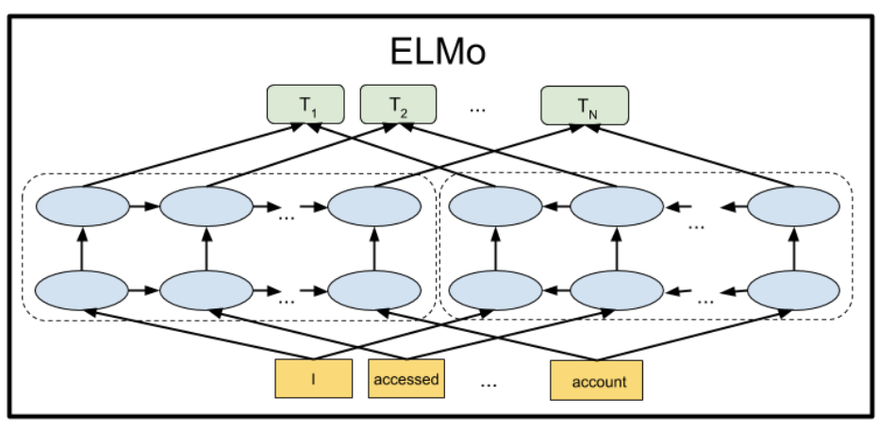
- Final representation is a **weighted sum of the representations** at the different layers in the model

$$\mathbf{ELMo}_k^{task} = E\left(R_k; \Theta^{task}\right) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

Sum over each Layer

Softmax-normalized task weights

Feature vector for token *k* at layer *j*



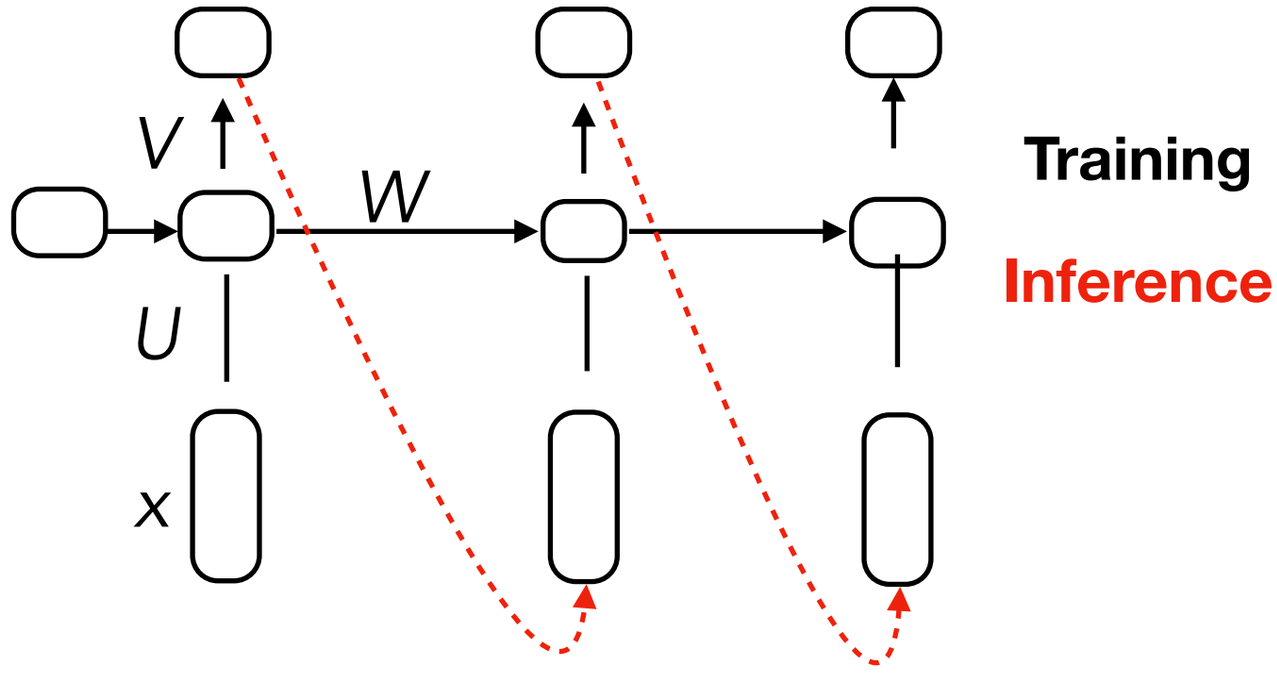
Peters et al. 2018;
Figure: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

Autoregressive Recurrent Neural Networks

- Elman-style model (Cognitive Science, 1990)
  - LSTM (Hochreiter and Schmidhuber, 1997)
  - Gated Recurrent Unit (Cho et al. 2014)

$$o_t = \text{softmax}(V s_t)$$

$$s_t = f(U x_t + W s_{t-1})$$



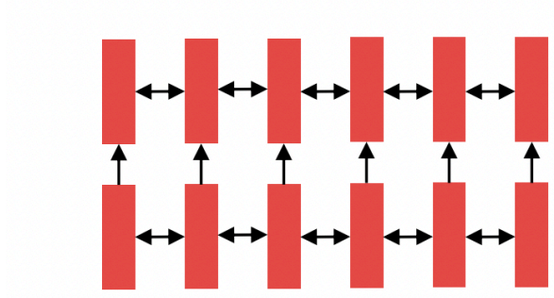
- Create a **contextualized representation** of a word in the sequence of hidden states s_t

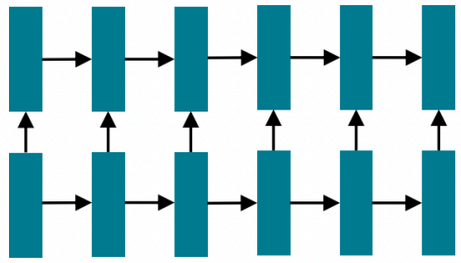
From RNNs to Attention-based Models

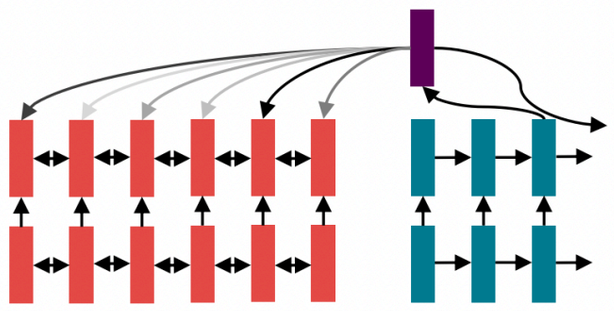
Slide: Modified from John Hewitt

State of NLP pre-2017

- Contextualization via bidirectional RNNs (LSTMs / GRUs):
  - 1. Encode input sequence
  - 2. Define outputs
  - 3. Use attention to remember and access relevant information







Overall, this worked pretty well!

Benefits of RNNs

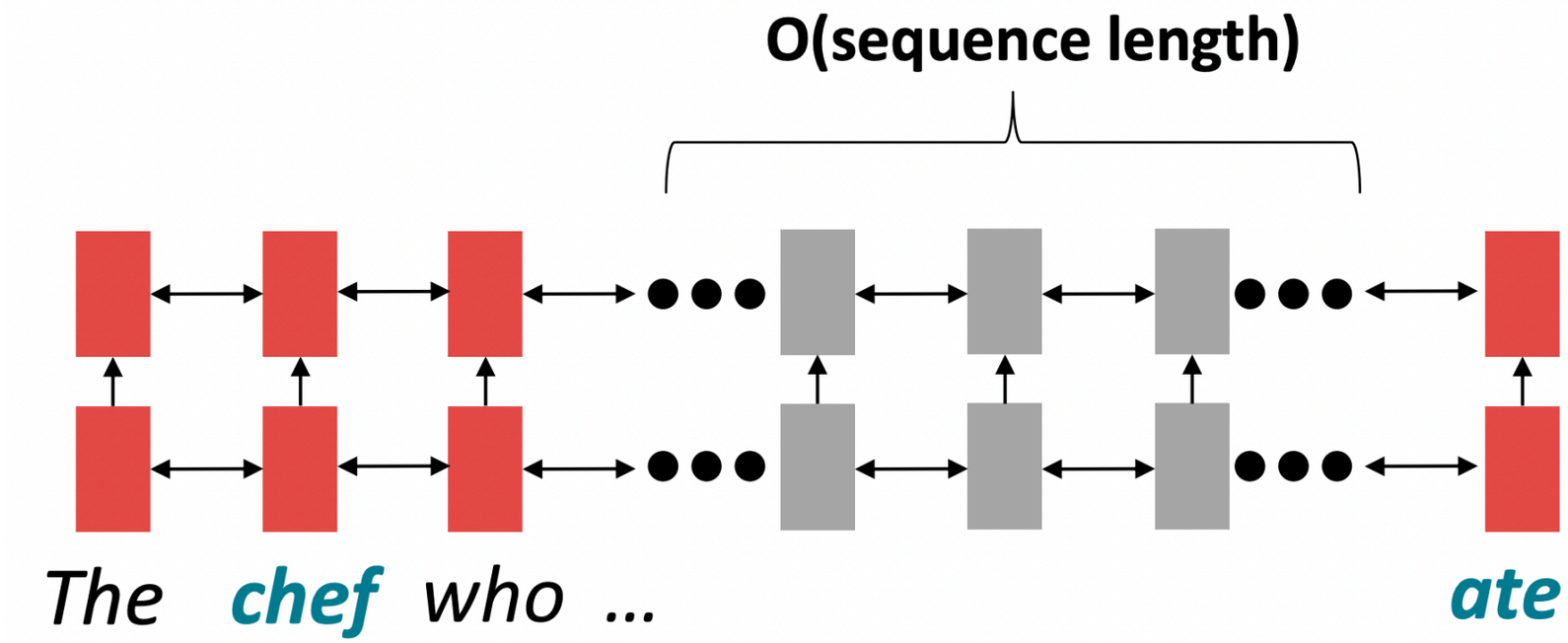
- Work reasonably well for long sequences
  (LSTMs and GRUs mitigate vanishing gradients)

- Don’t require a lot of memory

- Contextualization and linear interaction distance
  (near words affect each other)

Slide: Modified from John Hewitt

Drawbacks of RNNs

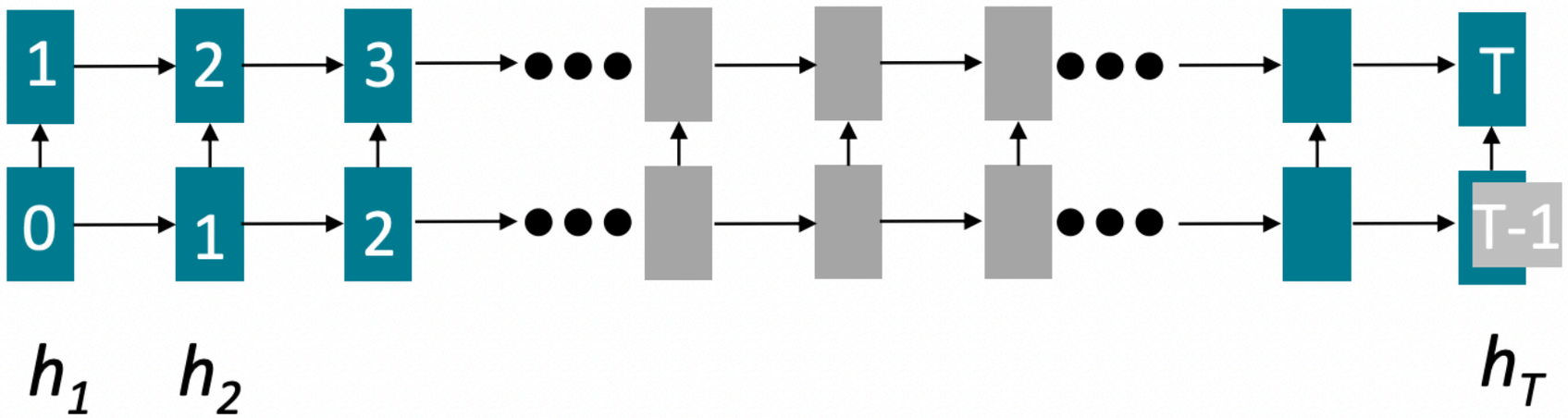


1. Long-range dependencies are hard to learn

2. Inductive bias: We force the sequential structure into our model, which may not be helpful

Slide: Modified from John Hewitt

Drawbacks of RNNs



Numbers indicate min # of steps before a state can be computed

Sequential processing means we **cannot parallelize** across time steps

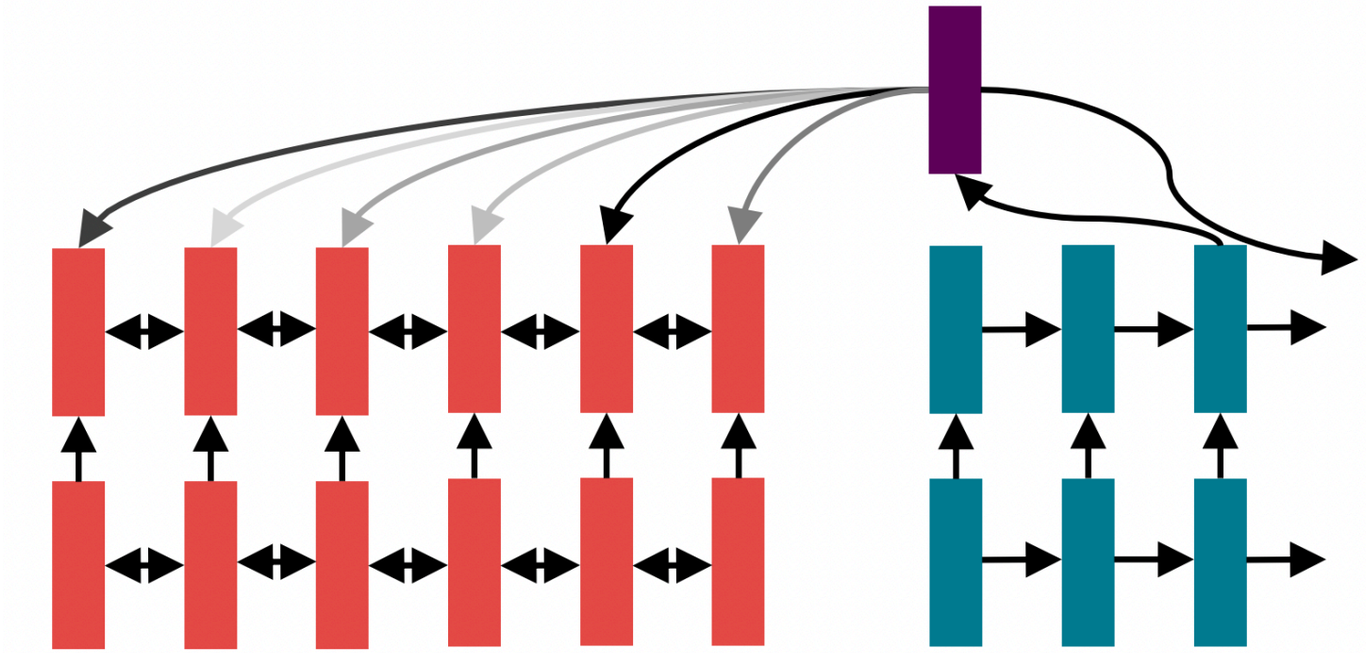
→ Our matrix multiplication hardware (GPUs / TPUs) aren't fully utilized

→ Inhibits training on larger datasets

What can we do instead?

Slide: Modified from John Hewitt

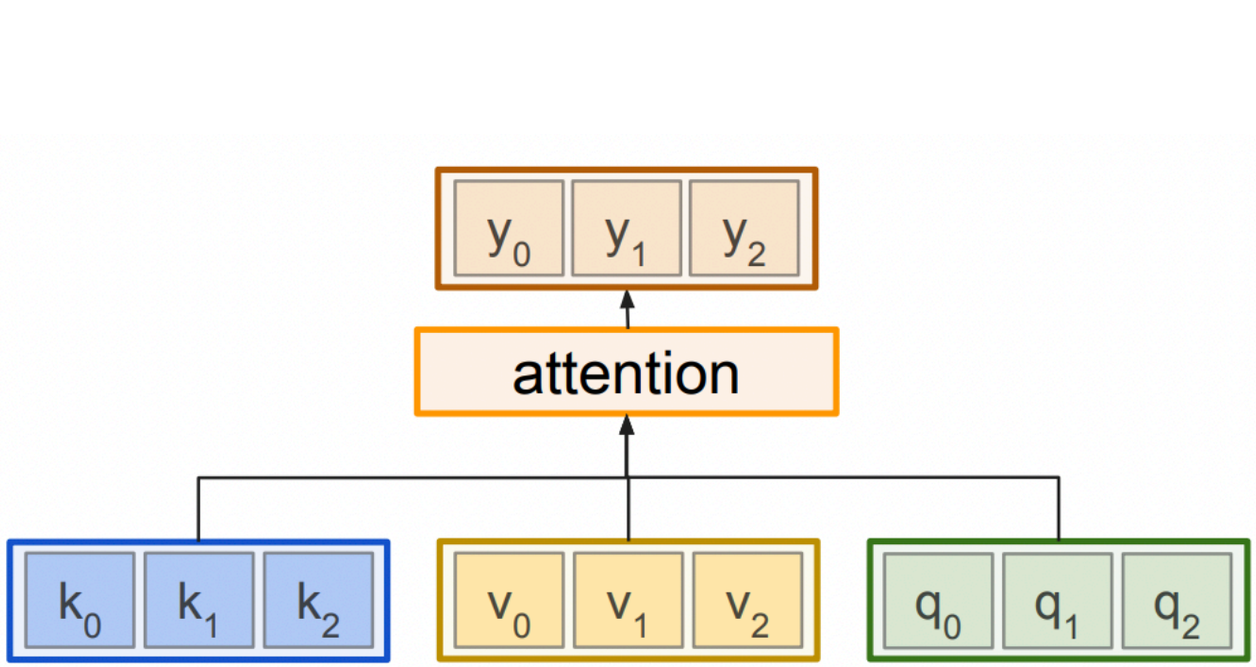
Attention in Seq2Seq RNNs



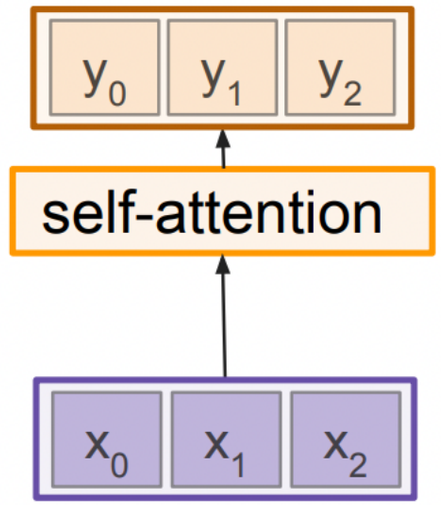
Word representations treated as **queries** to access (via **keys**) and incorporate information from a set of **values**

Concretely: Attend from **decoder** to **encoder** hidden states

General Attention vs Self-Attention

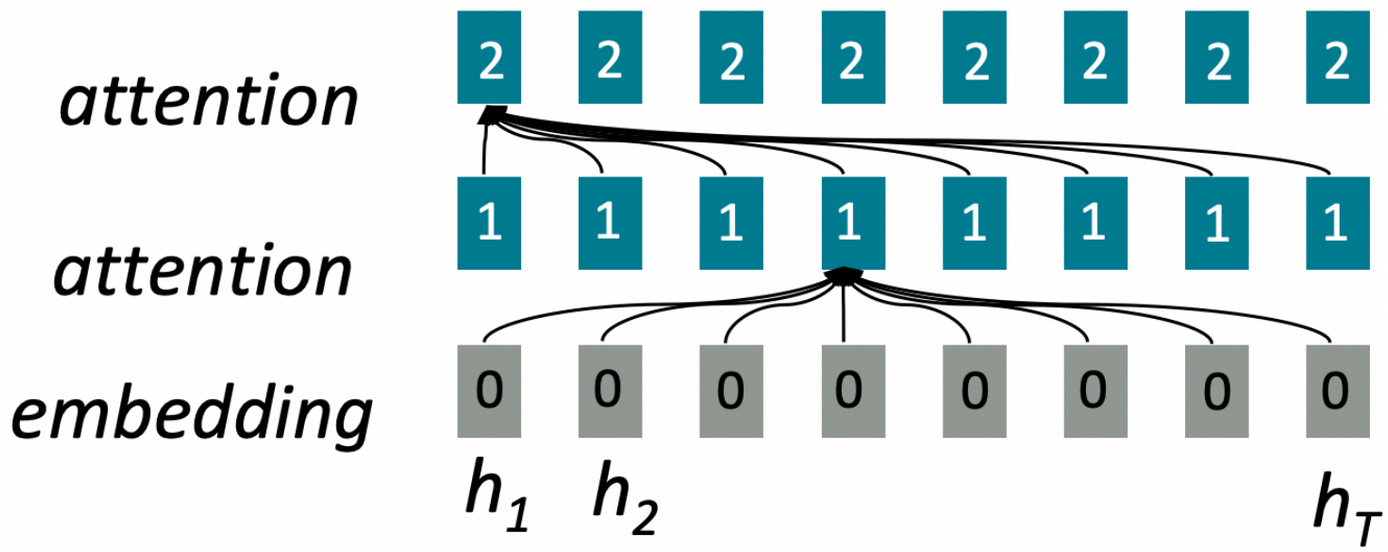


Encoder **Encoder** **Decoder**



Encoder or Decoder

Benefits of Self-Attention



All words attend to all words in previous layer; most arrows here are omitted

1. Now we **can parallelize** across timesteps

   → Enables us to train on large datasets, crucial for transfer learning!

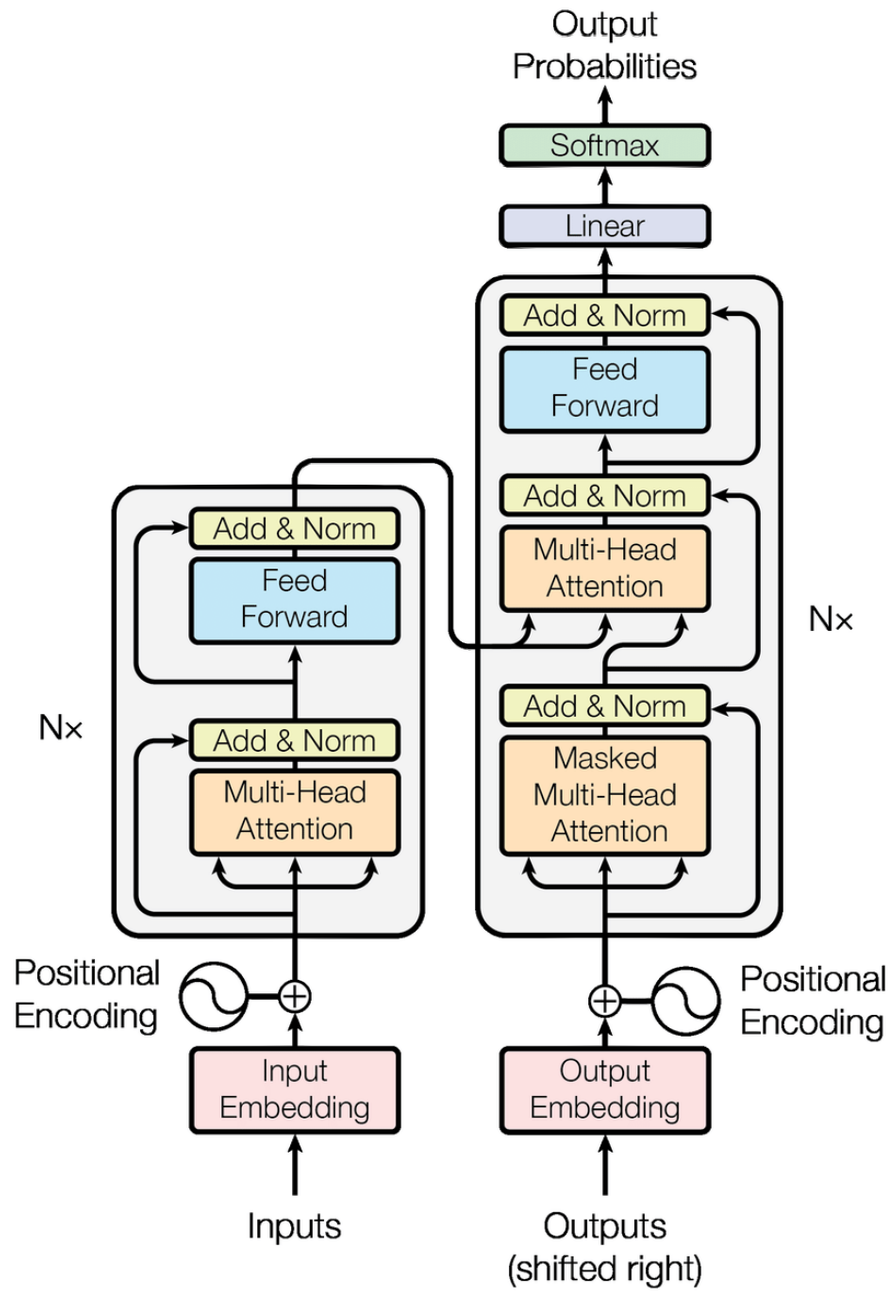
2. Maximum interaction distance is **O(1)**

Transformer Networks

Transformer Networks

No recurrence,
only blocks of self-attention

(And a couple of extra things to get it to work)

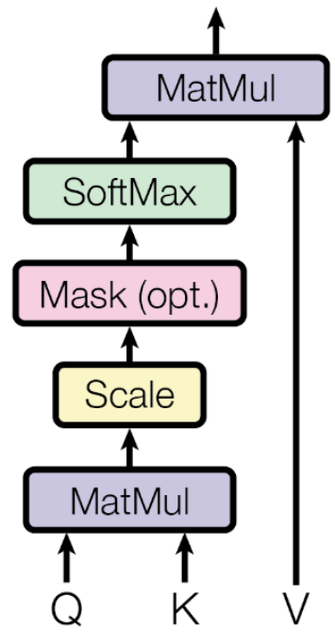


Vaswani et al. (NeurIPS 2017)

Scaled dot-product attention

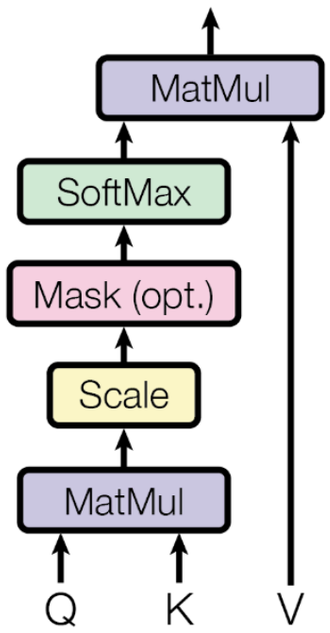
- Scaled-dot product attention

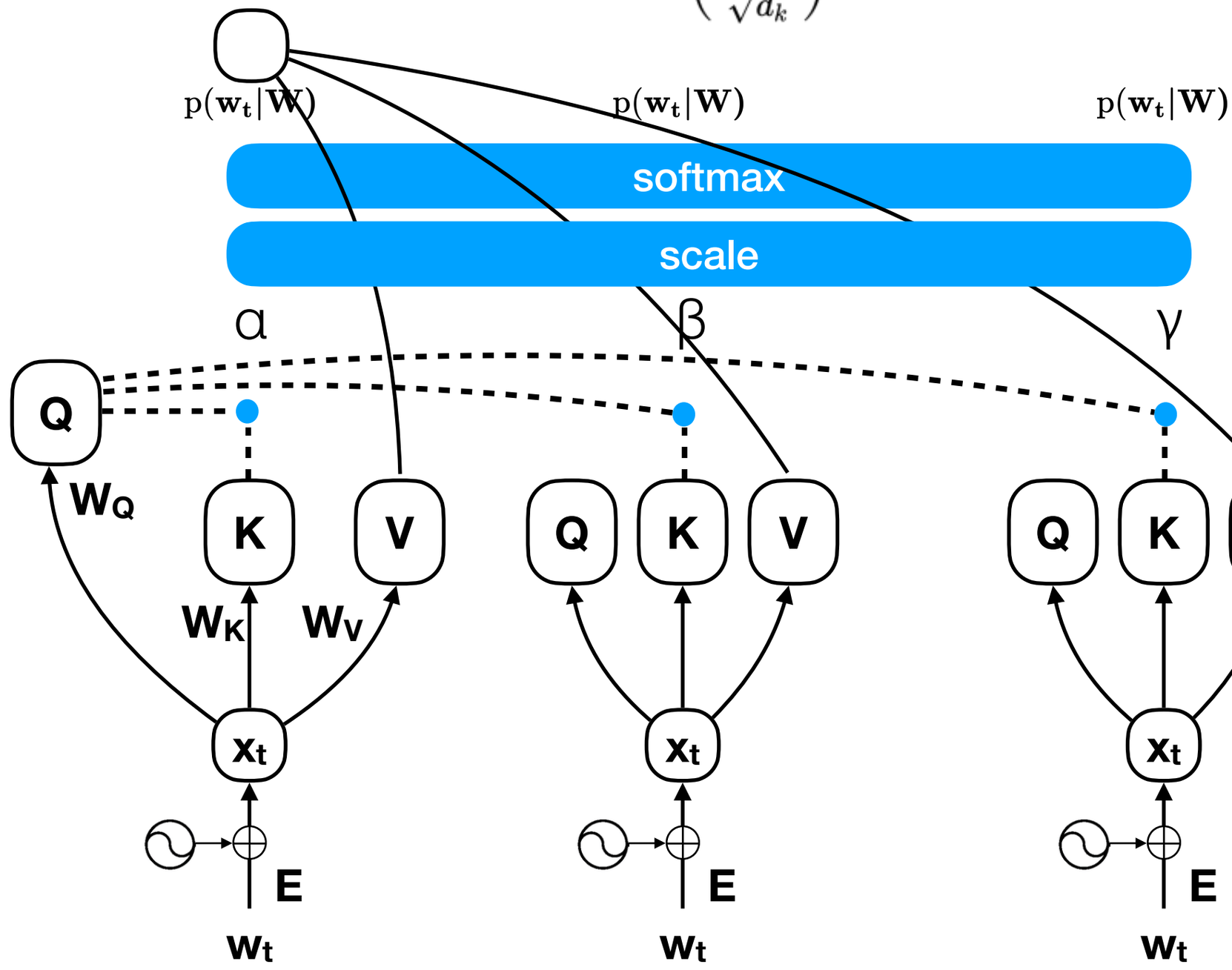
$$\text{Attention}\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

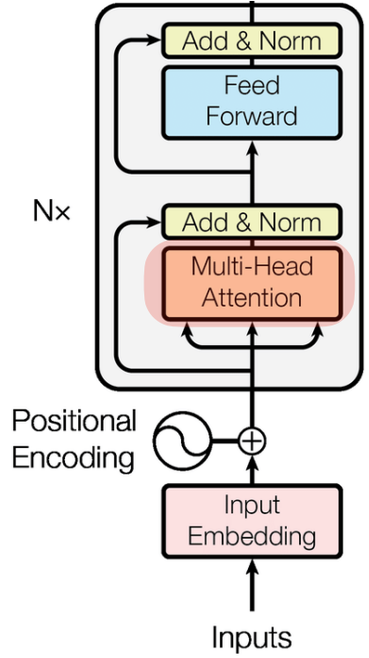


Scaled dot-product attention

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$







Dot-product attention

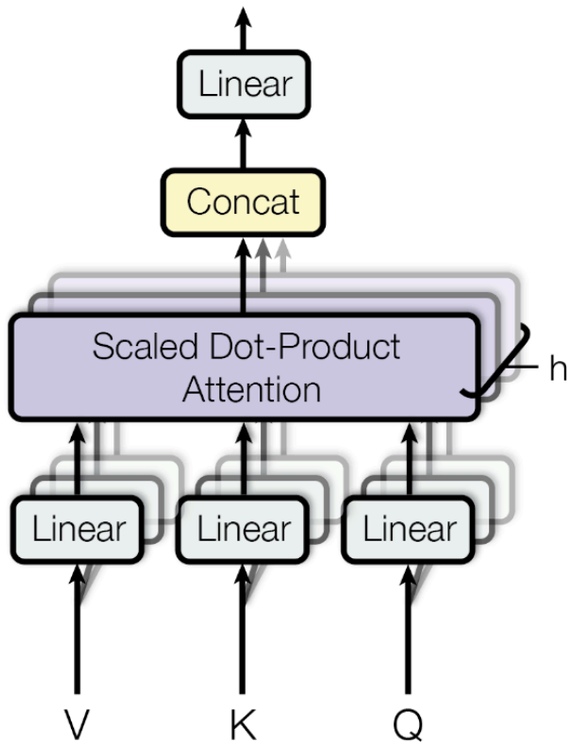
W A simple example

Multi-head Self-Attention

- Calculate $\mathbf{h}$ different self-attention vectors.

- Each attention “head” can learn different properties of the input

$$\begin{aligned}\text{MultiHead}\left(Q,K,V\right) &= \text{Concat}\left(\text{head}_1,\ldots,\text{head}_\text{h}\right)W^O \\ \text{where head}_\text{i} &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)\end{aligned}$$

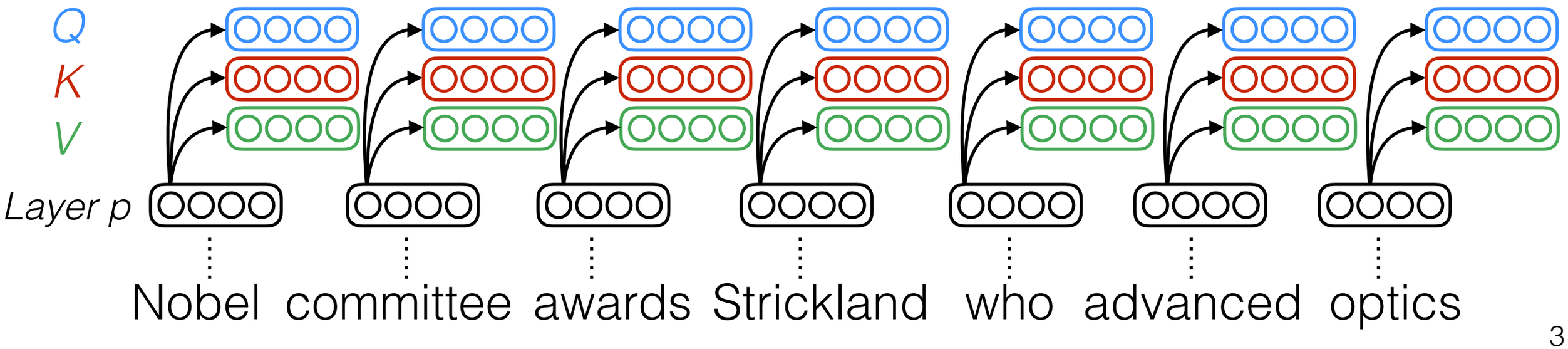


Self-Attention Walkthrough

Slides: Emma Strubell

[Vaswani et al. 2017]

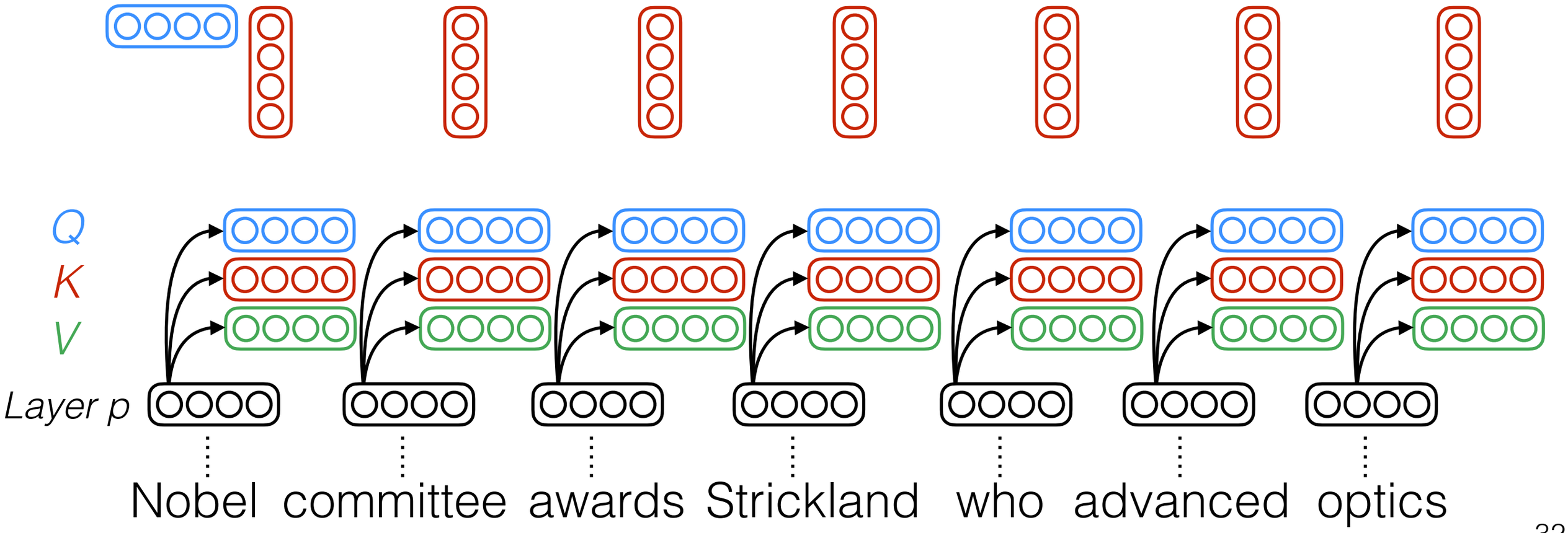
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

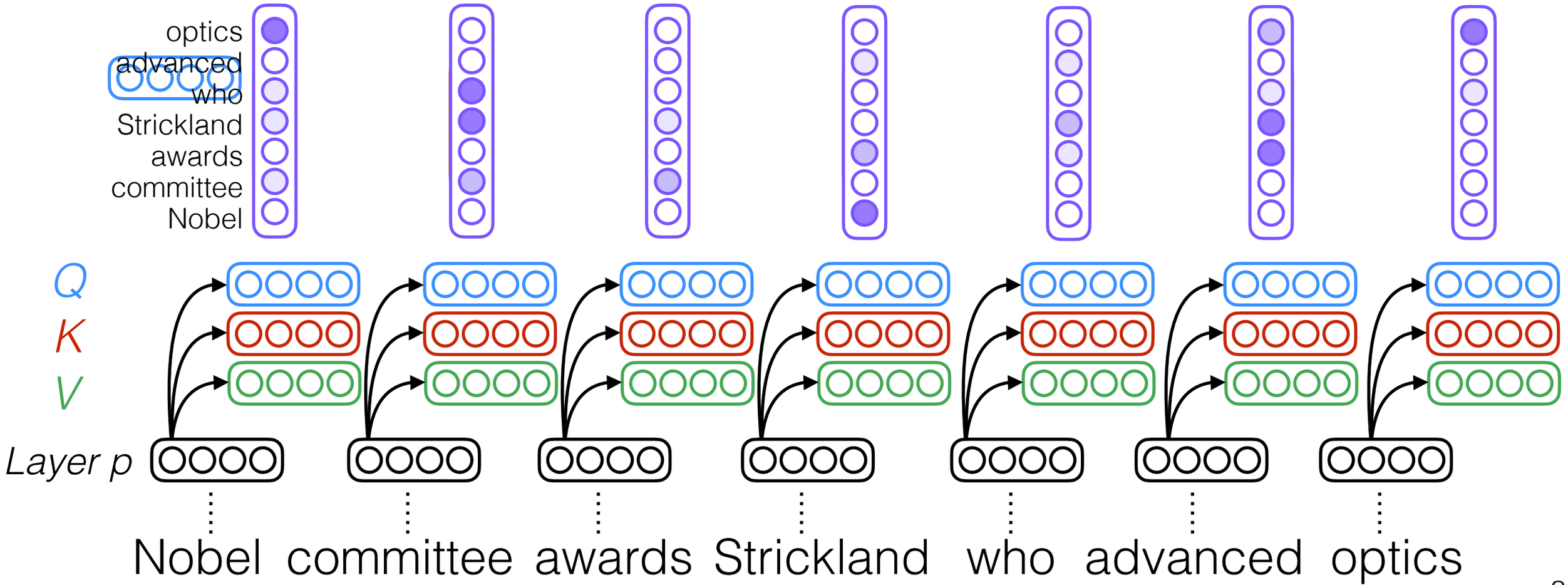
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

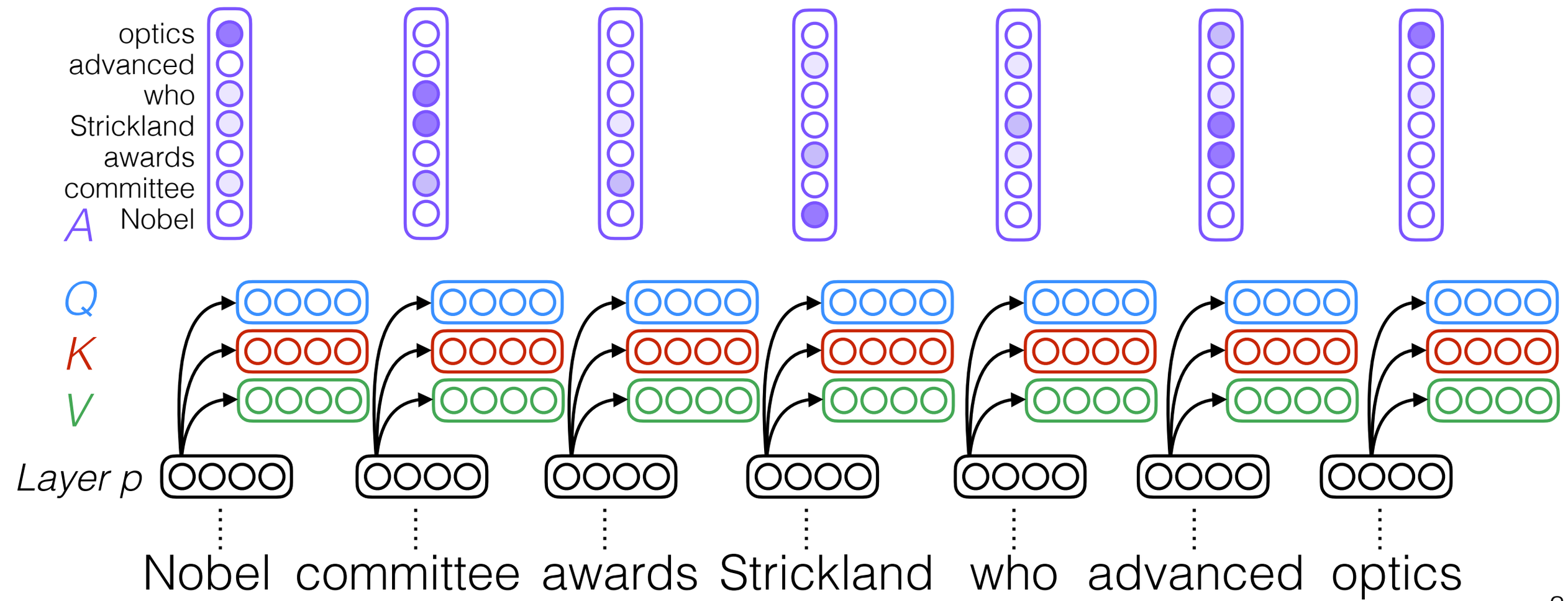
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

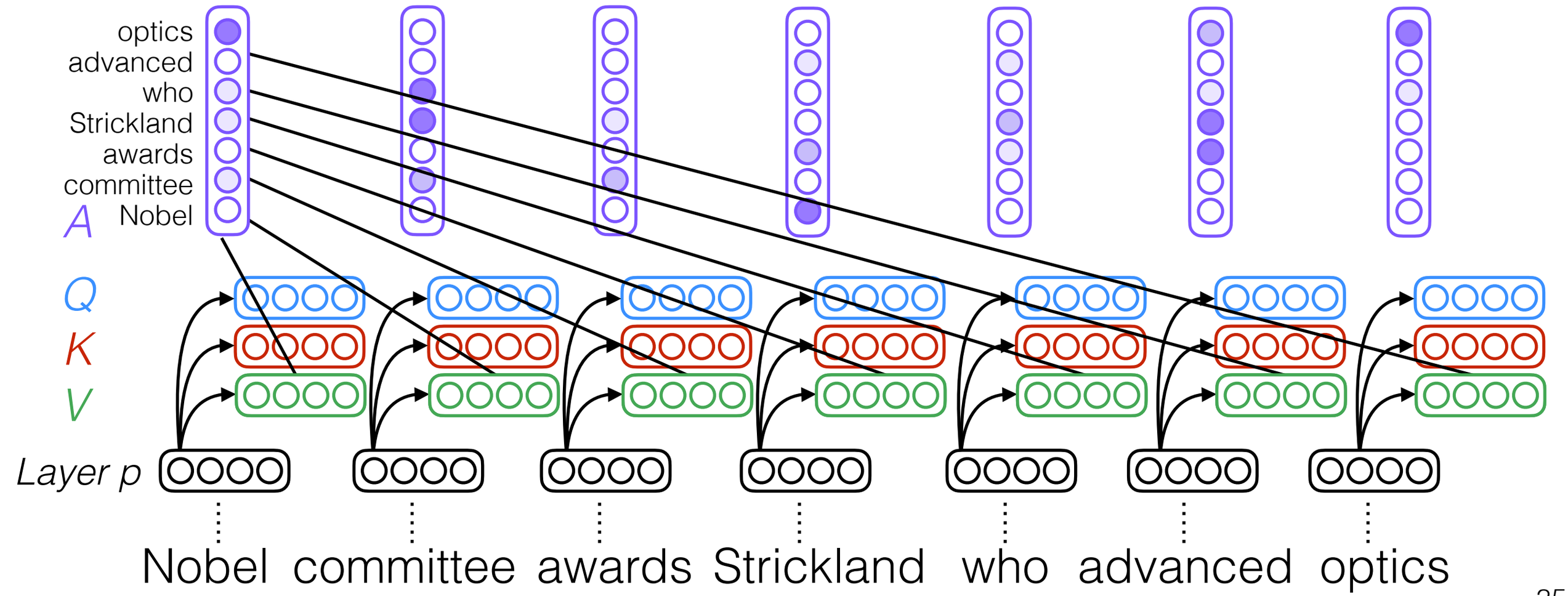
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

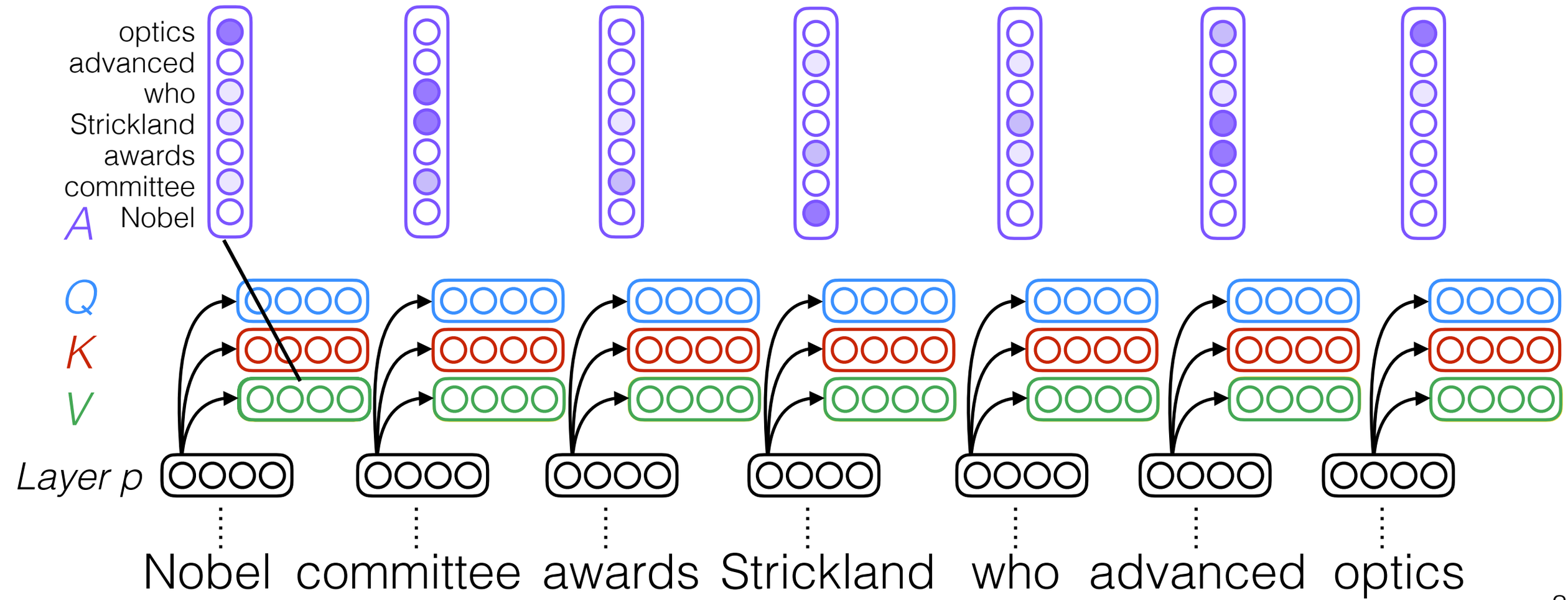
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

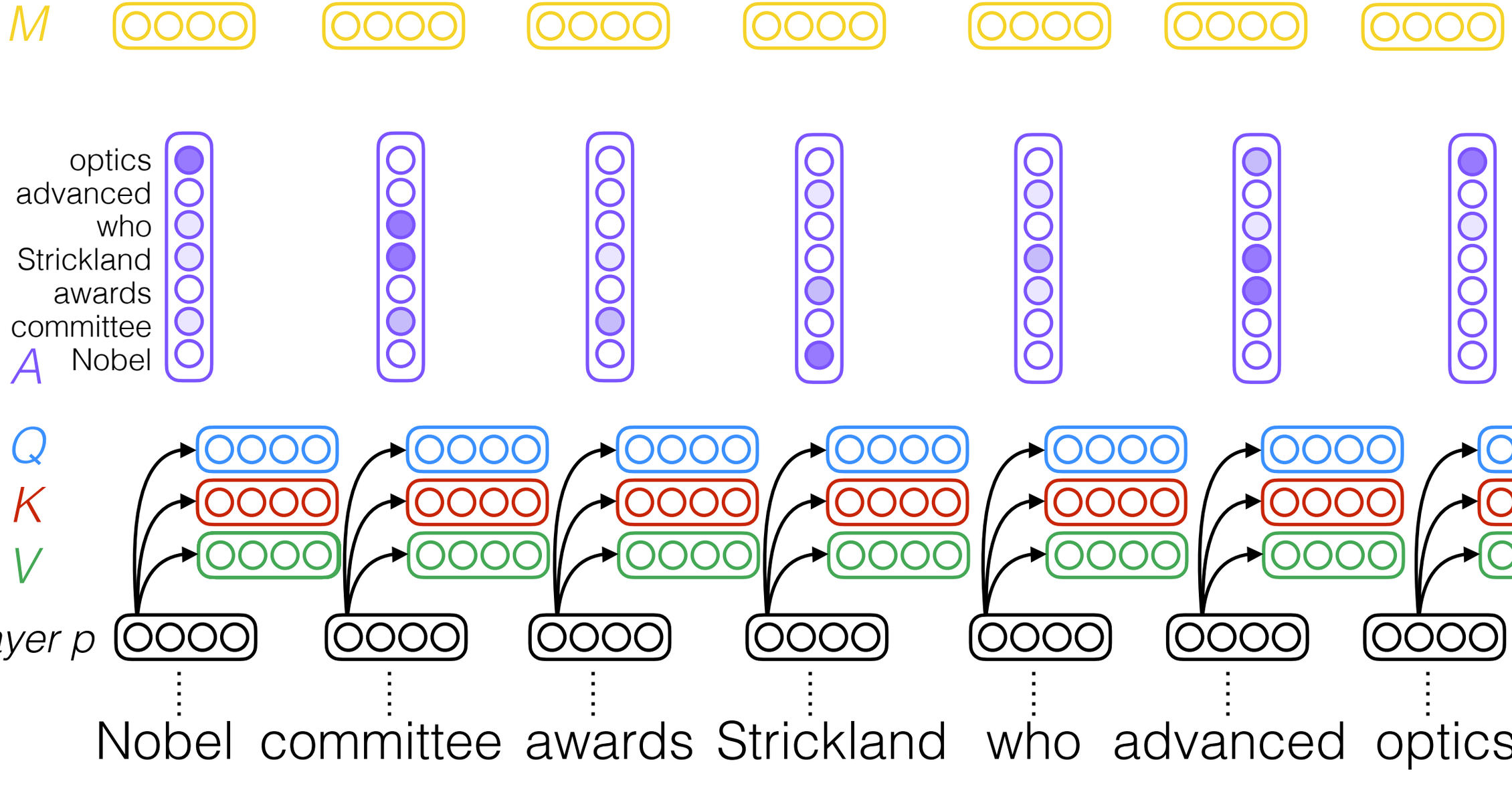
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

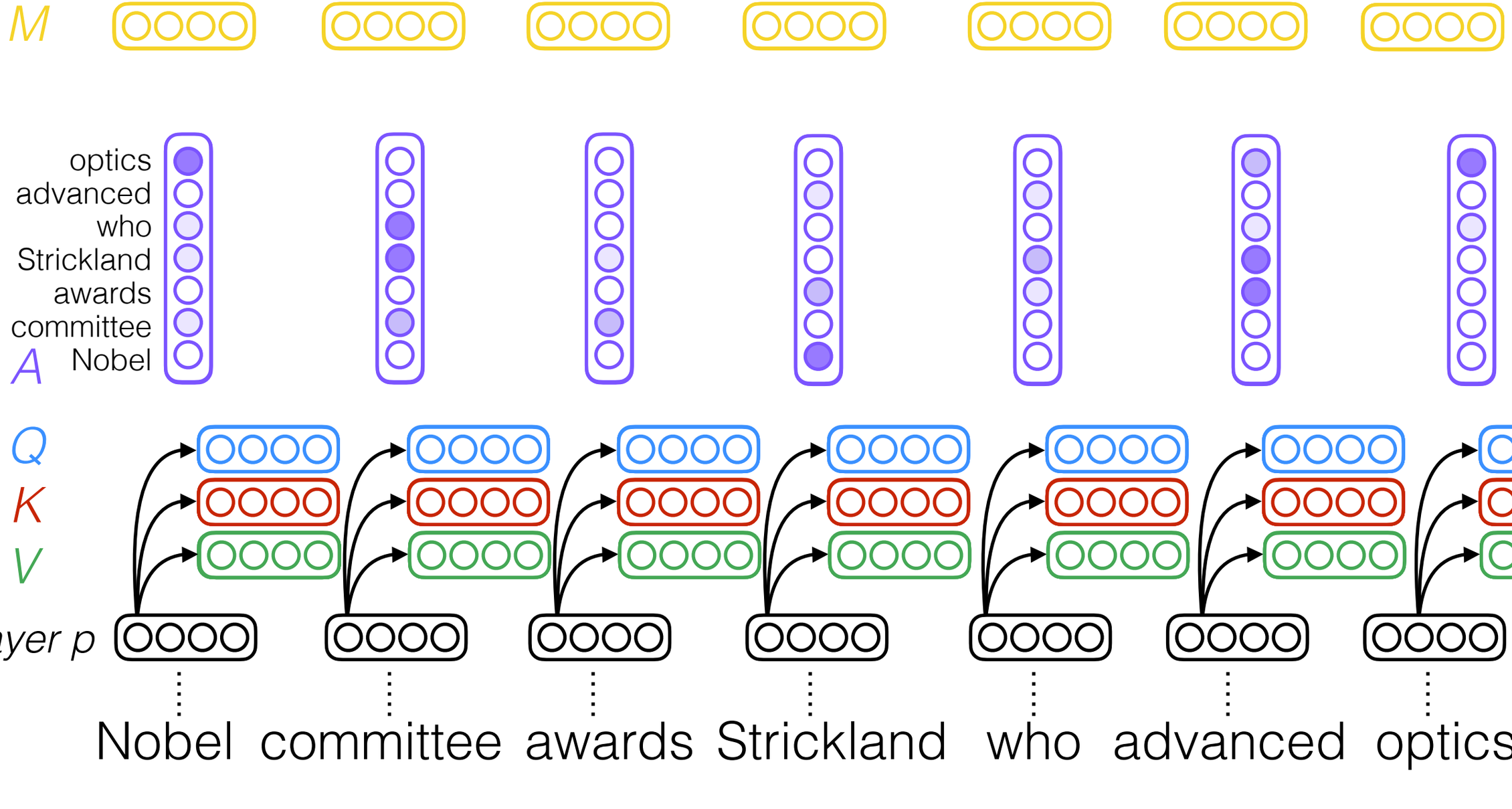
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

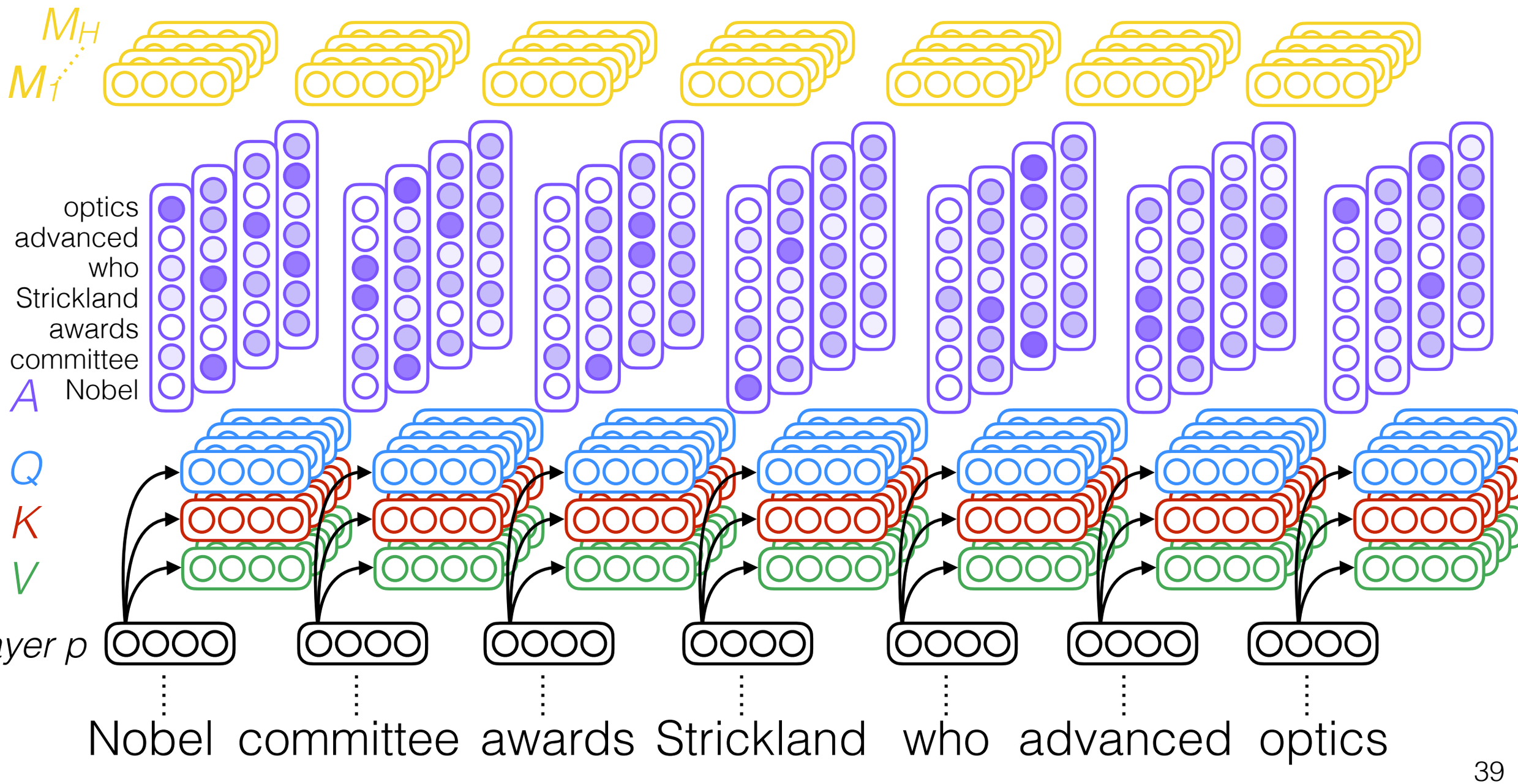
Self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

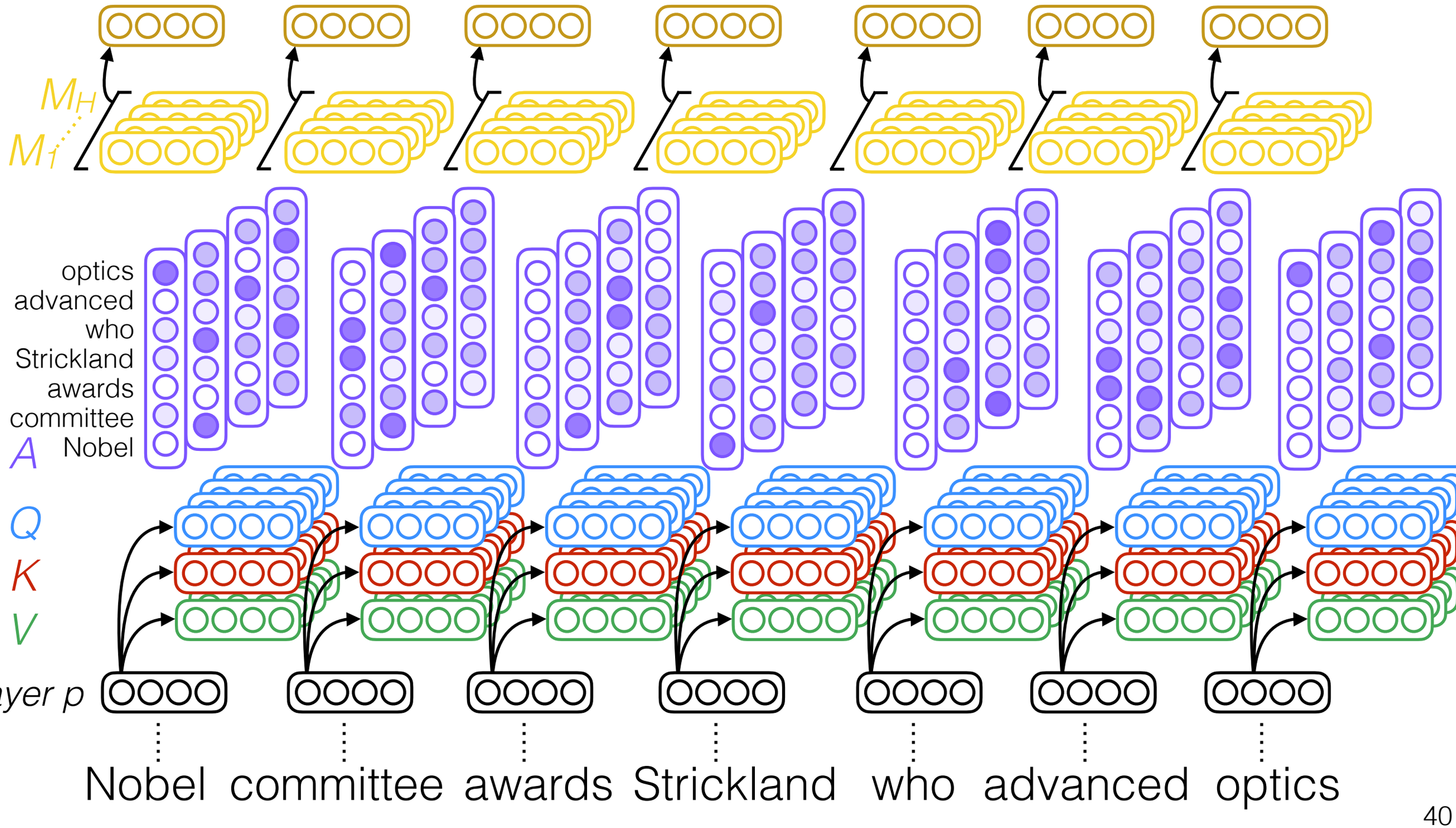
Multi-head self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

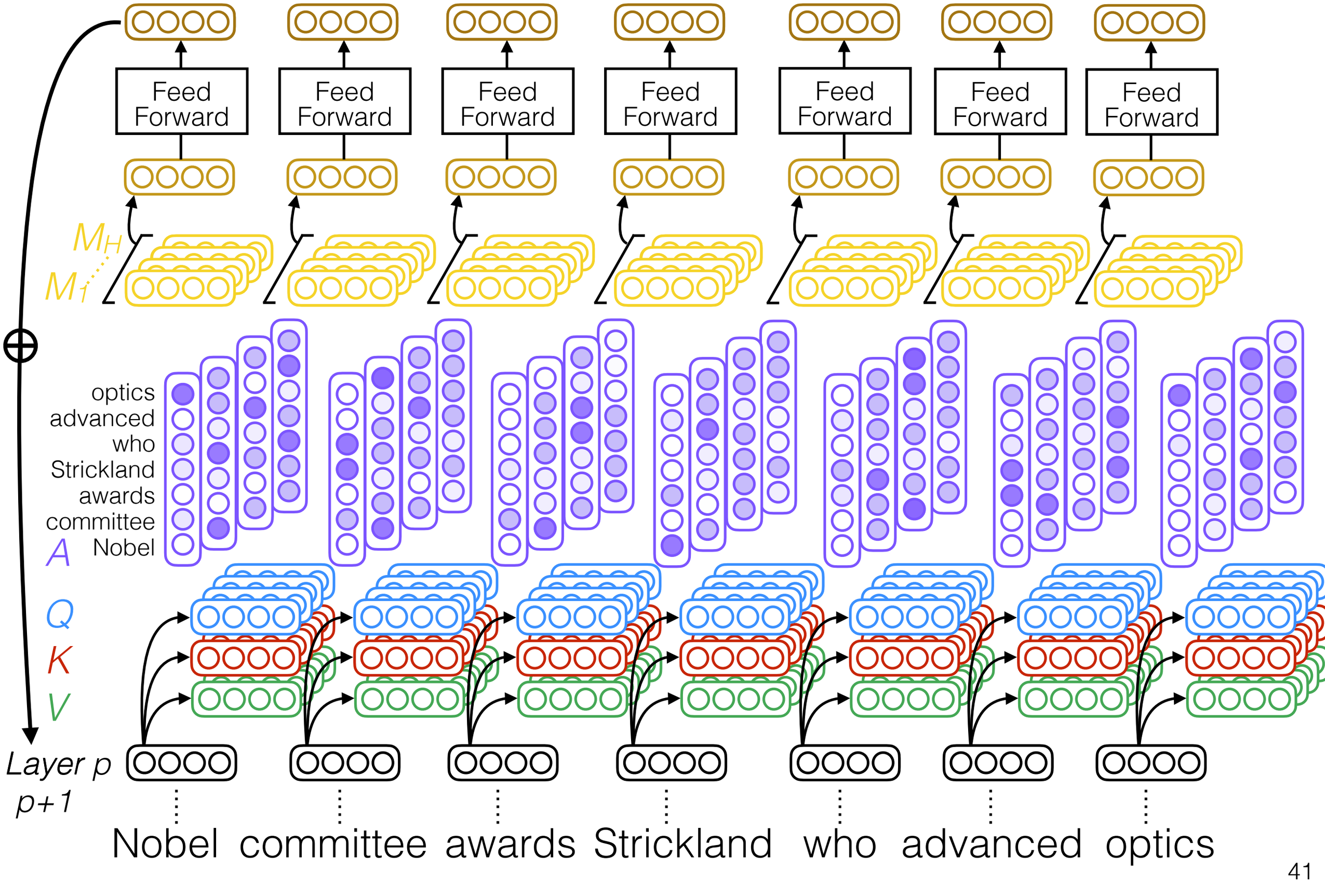
Multi-head self-attention



Slides: Emma Strubell

[Vaswani et al. 2017]

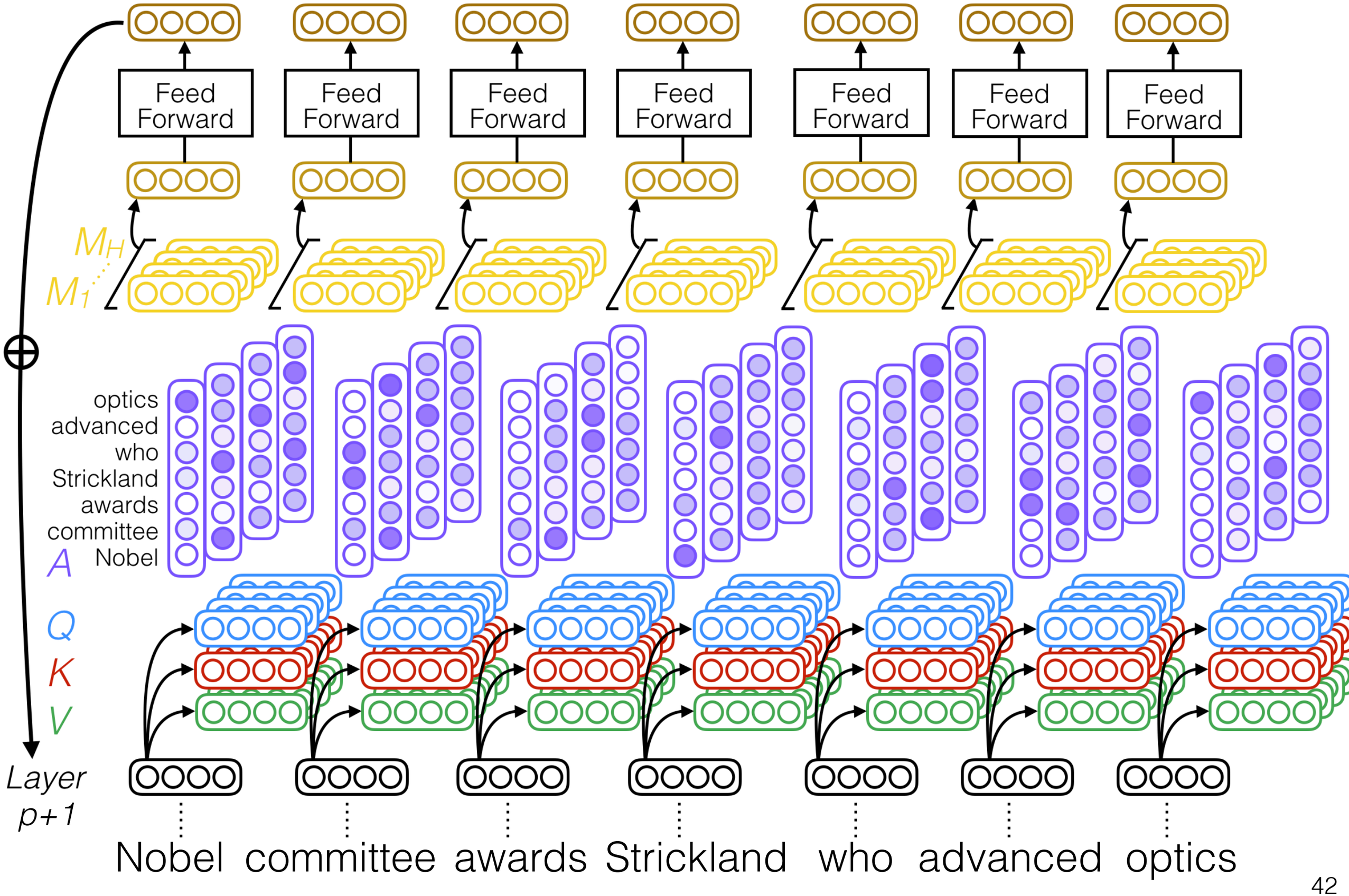
Multi-head self-attention



Slides: Emma Strubell

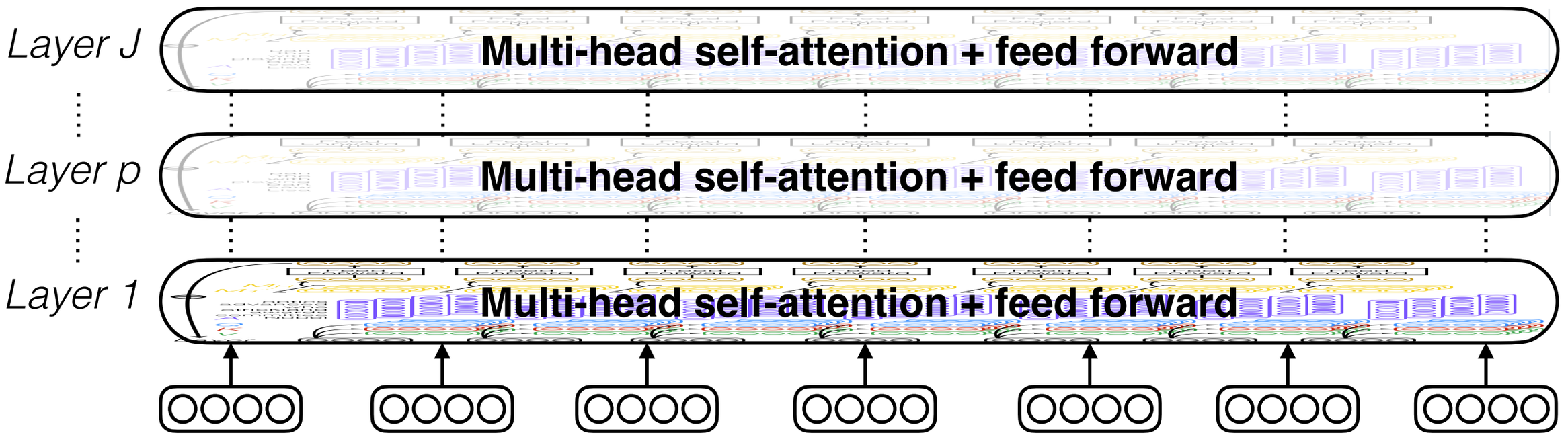
[Vaswani et al. 2017]

Multi-head self-attention

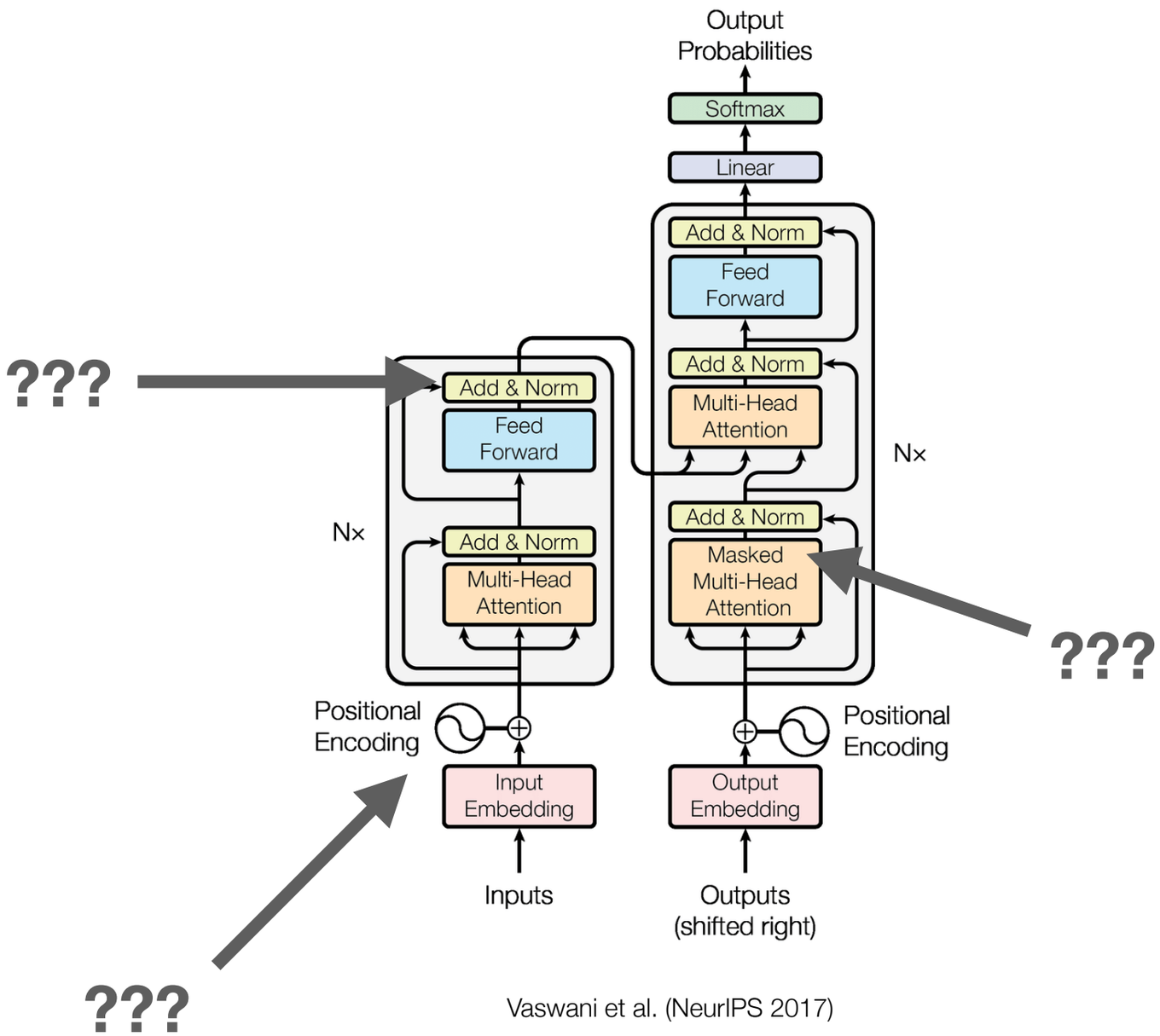


[Vaswani et al. 2017]

Multi-head self-attention

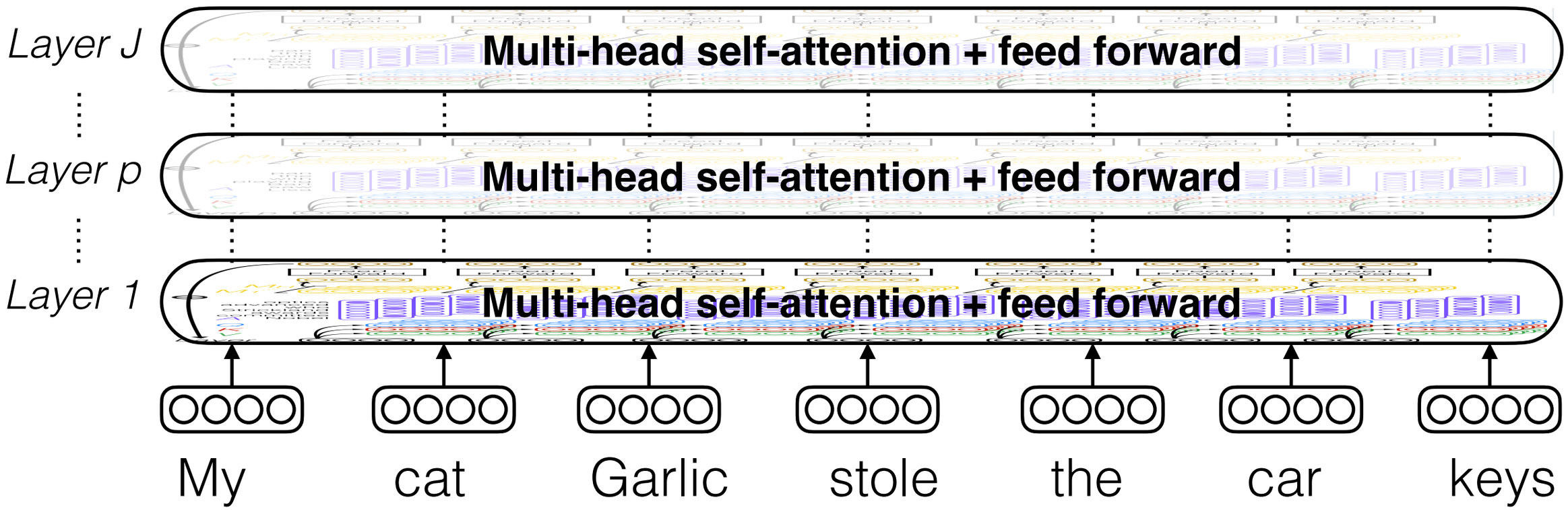


Yay, now we understand multi-head self-attention. Is this everything?



Vaswani et al. (NeurIPS 2017)

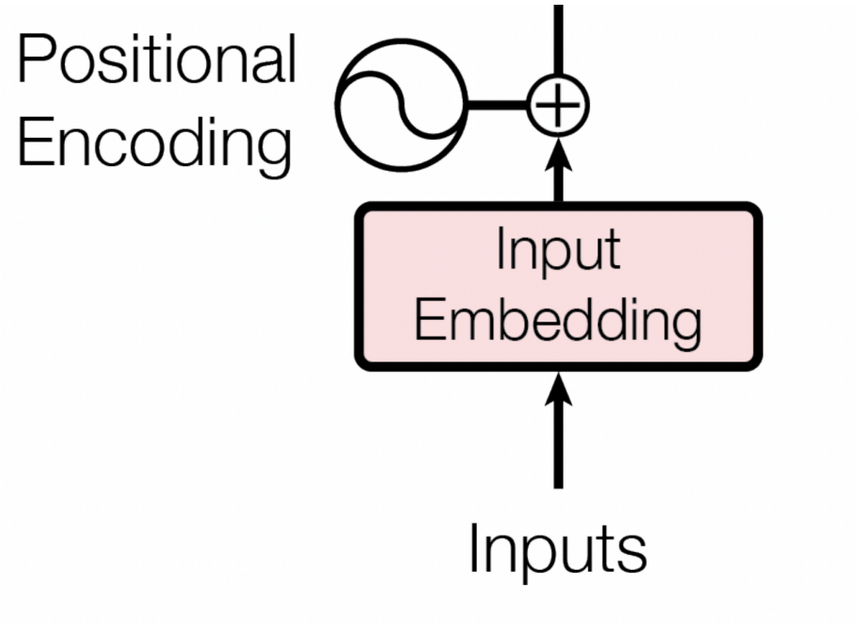
Injecting Order Information



This is like a bag of words! How is the model supposed to know the correct ordering?

Positional Encodings

We can simply add position representations to our old key, query, and value vectors (or directly to the embeddings)



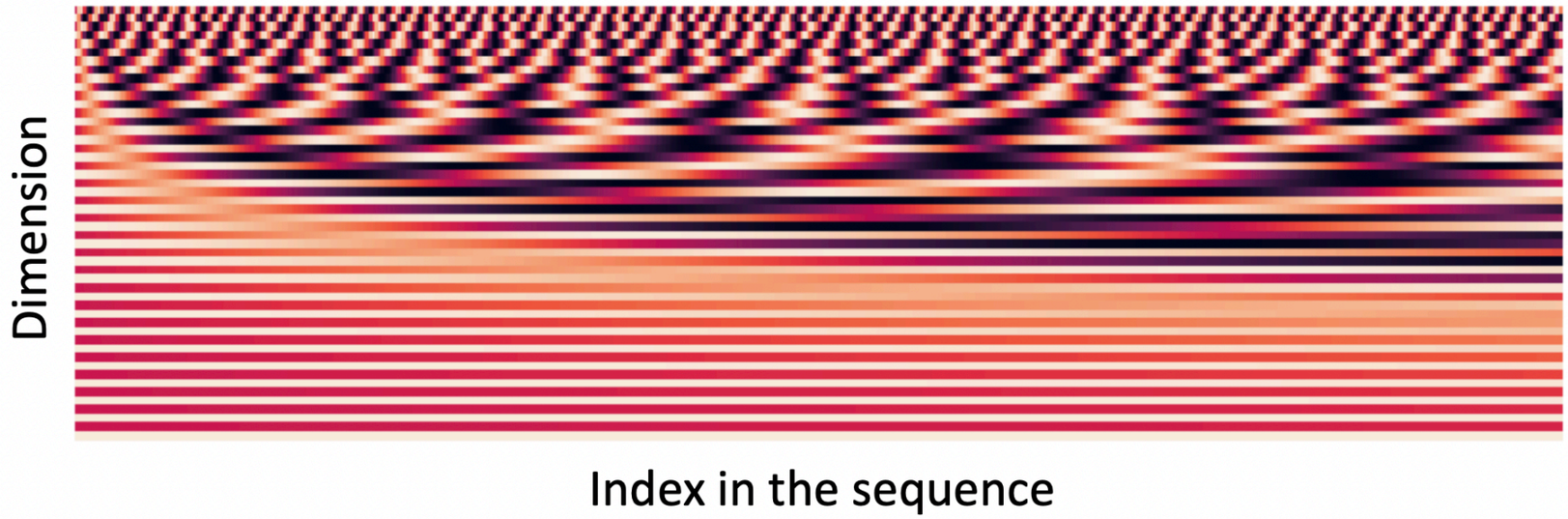
$$v_i = \tilde{v}_i + p_i$$
$$q_i = \tilde{q}_i + p_i$$
$$k_i = \tilde{k}_i + p_i$$

What is a suitable representation for p?

Slide: Modified from John Hewitt

Positional Encodings

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



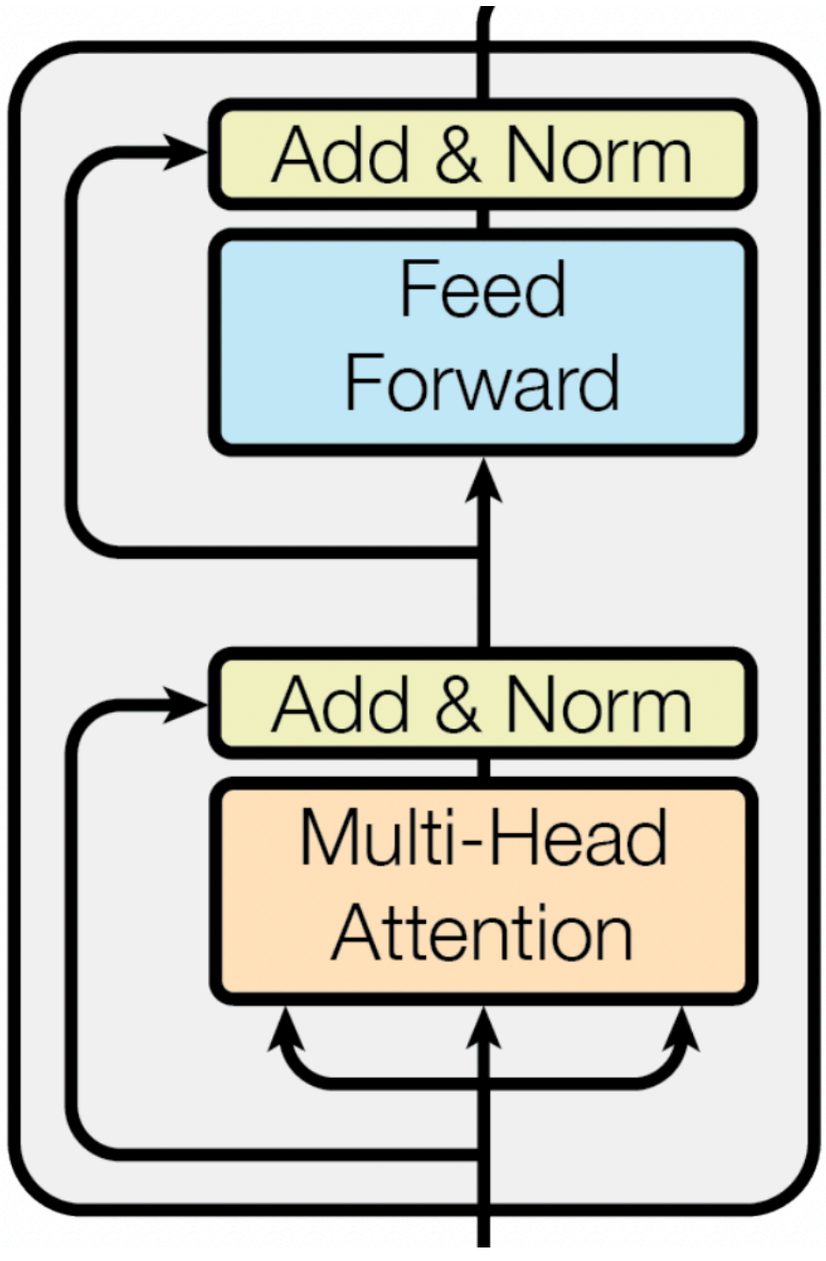
Fixed vector of sinusoidal functions of varying periods

Alternative (e.g. BERT): Just make each p_i a learnable parameter and let the model figure it out on its own

Key components to stable training: Residual Connections & Layer Normalization

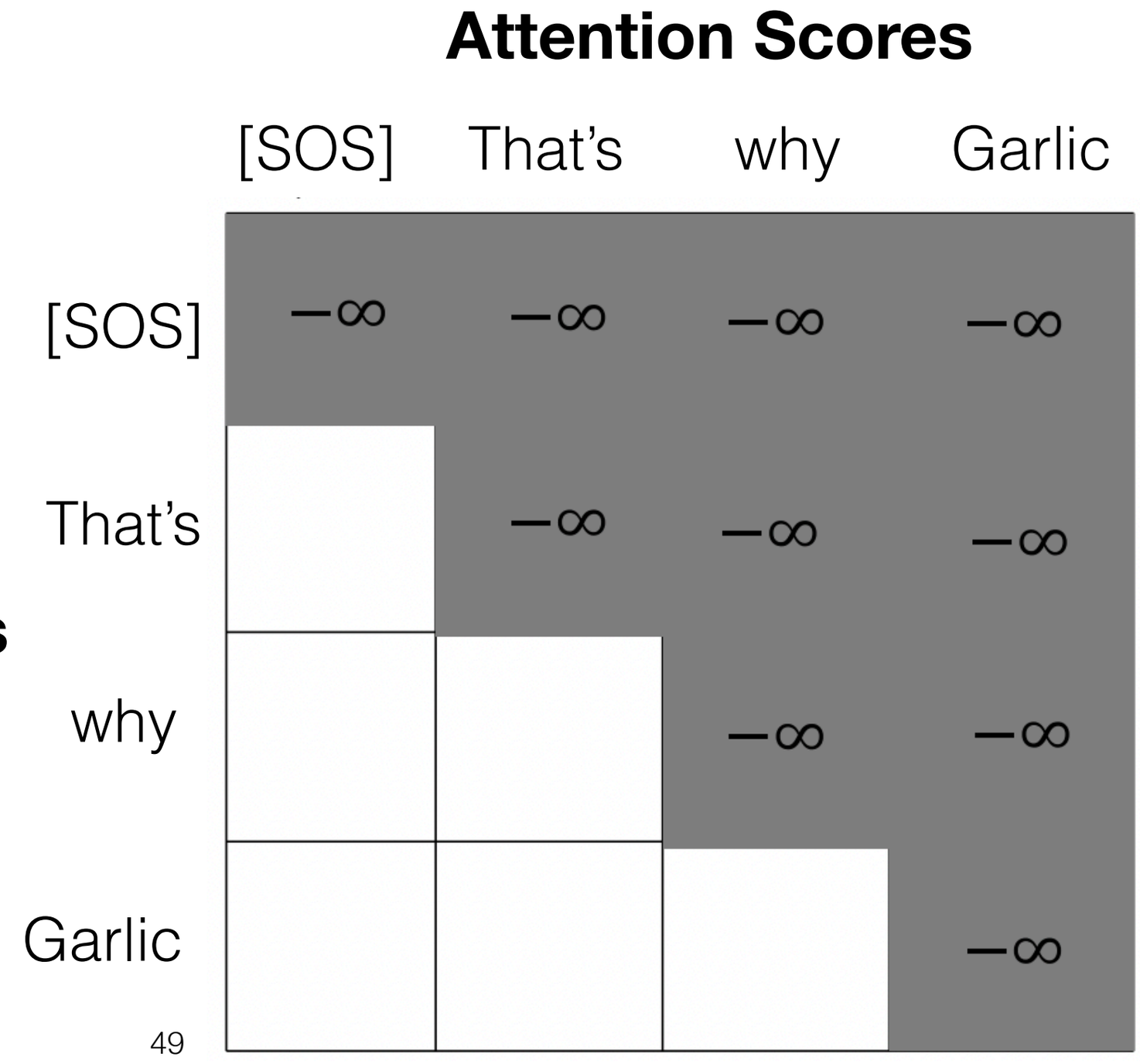
Residual Connections (He et al. 2016, ResNets) improve gradient flow in deep networks and make it easy to learn the identity if necessary

Layer Normalisation (Ba et al. 2018) prevents inputs to each layer from shifting too much by normalizing to mean = 0 and std =1



Prevent decoder from looking into the future

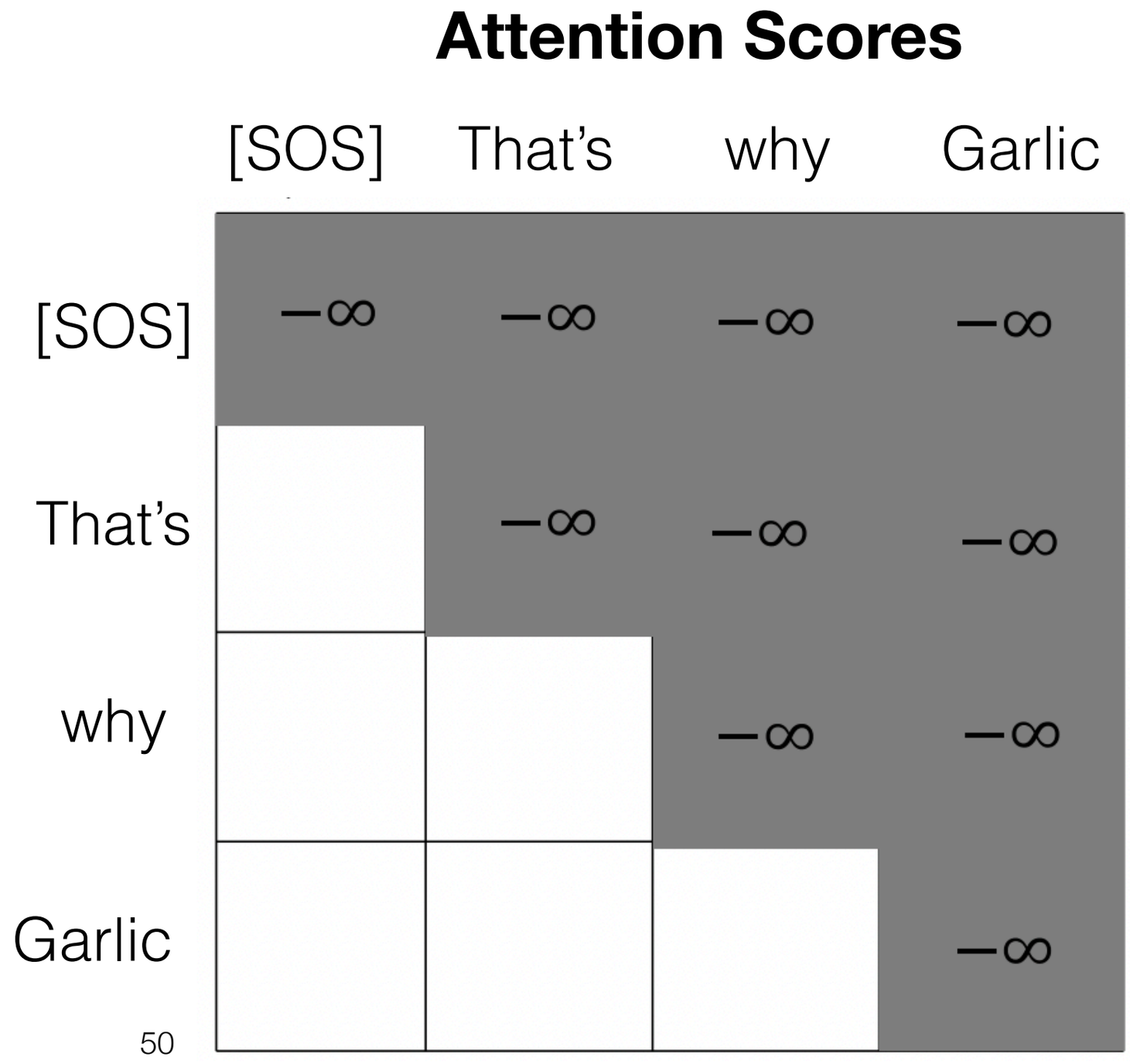
Q: Why do this instead of changing set of queries, keys, and values?



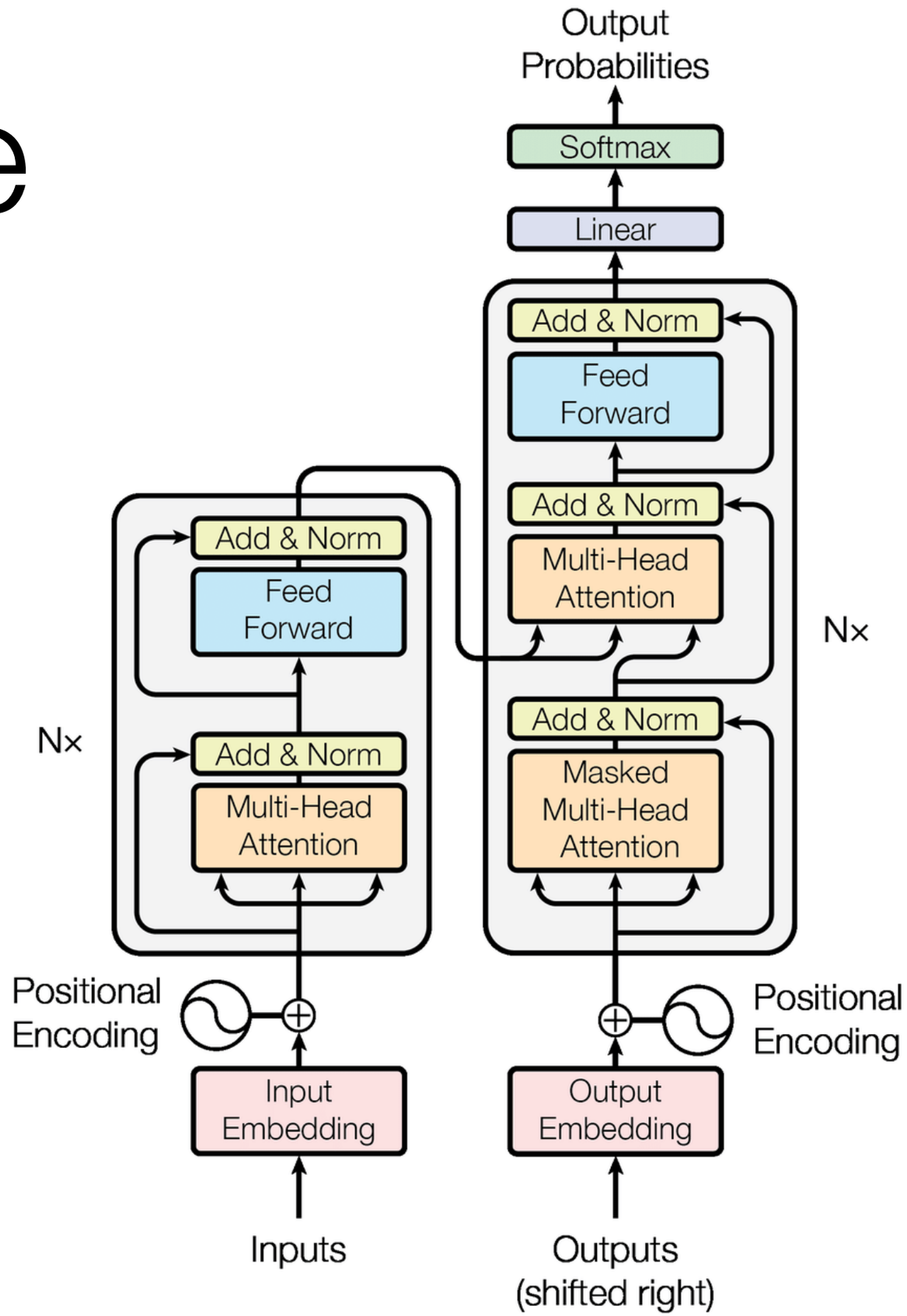
Prevent decoder from looking into the future

Q: Why do this instead of changing set of queries, keys, and values?

A: This approach keeps dimensions the same, so parallelization is unaffected



Now we're all set



Vaswani et al. (NeurIPS 2017)

Original Transformer Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Vaswani et al. (NeurIPS 2017)

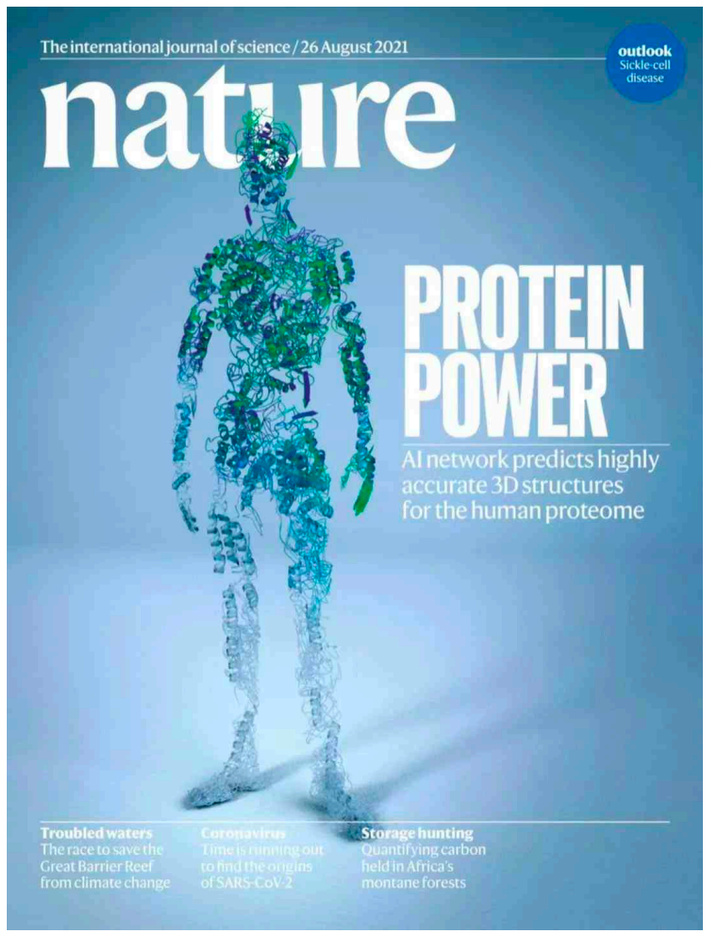
Nothing (currently) beats transformers

SuperGLUE GLUE

	Rank	Name	Model
✚	1	Liam Fedus	ST-MoE-32B
	2	Microsoft Alexander v-team	Turing NLR v5
	3	ERNIE Team - Baidu	ERNIE 3.0
	4	Yi Tay	PaLM 540B
✚	5	Zirui Wang	T5 + UDG, Single Model (Google Brain)
✚	6	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4
	7	SuperGLUE Human Baselines	SuperGLUE Human Baselines

https://super.gluebenchmark.com/leaderboard

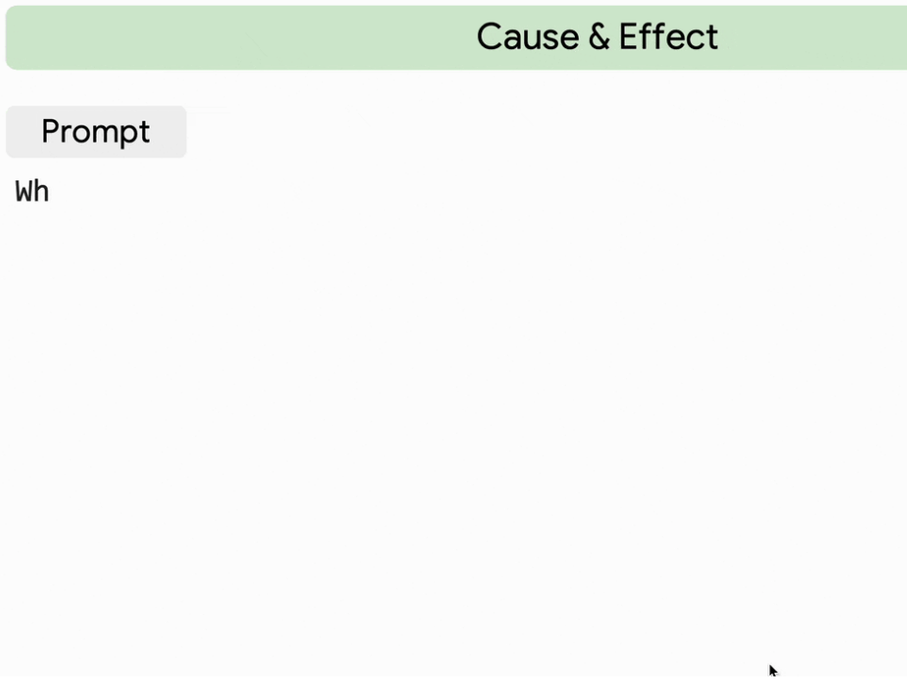
Transformers are everywhere



AlphaFold2
https://www.nature.com/articles/s41586-021-03819-2

Image generated from text prompt "*A magic space capybara wearing a glittering tutu and blessing you, digital art*"

OpenAI DALL·E 2
https://openai.com/dall-e-2/



PaLM
https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

BERT and Friends

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

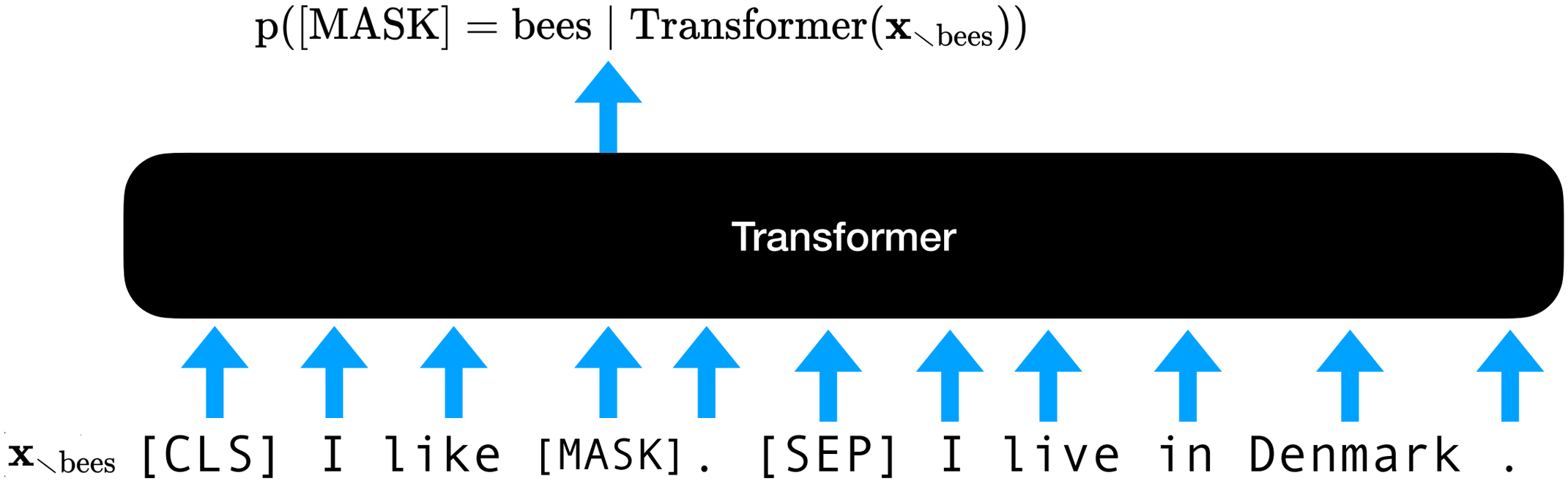
- Bi-directional Transformer Network
- Simultaneously process one *or* two sentences

- Two tasks:
  - Masked Language Modelling: Randomly mask 15% of the hidden states and train the model to predict the masked words
  - Next Sentence Prediction: Train binary classifier to predict if Sentence B is actually the sentence that proceeds Sentence A.

Devlin et al. 2018

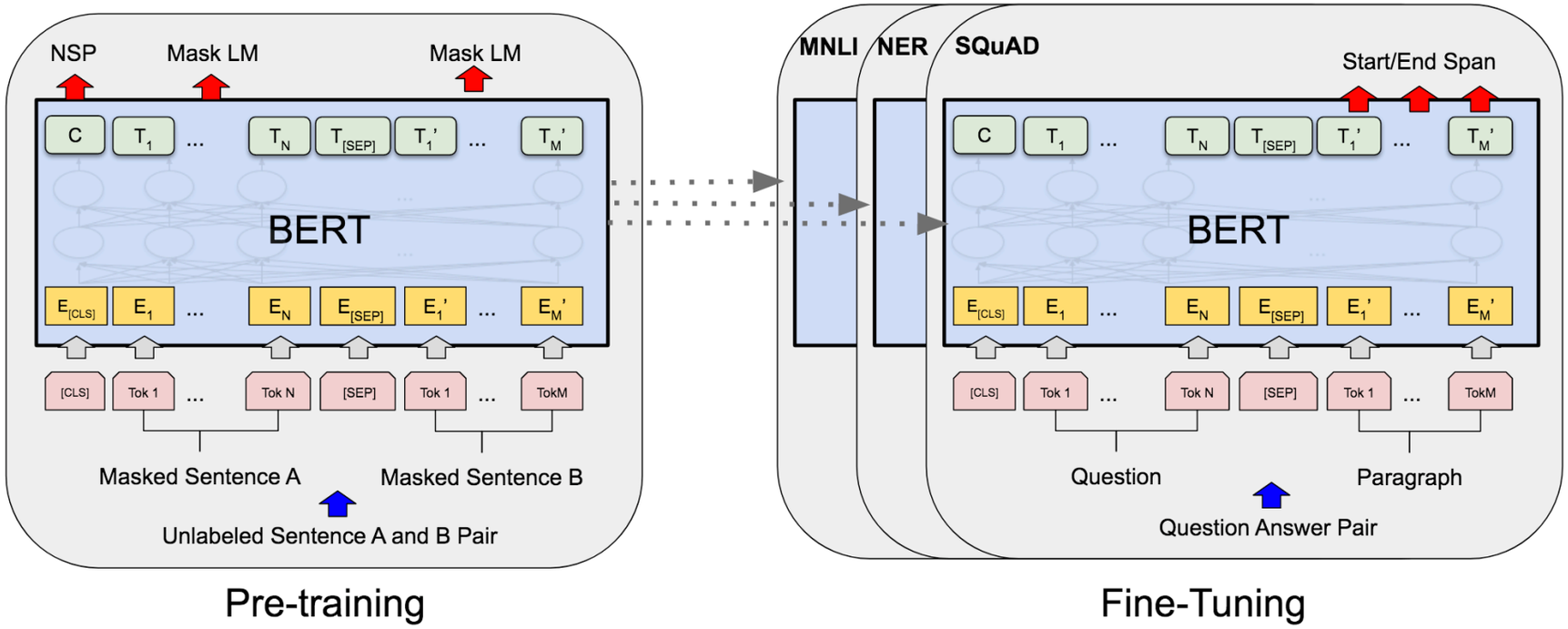
Masked Language Modelling

- Randomly mask 15% of the input words and train the model to predict the identity of the masked words



Devlin et al. 2018

BERT and Transfer Learning



	GLUE	SQuAD	SWAG	NER
SOTA	75.1	87.9	78.0	93.1
BERT	**82.1**	**90.9**	**86.3**	92.8

Figure: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

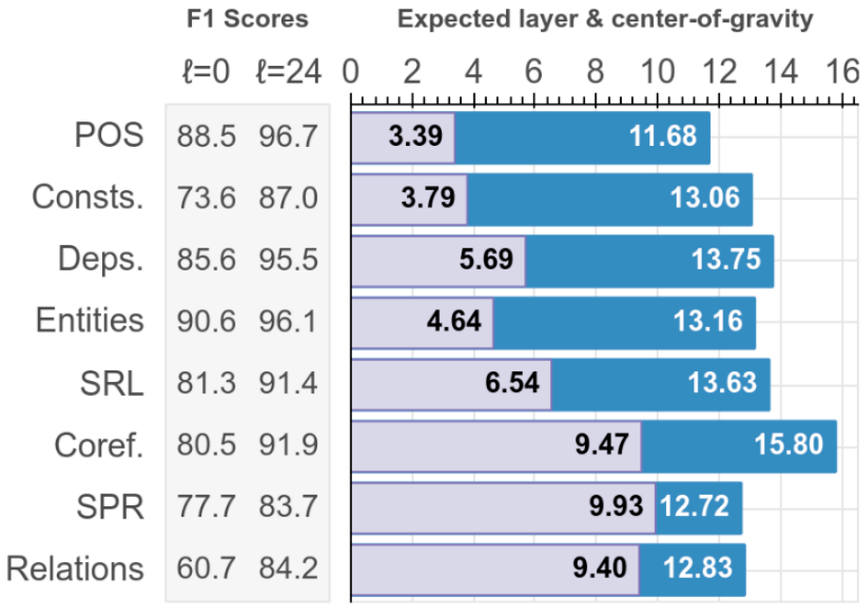
What does BERT encode?

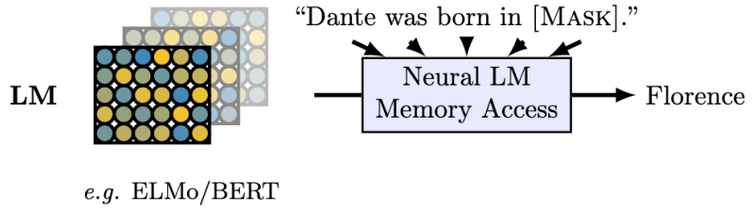
- Many follow-up studies in this area
  - Syntactic knowledge

    Tenney et al. (2019), Hewitt and Manning (2019)
  - Semantic knowledge

    Ettinger (2019), Wallace et al. (2019)
  - World knowledge

    Petroni et al. (2019), Forbes et al. (2019)
  - Where is this encoded?
    - Embeddings? Self-attention?

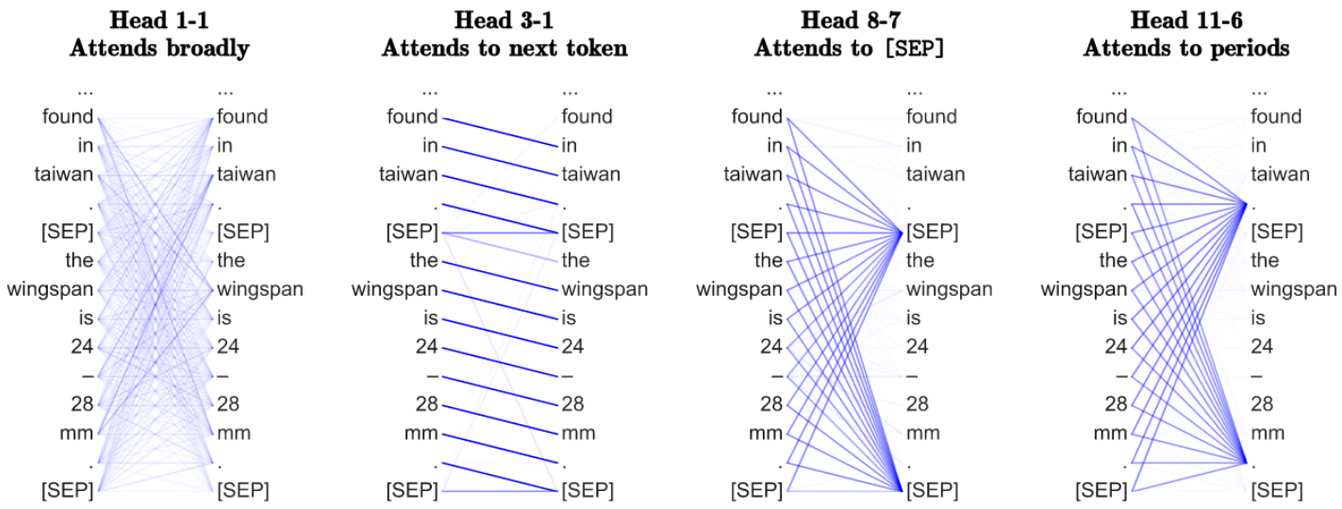




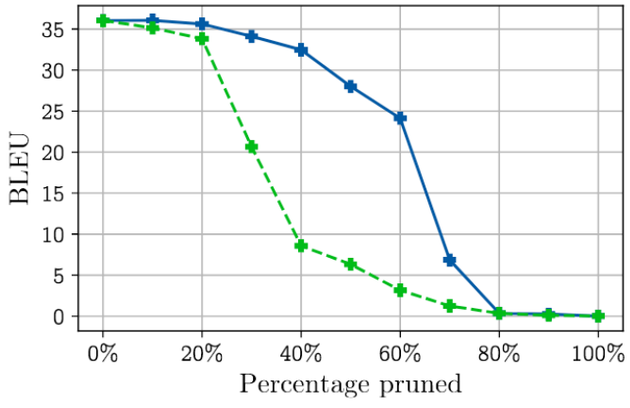
Rogers et al. 2020; Figures: Tenney et al. 2019, Petroni et al. 2019

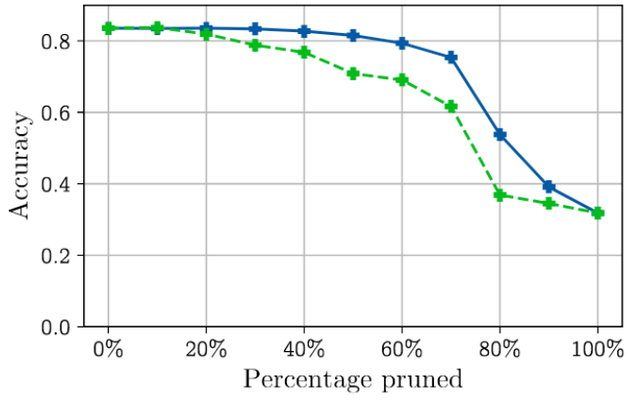
What do we know about Multi-head Self-Attention

- Some heads have very specialised behaviour



- It's possible to iteratively prune attention heads





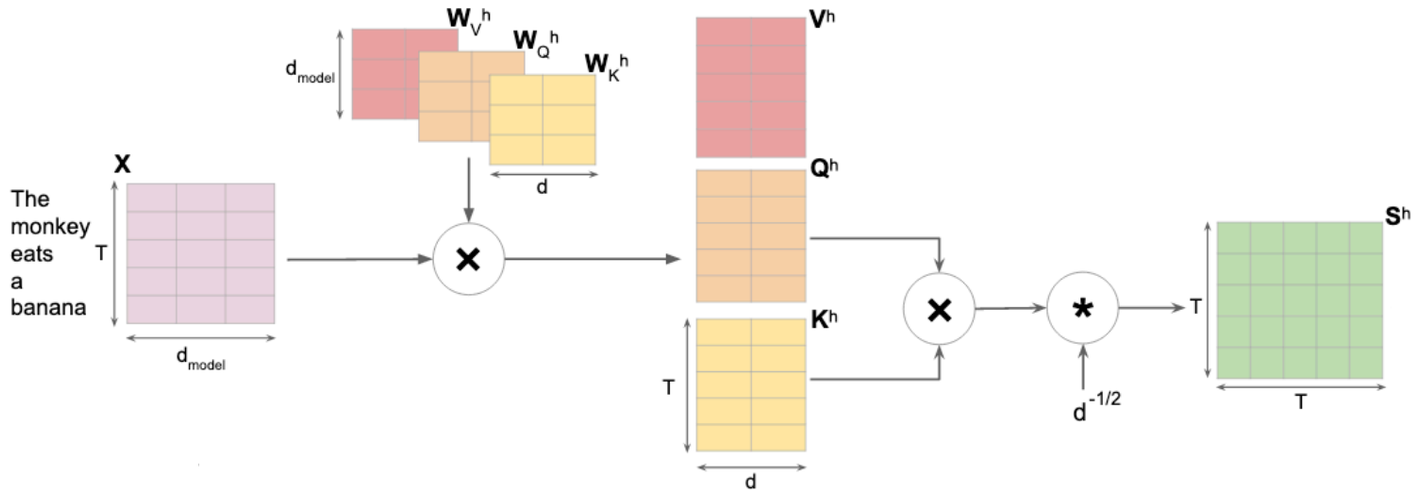
$$I_h = \mathbb{E}_{x \sim X} \left| \mathrm{Att}_h(x)^T \frac{\partial \mathcal{L}(x)}{\partial \mathrm{Att}_h(x)} \right|$$

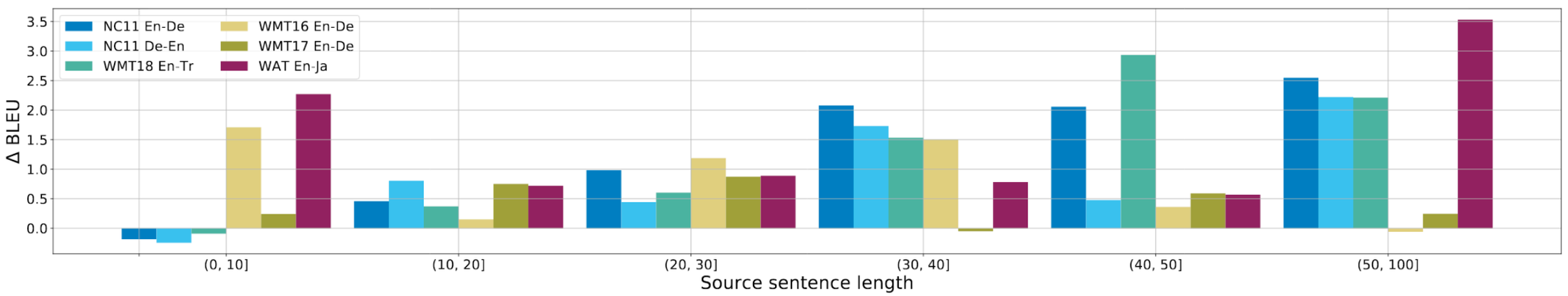
Prune heads based the expected importance of an attention head

Figures: Clark et al. 2019. "What Does BERT Look At? An Analysis of BERT's Attention"
Michel et al. 2019. Are Sixteen Heads Really Better than One?

Guiding Self-Attention

- We can constrain self-attention with dependency parses





Figures: Bugliarello and Okazaki 2020.

RoBERTA

- Deep analysis of the different aspects of BERT pre-training:
  - batch sizes
  - size of BPE vocabulary
  - utility of next sentence prediction task
  - number of pre-training steps

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Liu et al. 2019

Some Additional Resources

- The Annotated Transformer. Harvard NLP Group.
- FastText pre-trained embeddings. Facebook
- A Primer in BERTology: What We Know About How BERT Works. Rogers et al. 2020