CONTEXT-SPECIFIC WORD EMBEDDINGS

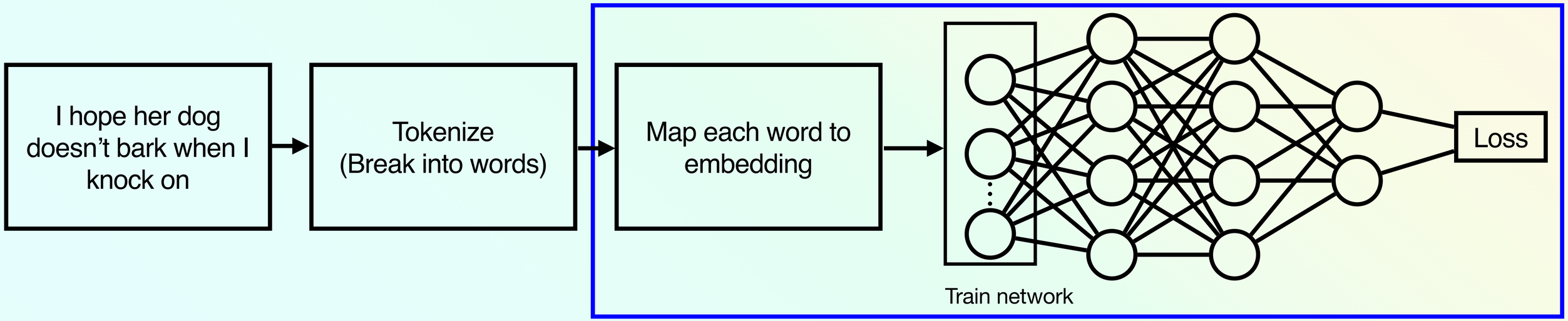
TODAY’S CLASS

GOALS

1. Understand the limitations of static embeddings
2. Develop the motivation for context-specific embeddings
3. Go over the self-attention mechanism to generate context-specific embeddings
    - This is the primary operation in LLMs that made them successful

SIMPLE LANGUAGE MODEL

WITH LEARNED WORD EMBEDDING

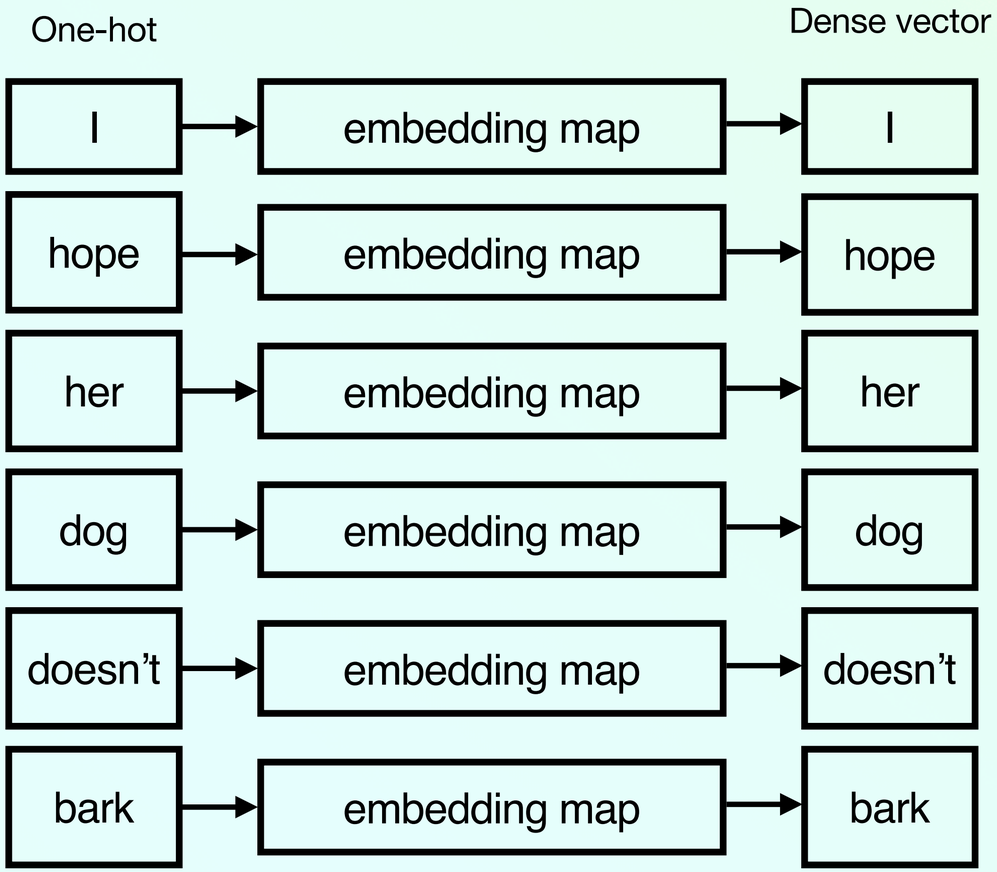


Can learn word embeddings that are good for the specific task

LLMs start with random embeddings and learn from scratch

WORD EMBEDDING

Spatial Invariance



Embedding operation is applied to each word

Same mapping regardless of position

This is a special case of a convolution operation

Spatial invariance to the position of the word

(Sometimes, the position of the word matters!)

WORD EMBEDDINGS

AS WE HAVE DEFINED THEM SO FAR

Provide a useful space to represent words such that similar words are close

Embedding captures features of the word for **ALL** its uses

WORD EMBEDDINGS

AS WE HAVE DEFINED THEM SO FAR

Provide a useful space to represent words such that similar words are close

Embedding captures features of the word for **ALL** its uses

- Even the bits that are irrelevant to the current use

Ideally, a good representation provides only the necessary information about the data

- Irrelevant data makes learning harder

WORD EMBEDDING

EXAMPLE OF STATIC REPRESENTATION LIMITATION

“I hope her dog doesn’t **bark** when I knock on the door.”

“The cedar’s tree **bark** has been used to make clothing, baskets, and many more items”

“Chocolate and mint are perfect partners in this delicious **bark** that makes a lovely gift”

Bark —> a sharp noise (usually by a dog)

Bark —> outer layer of a tree

Bark —> a treat that is a blend of chocolate and other candies

WordEmbedding(bark) = vector including all this information

WORD EMBEDDING

EXAMPLE OF STATIC REPRESENTATION LIMITATION

“I hope her dog doesn’t **bark** when I knock on the door.”

WORD EMBEDDING

EXAMPLE OF STATIC REPRESENTATION LIMITATION

“I hope her dog doesn’t **[outer layer of a tree, sharp noise, treat with chocolate]** when I knock on the door.”

WORD EMBEDDING

EXAMPLE OF STATIC REPRESENTATION LIMITATION

“I hope her dog doesn’t **[~~outer layer of a tree~~, sharp noise, ~~treat with chocolate~~]** when I knock on the door.”

The network has to learn to filter out part of the embeddings that are irrelevant to the current context

— it might take more hidden units, more layers, and/or more training time to figure this out.

CONTEXT-SPECIFIC WORD EMBEDDING

MOTIVATION

“I hope her dog doesn’t bark when I knock on the door.”

Change each embedding so that it captures the properties of the context

[outer layer of a tree, sharp noise, treat with chocolate] —> [0, sharp noise, 0]

CONTEXT-SPECIFIC WORD EMBEDDING

MOTIVATION

“I hope her **dog** doesn’t **bark** when I knock on the door.”

“The **dog’s bark** is not that loud”

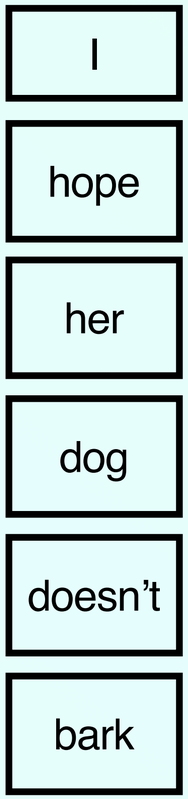
“When I took the **dog** to the park, it played with other **dogs**. It didn’t **bark** once.

Heuristic: if the word dog appears before bark, then change bark to only represent a noise

How do we do this in math?

CONTEXT-SPECIFIC WORD EMBEDDING

MOTIVATION



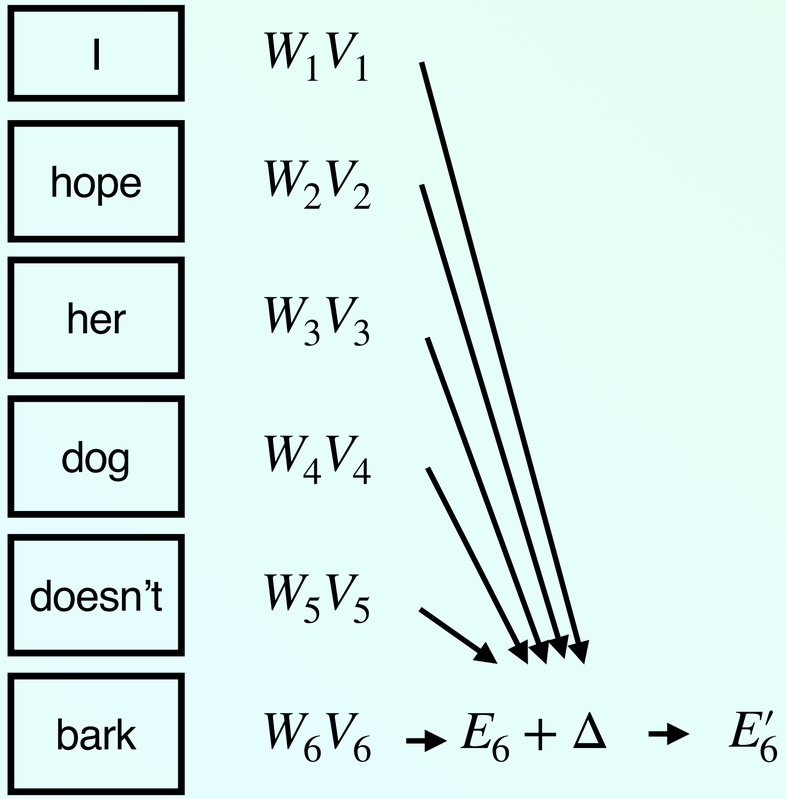
$E_i =$ embedding vector for the i^{th} word

$E_6' = E_6 + \Delta$ represent bark noise

Does dog appear before the word bark?

CONTEXT-SPECIFIC WORD EMBEDDING

MOTIVATION



$E_i =$ embedding vector for the i^{th} word

$E'_6 = E_6 + \Delta$ represent bark noise

Does dog appear before the word bark?

$$\Delta = \sum_{i=1}^{6} W_i V_i$$

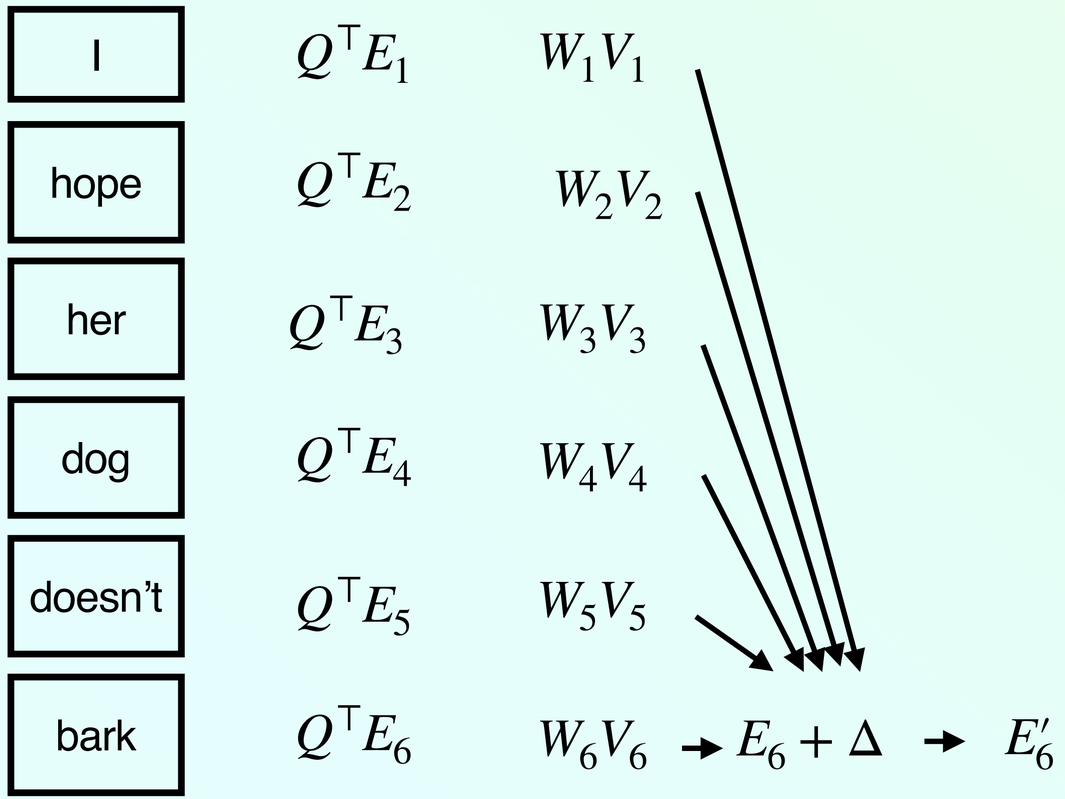
$V_i -$ value, a vector representing the change to E

$V_i = 0$ if the i^{th} word is not dog, else vector to remove not noise

$W_i -$ weight indicating to include V_i or not

CONTEXT-SPECIFIC WORD EMBEDDING

MOTIVATION



How do we generate weights W_i

Want it to be near 1 if word is dog, 0 otherwise

Q — Query, a vector representing the word dog

$W_i \propto Q^\top E_i$ — positive dot product means similar

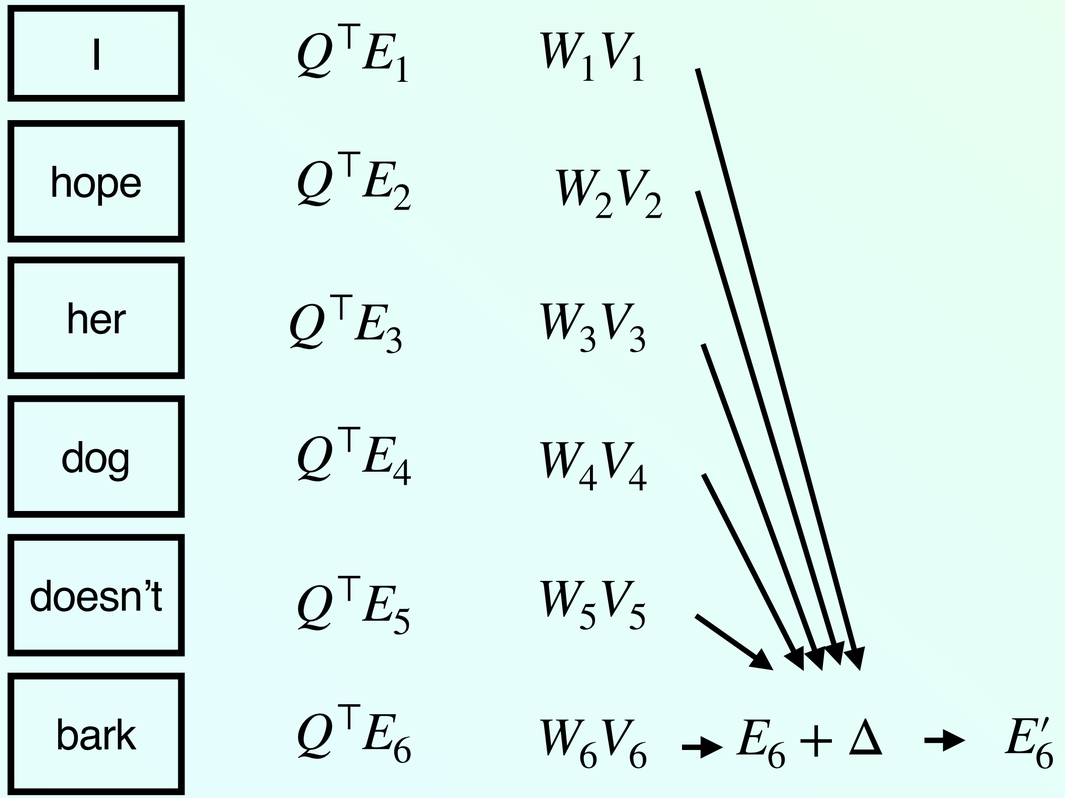
pointing same direction and the higher the value the more they point to the same direction

$$W_i = \frac{e^{Q^\top E_i}}{\sum_{j=1}^{6} e^{Q^\top E_j}}$$

softmax

CONTEXT-SPECIFIC WORD EMBEDDING

MOTIVATION



Where do Q and V come from?

TRANSFORMER SELF-ATTENTION

OVERVIEW

To get E_i' we need to generate a query Q_i specific to E_i.

Value vector V_j depends on E_j

smaller dimensional space

Instead of comparing Q_i to E_j we will compare to a smaller key vector K_j for each E_j

Queries and keys only need to look for small matches, not the whole space of E

TRANSFORMER SELF-ATTENTION

OVERVIEW

For a sequence of n tokens, for each $E_i \in \mathbb{R}^{1\times n_E}$, we generate a Q_i, K_i, and V_i,

$$Q_i = E_i W_Q, \quad W_Q \in \mathbb{R}^{n_E \times n_Q}, \quad Q_i \in \mathbb{R}^{1\times n_Q}$$

$$K_i = E_i W_K, \quad W_K \in \mathbb{R}^{n_E \times n_Q}, \quad K_i \in \mathbb{R}^{1\times n_Q}$$

$$V_i = E_i W_V, \quad W_V \in \mathbb{R}^{n_E \times n_E}, \quad V_i \in \mathbb{R}^{1\times n_E}$$

TRANSFORMER SELF-ATTENTION

OVERVIEW

For a sequence of n tokens, for each $E_i \in \mathbb{R}^{1\times n_E}$, we generate a Q_i, K_i, and V_i,

$$Q_i = E_i W_Q, \quad W_Q \in \mathbb{R}^{n_E \times n_Q}, \quad Q_i \in \mathbb{R}^{1\times n_Q}$$

$$K_i = E_i W_K, \quad W_K \in \mathbb{R}^{n_E \times n_Q}, \quad K_i \in \mathbb{R}^{1\times n_Q}$$

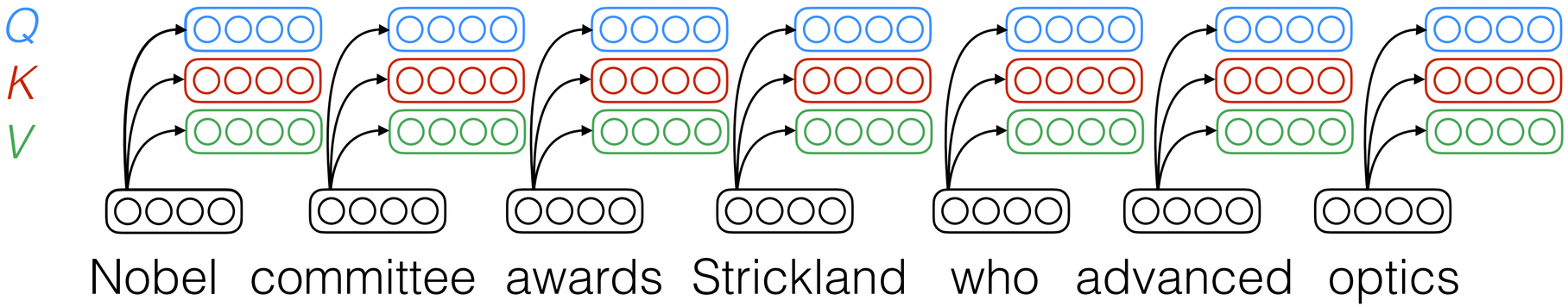
$$V_i = E_i W_V, \quad W_V \in \mathbb{R}^{n_E \times n_E}, \quad V_i \in \mathbb{R}^{1\times n_E}$$

$$A_{i,j} = \frac{e^{Q_i K_j^\top}}{\sum_{k=1}^{n} e^{Q_i K_j^\top}}$$

$$E'_i = E_i + \sum_{j=1}^{n} A_{i,j} V_j$$

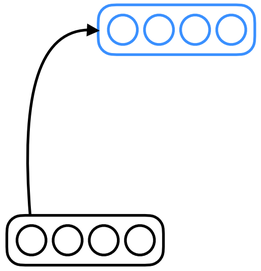
Attention weights — Does E_j match the query Q_i

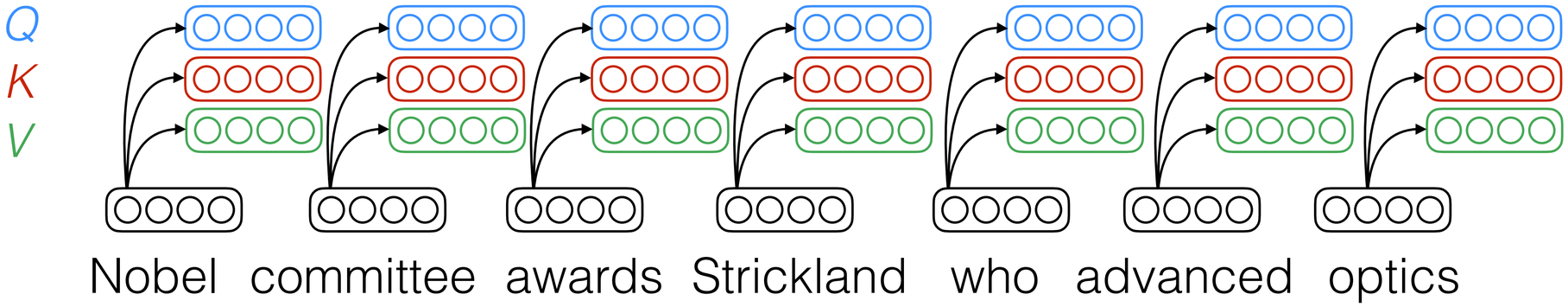
Transformer self-attention



Slide from Emma Strubell

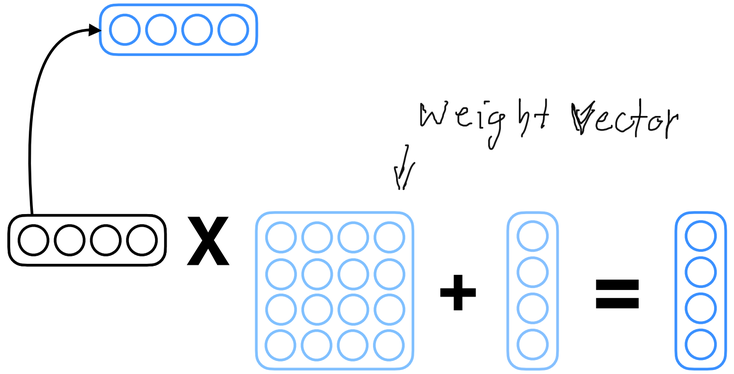
Transformer self-attention

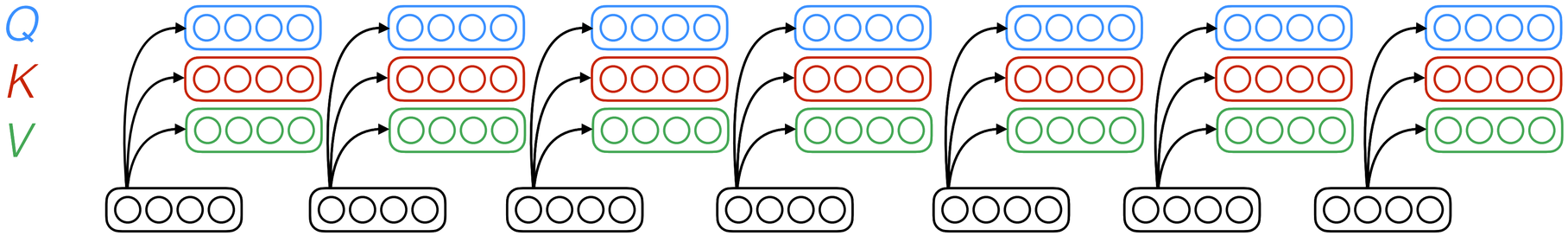




Slide from Emma Strubell

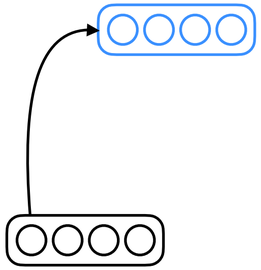
Transformer self-attention

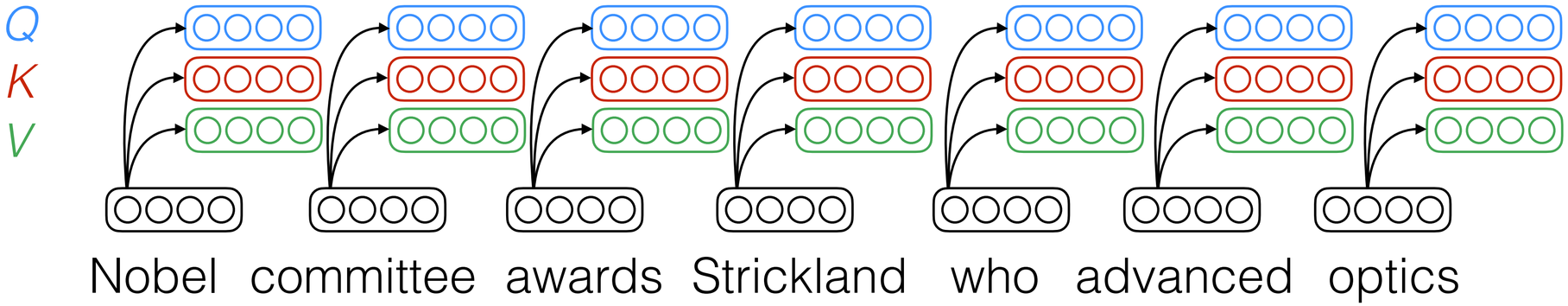




Slide from Emma Strubell

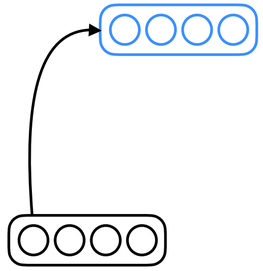
Transformer self-attention

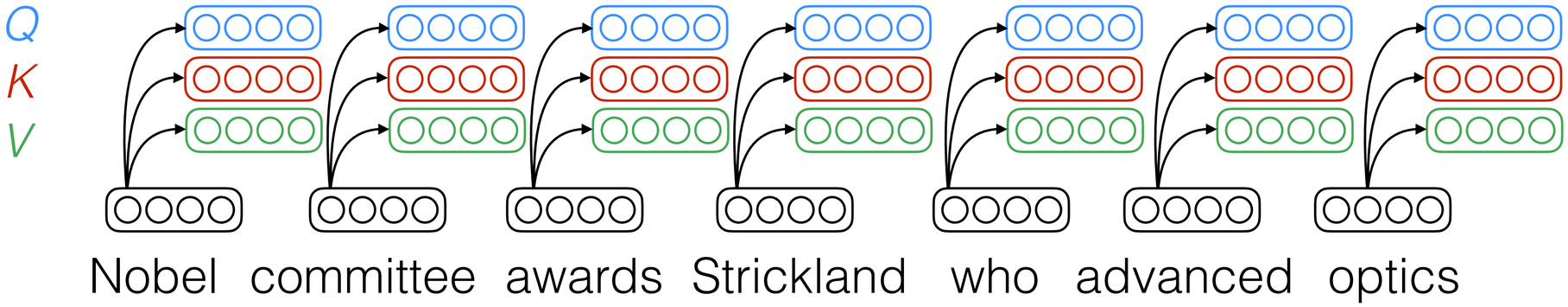




Slide from Emma Strubell

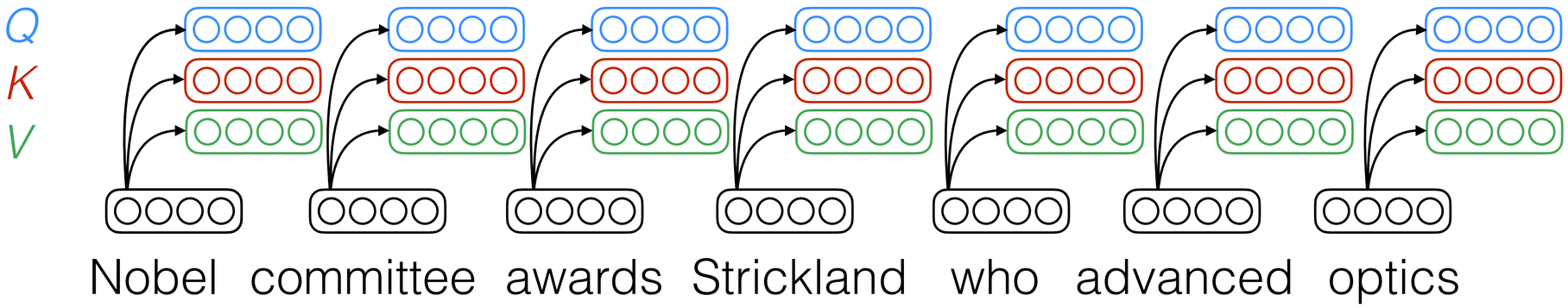
Transformer self-attention





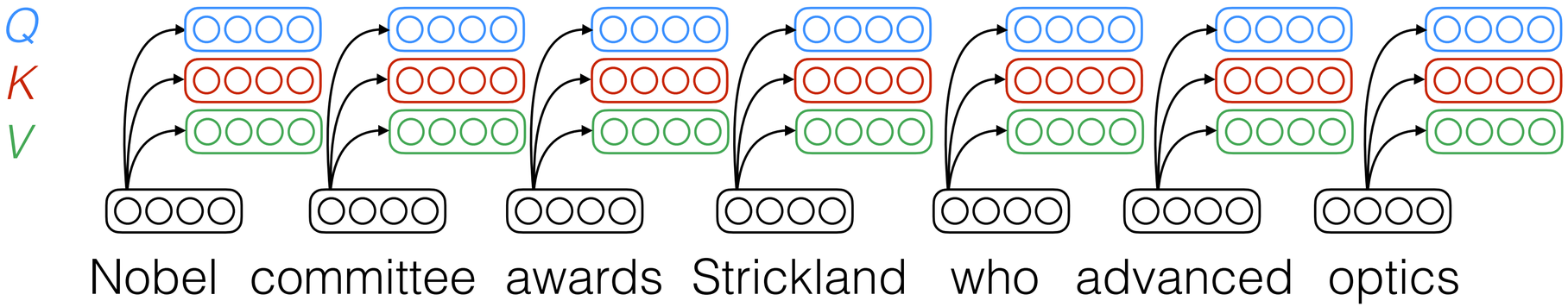
Slide from Emma Strubell

Transformer self-attention



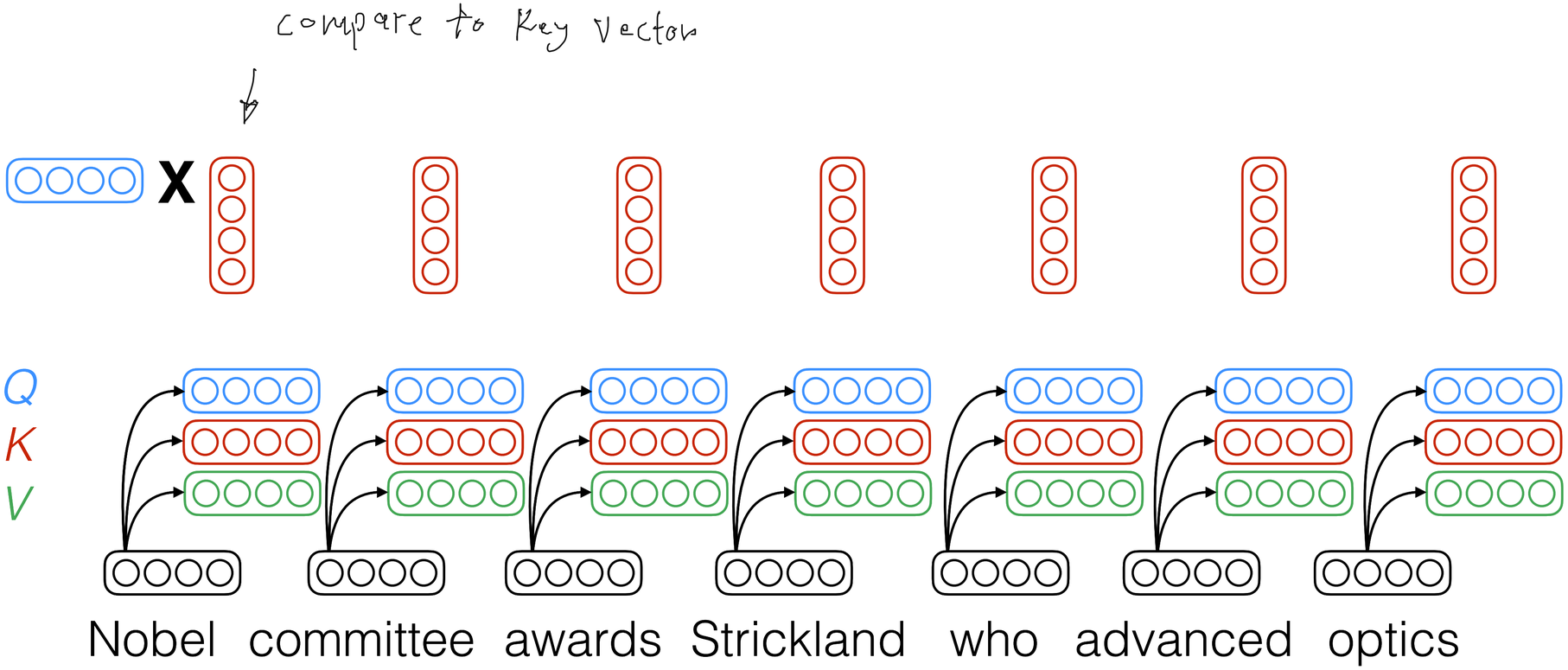
Slide from Emma Strubell

Transformer self-attention



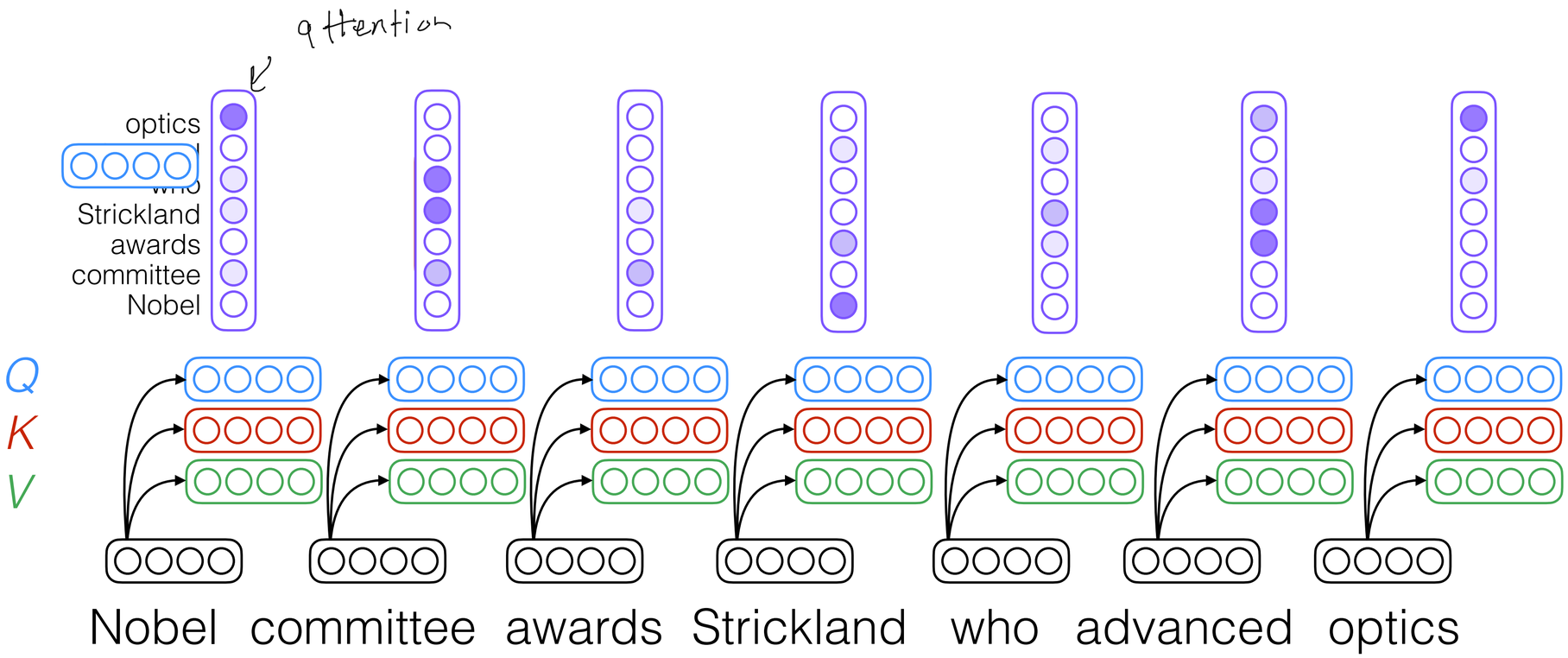
Slide from Emma Strubell

Transformer self-attention



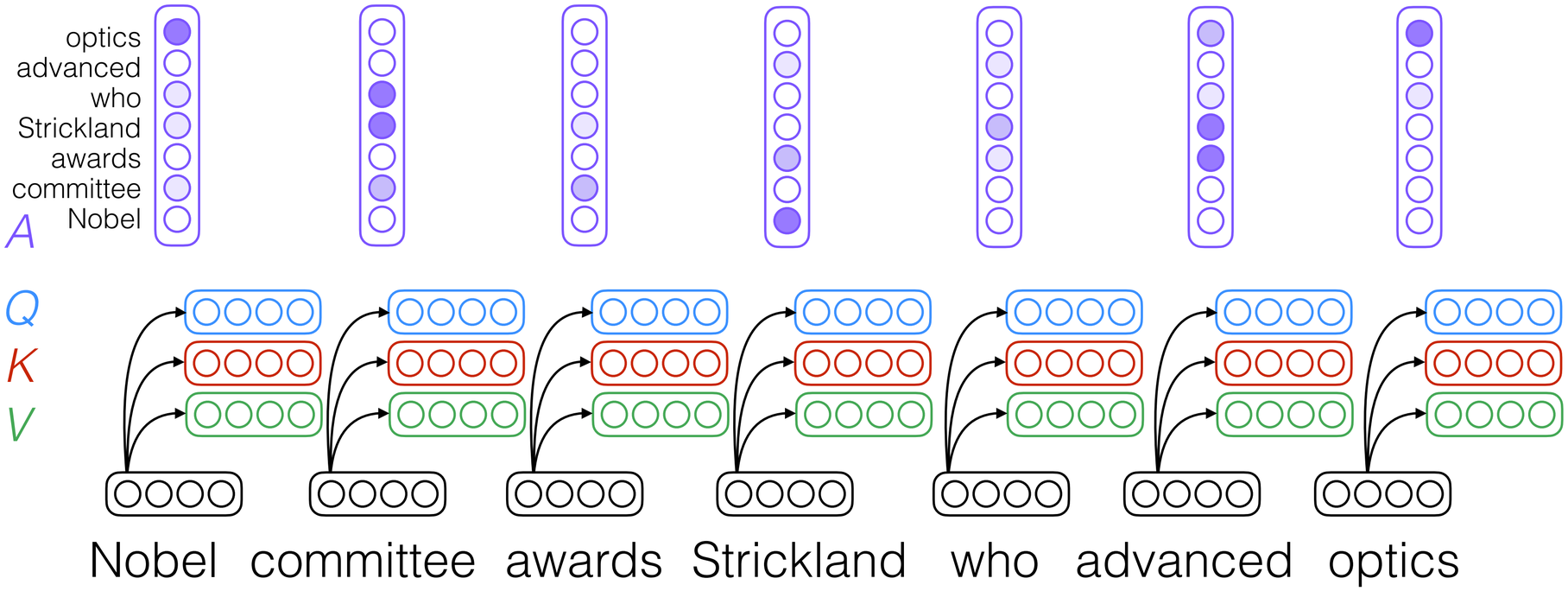
Slide from Emma Strubell

Transformer self-attention



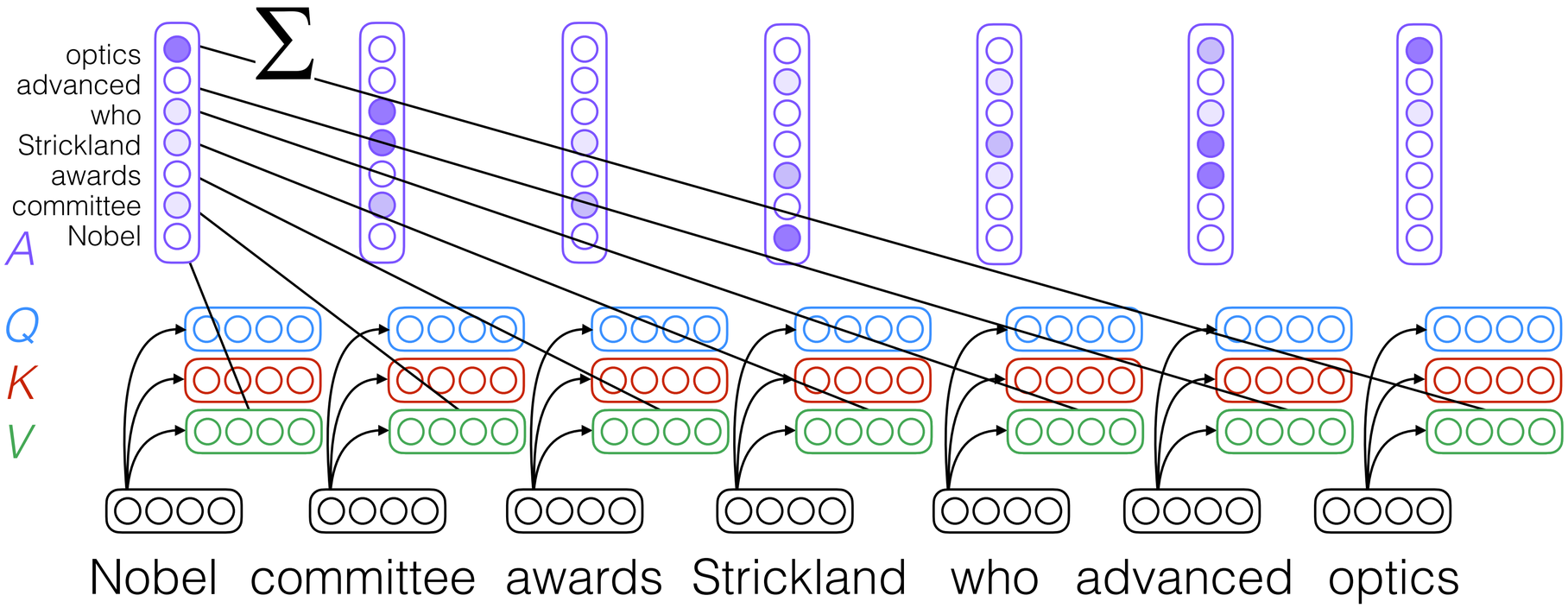
Slide from Emma Strubell

Transformer self-attention



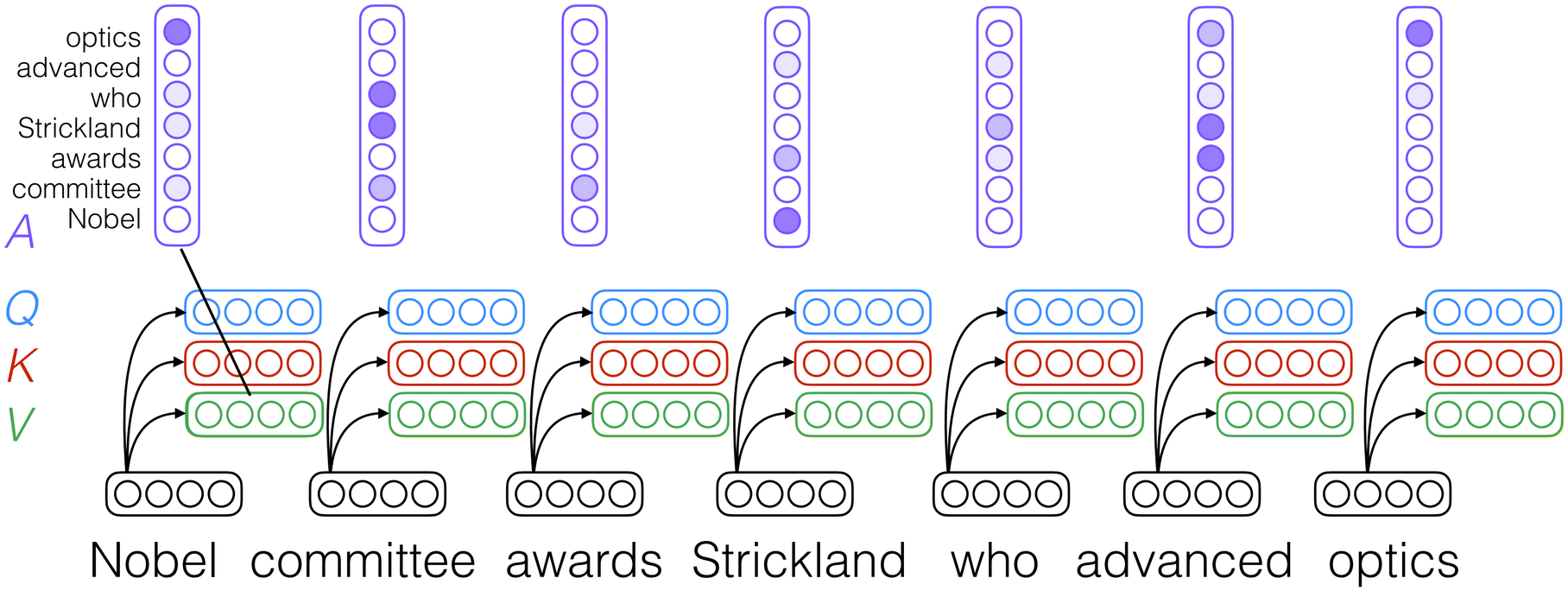
Slide from Emma Strubell

Transformer self-attention



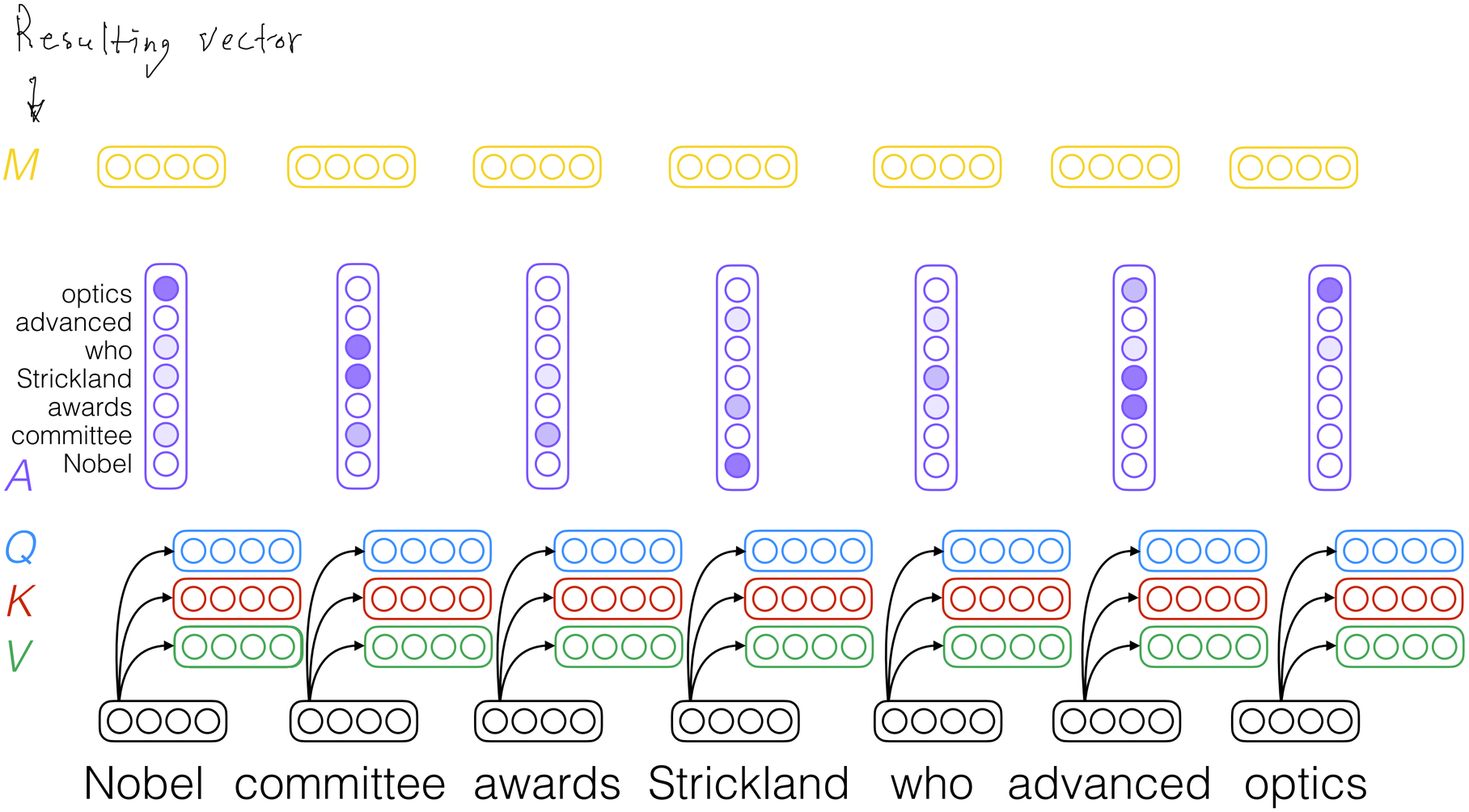
Slide from Emma Strubell

Transformer self-attention



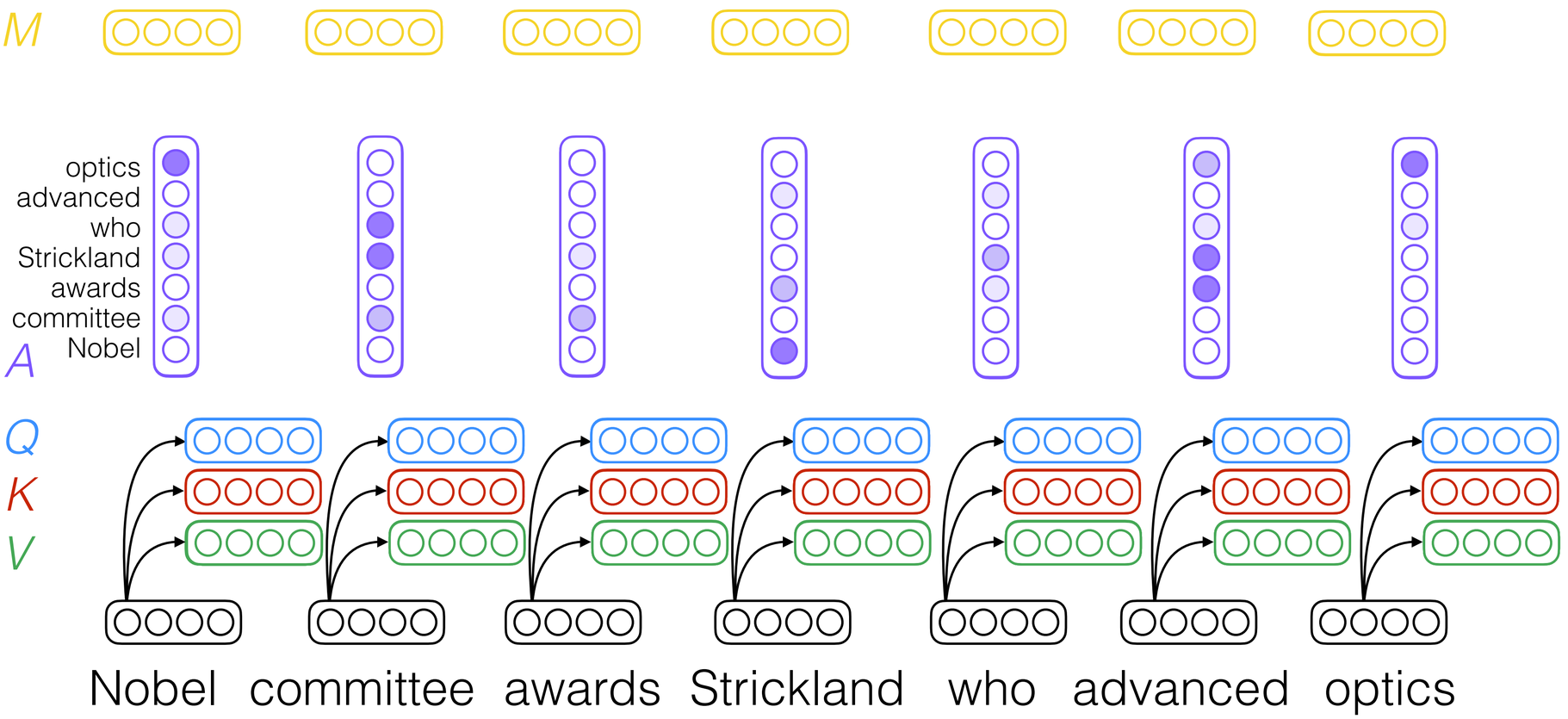
Slide from Emma Strubell

Transformer self-attention



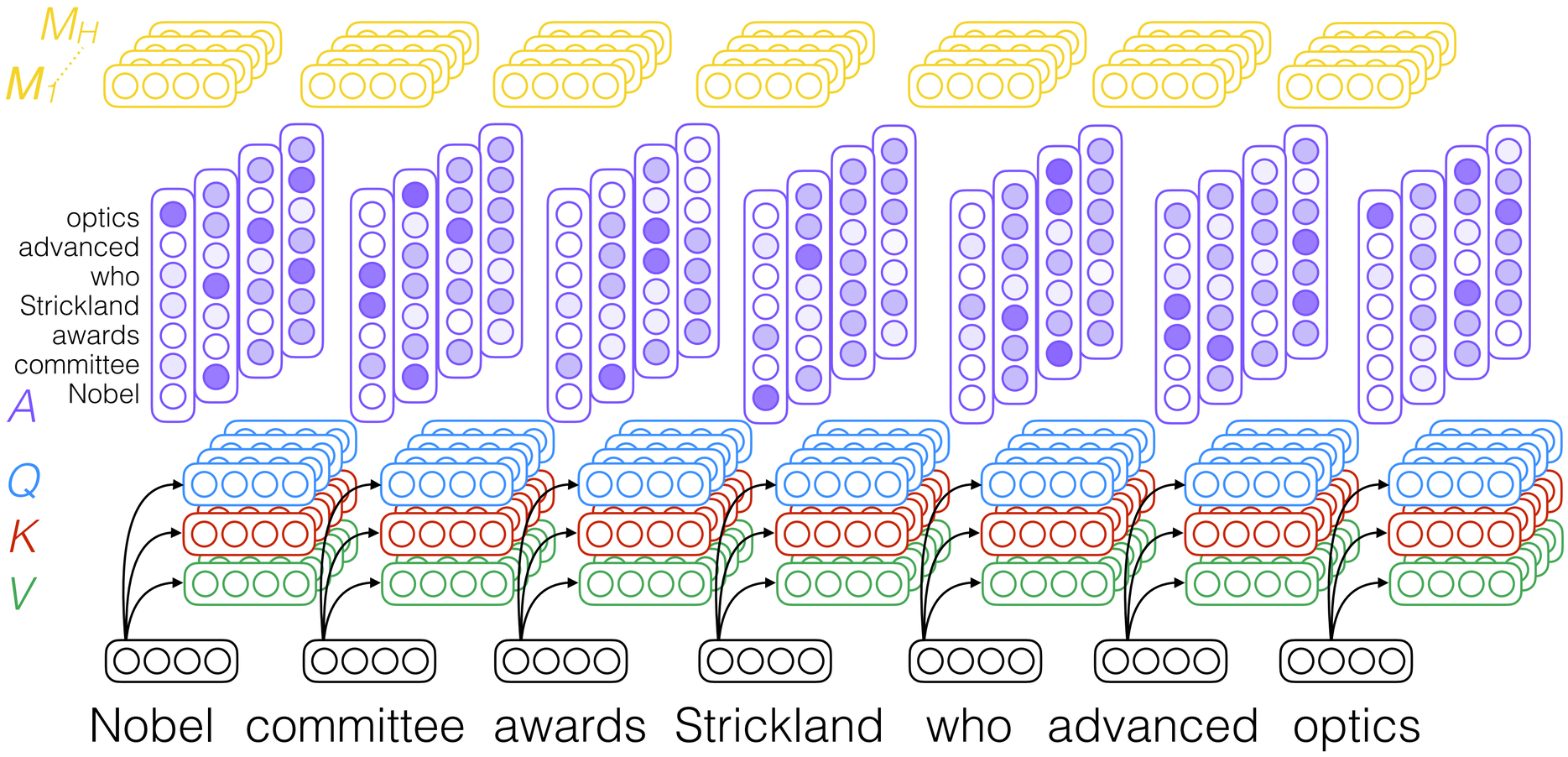
Slide from Emma Strubell

Transformer self-attention



Slide from Emma Strubell

Multi-head self-attention



Slide from Emma Strubell

TRANSFORMER SELF-ATTENTION

SPECIFICS

For a sequence of n tokens, for each E_i, we generate a Q_i, K_i, and V_i,

$$Q_i = E_i W_Q, \quad W_Q \in \mathbb{R}^{n_E \times n_Q}, \quad Q_i \in \mathbb{R}^{1 \times n_Q}$$

$$K_i = E_i W_K, \quad W_K \in \mathbb{R}^{n_E \times n_Q}, \quad K_i \in \mathbb{R}^{1 \times n_Q}$$

$$V_i = E_i W_V, \quad W_V \in \mathbb{R}^{n_E \times n_E}, \quad V_i \in \mathbb{R}^{1 \times n_E}$$

$$A_{i,j} = \frac{e^{Q_i K_j^\top}}{\sum_{k=1}^{n} e^{Q_i K_j^\top}}$$

$$E'_i = E_i + \sum_{j=1}^{n} A_{i,j} V_j$$

QUIZ

TRANSFORMER SELF-ATTENTION

SPECIFICS

The larger the query/key vectors (n_Q increases), the larger $|Q_i K_i^\top|$

Large $Q_i K_i^\top$ —> attention on only one element

TRANSFORMER SELF-ATTENTION

SPECIFICS

The larger the query/key vectors (n_Q increases), the larger $|Q_i K_i^\top|$

Large $Q_i K_i^\top$ —> attention on only one element

Rescale based on query dimension

To solve this problem

$$A_{i,j} = \frac{e^{Q_i K_j^\top / \sqrt{n_Q}}}{\sum_{k=1}^{n} e^{Q_i K_j^\top / \sqrt{n_Q}}}$$

optimization is much easier. The derivatives of softmax

$$E_i' = E_i + \sum_{j=1}^{n} A_{i,j} V_j$$

TRANSFORMER SELF-ATTENTION

SPECIFICS

Do not want to use later words to influence earlier words' representation.

Need to mask out the attention of early words to later words.

If $j > i$, $Q_i K_j^\top = -\infty$

means next word

← Numpy will turn into Zero

$$A_{i,j} = \frac{e^{Q_i K_j^\top / \sqrt{n_Q}}}{\sum_{k=1}^{n} e^{Q_i K_j^\top / \sqrt{n_Q}}} = \frac{e^{Q_i K_j^\top / \sqrt{n_Q}}}{\sum_{k=1}^{i} e^{Q_i K_j^\top / \sqrt{n_Q}} + \underbrace{(n-i)e^{-\infty}}_{=0}}$$

masked self-attention

NEXT CLASS

Transformers