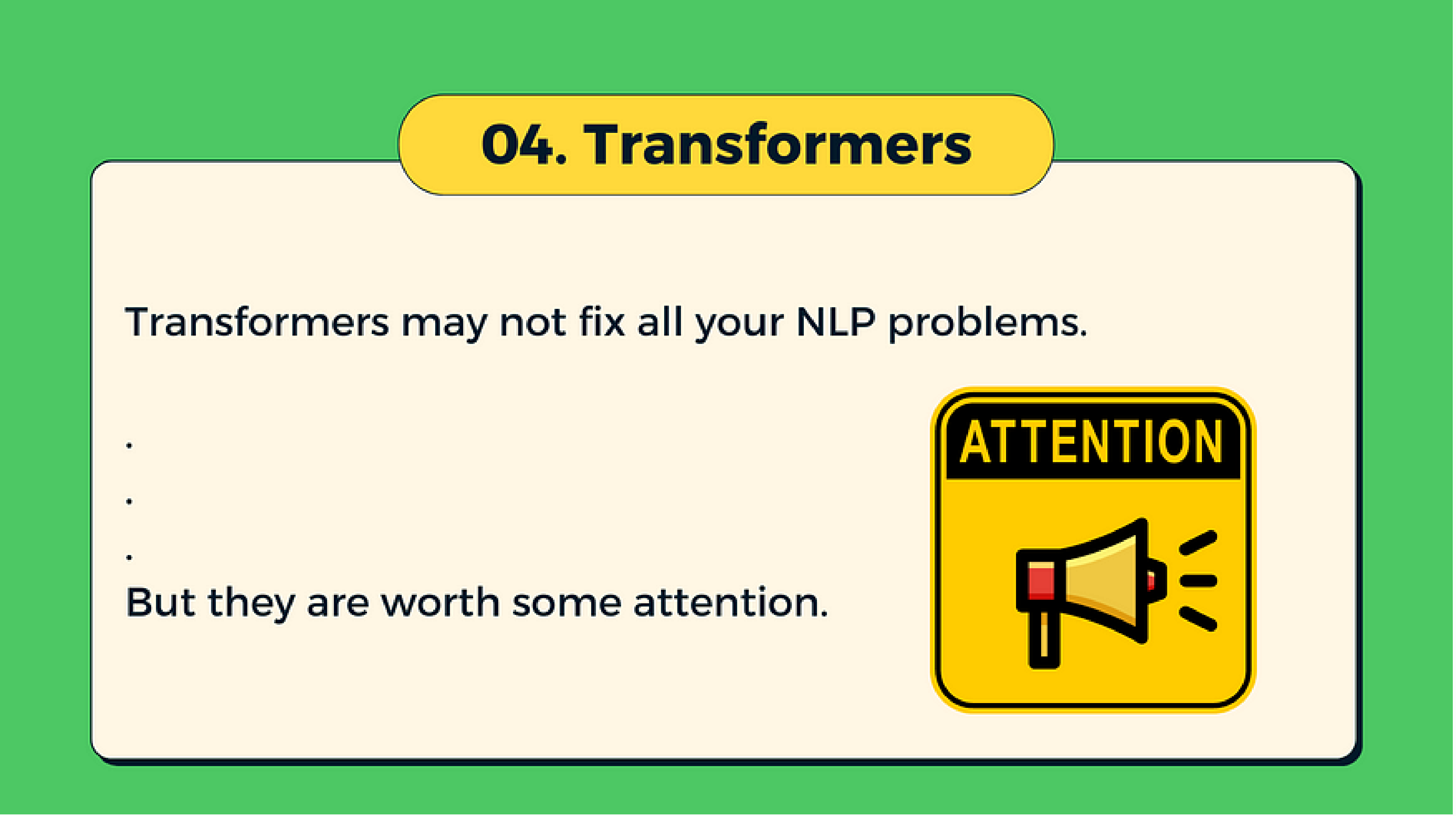


04. Transformers

Transformers may not fix all your NLP problems.

.

.

.

But they are worth some attention.

Source: Ramsri Goutham

CS 1671 / CS 2071 / ISSP 2071
Human Language Technologies

Session 17: Transformers part 1

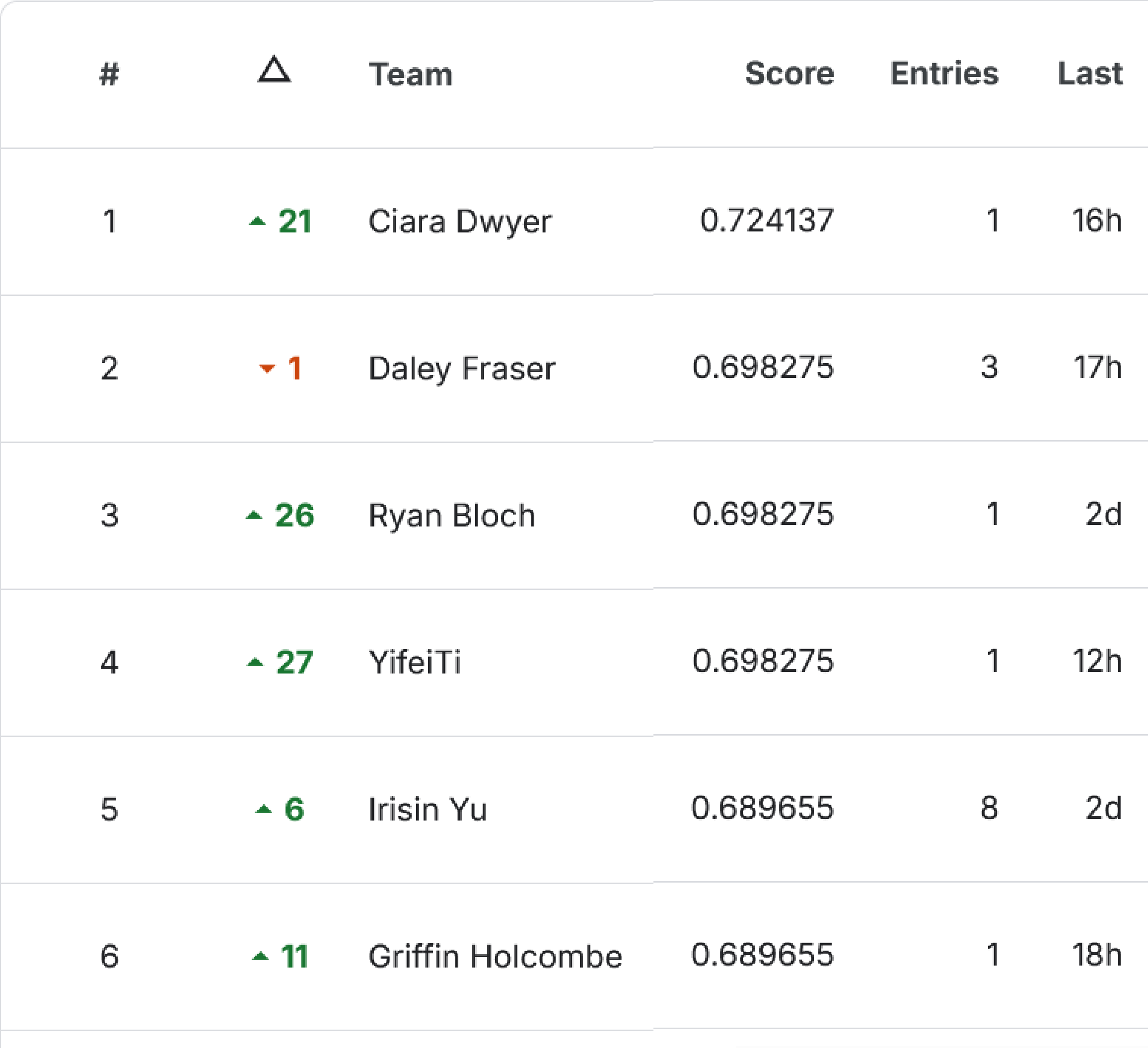
Michael Miller Yoder

March 17, 2026

University of Pittsburgh School of Computing and Information

Assessments: Homework 2

- Congrats to Ciara, Daley, Ryan, and Yifei!
- Grades for HW2 should be released next week



#	△	Team	Score	Entries	Last
1	▲ 21	Ciara Dwyer	0.724137	1	16h
2	▼ 1	Daley Fraser	0.698275	3	17h
3	▲ 26	Ryan Bloch	0.698275	1	2d
4	▲ 27	YifeiTi	0.698275	1	12h
5	▲ 6	Irisin Yu	0.689655	8	2d
6	▲ 11	Griffin Holcombe	0.689655	1	18h

Assessments: quiz

- Quiz during class **next Mon Mar 23** covering:
    - Session 11: J+M 4.7-4.10, 4.12
    - Session 12: J+M 5-5.2, 5.5-5.8, 5.10
    - Session 13: J+M 6-6.1, 6.3-6.4
    - Session 14: J+M 6.5-6.6
    - Session 16 (today): J+M 7-7.5, 7.7

Assessments: project

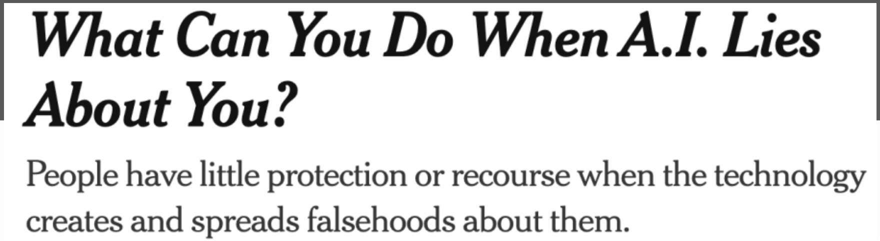
- Project progress report is due **next Thu Mar 26**
- Part 1: Basic data analysis (if any updates are required from the proposal)
- Part 2: Result from baseline approach
  - Ideally performance metric result from the baseline system you proposed
- Part 3: LLM proposal
  - How might you use an LLM programmatically to attempt your task?
  - Zero-shot and more advanced approaches

Overview: Transformers part 1

- Discussion: LLM harms
- Contextual word embeddings
- Self-attention
  - Activity: work through self-attention
- Multi-headed attention

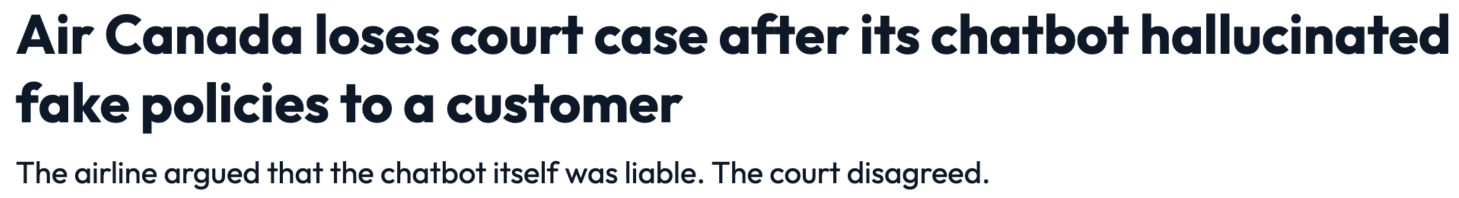
Harms from LLMs

Hallucination



What Can You Do When A.I. Lies About You?

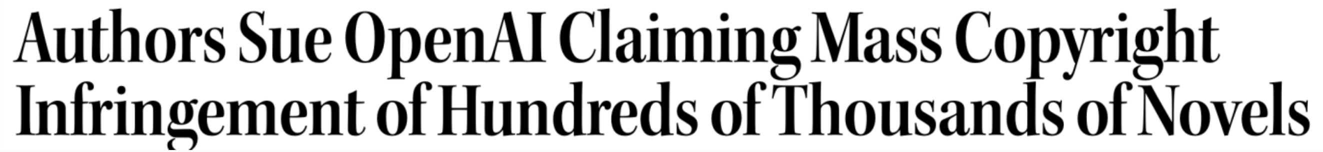
People have little protection or recourse when the technology creates and spreads falsehoods about them.



Air Canada loses court case after its chatbot hallucinated fake policies to a customer

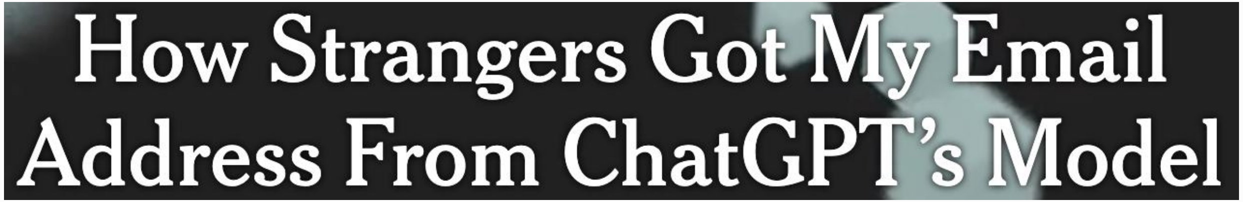
The airline argued that the chatbot itself was liable. The court disagreed.

Copyright



Authors Sue OpenAI Claiming Mass Copyright Infringement of Hundreds of Thousands of Novels

Privacy

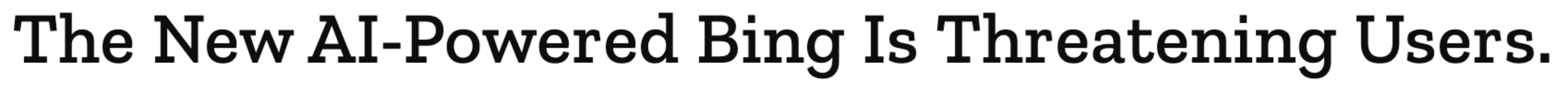


How Strangers Got My Email Address From ChatGPT's Model

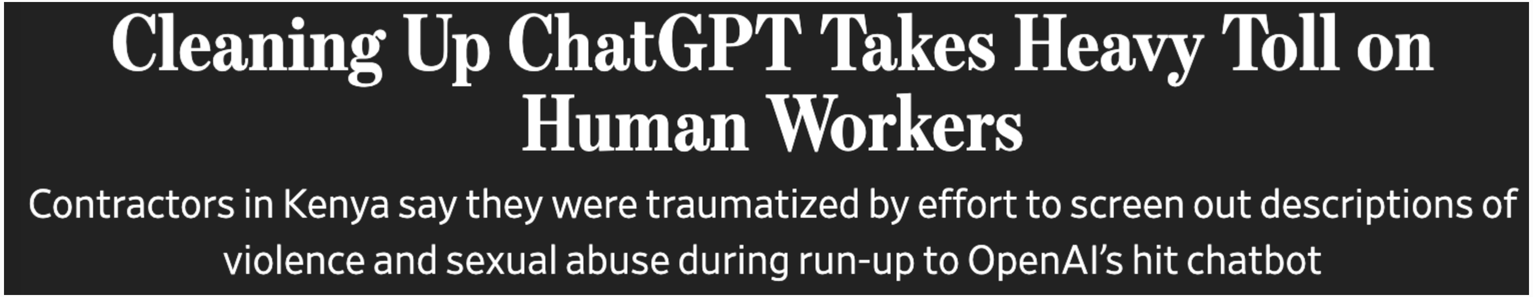
Slide adapted from Jurafsky and Martin

Harms from LLMs

Toxicity and abuse



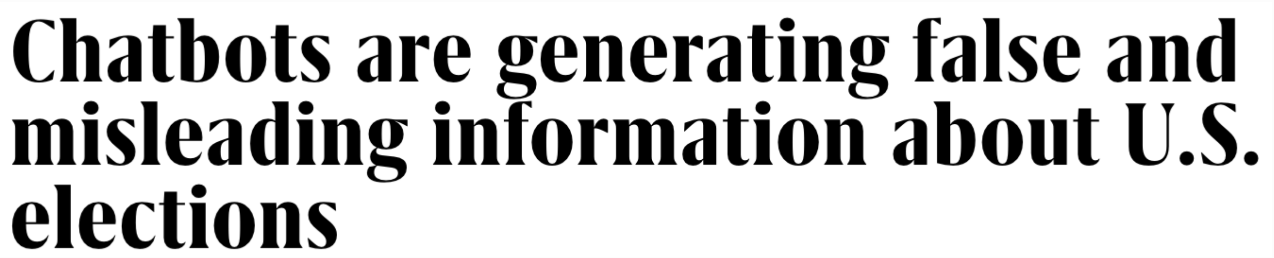
The New AI-Powered Bing Is Threatening Users.



Cleaning Up ChatGPT Takes Heavy Toll on Human Workers

Contractors in Kenya say they were traumatized by effort to screen out descriptions of violence and sexual abuse during run-up to OpenAI's hit chatbot

Misinformation



Chatbots are generating false and misleading information about U.S. elections

Slide adapted from Jurafsky and Martin

Discussion: harms from LLMs

- What potential harms are you concerned about from LLMs?
- What training data could be linked to those harms?

Contextual word embeddings

Problem with static embeddings (word2vec)

They are static! The embedding for a word doesn't reflect how its meaning changes in context.

`The chicken didn't cross the road because` **`it`** `was too tired`

What is the meaning represented in the static embedding for "it"?

Slide adapted from Jurafsky and Martin

Contextual Embeddings

- Intuition: a representation of meaning of a word should be different in different contexts!
- **Contextual embedding**: each word has a different vector that expresses different meanings depending on the surrounding words
- How to compute contextual embeddings? **Attention**

Slide adapted from Jurafsky and Martin

Contextual Embeddings

`The chicken didn't cross the road because it`

What should be the properties of "it"?

The chicken didn't cross the road because it was too **tired**

The chicken didn't cross the road because it was too **wide**

At this point in the sentence, it's probably referring to either the chicken or the street

Slide adapted from Jurafsky and Martin

Self-attention

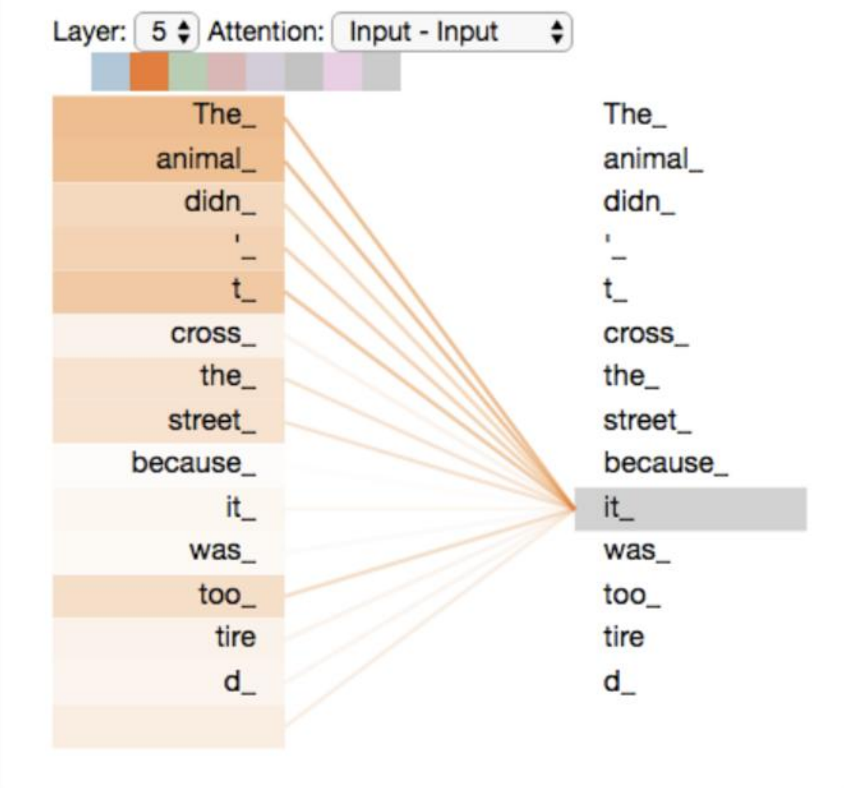
Vaswani et al. 2017,
“Attention is all you need”

Intuition of self-attention

- Transformers give one output vector (embedding) for every input token
- Build up the contextual embedding for each token by selectively integrating information from all the neighboring words
- We say that a word "attends to" some neighboring words more than others

Slide adapted from Jurafsky and Martin

Self-attention illustrated



Layer: 5 Attention: Input - Input

Source: The Illustrated Transformer

Attention definition

A mechanism for helping compute the embedding for a token by selectively attending to and integrating information from surrounding tokens (at the previous layer).

More formally: a method for doing a weighted sum of vectors.

Slide adapted from Jurafsky and Martin

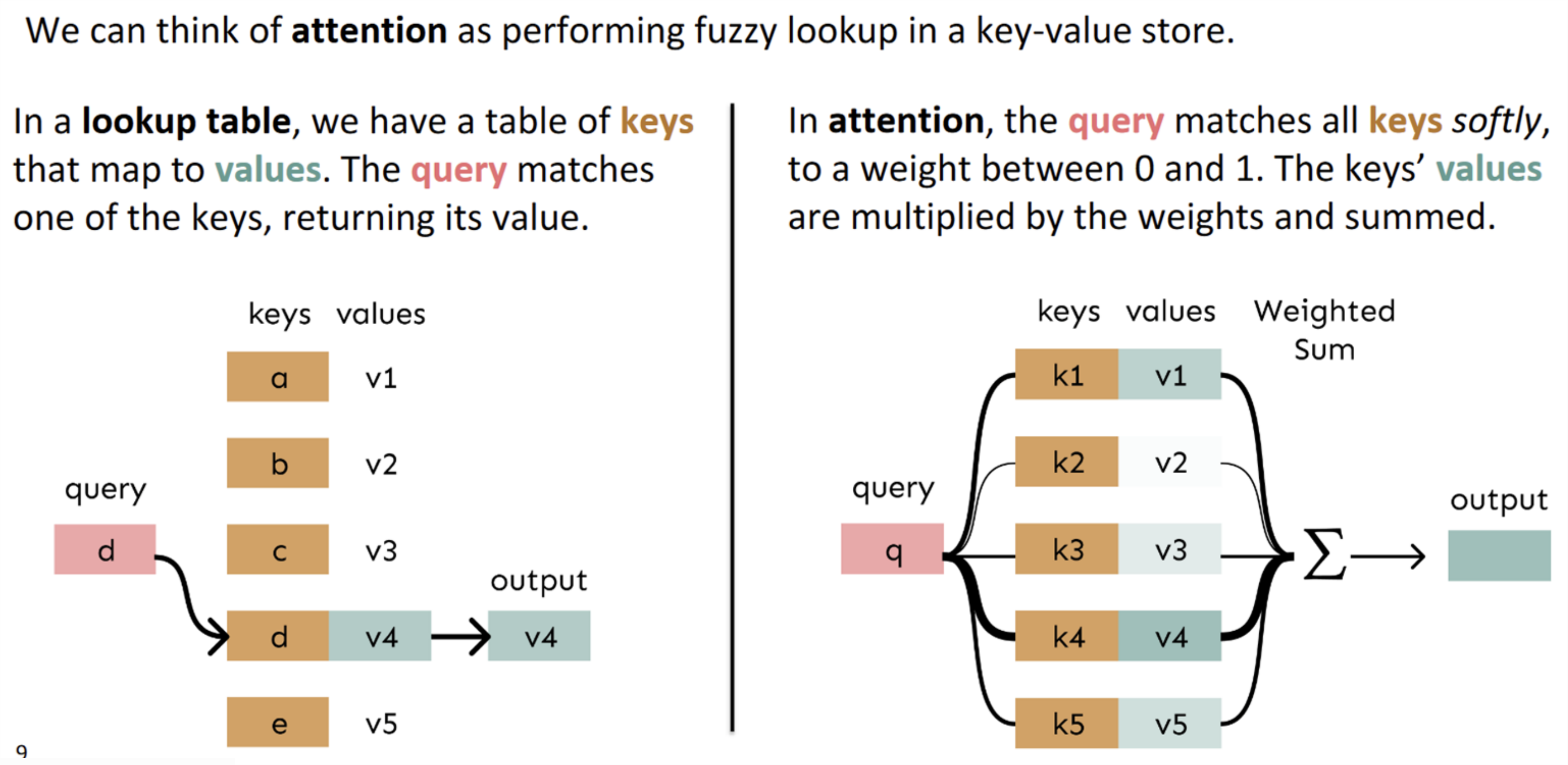
An actual attention head: slightly more complicated

High-level idea: we'll represent 3 separate roles the vector for each word, x_i plays:

- **query:** As *the current element* being compared to the other inputs.
- **key:** as *an input* that is being compared to the current element to determine a similarity
- **value:** a value of a preceding element that gets weighted and summed

Slide adapted from Jurafsky and Martinc

Attention as a soft, averaging lookup table



We can think of **attention** as performing fuzzy lookup in a key-value store.

In a **lookup table**, we have a table of **keys** that map to **values**. The **query** matches one of the keys, returning its value.

In **attention**, the **query** matches all **keys** *softly*, to a weight between 0 and 1. The keys' **values** are multiplied by the weights and summed.

Slide credit: John Hewitt

Parameters: weight matrices for queries, keys and values

- We'll use matrices to project each vector x_i into a representation of its role as query, key, value:

- query: W^Q
- key: W^K
- value: W^V

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{V}}$$

Slide adapted from Jurafsky and Martin

An actual attention head: slightly more complicated

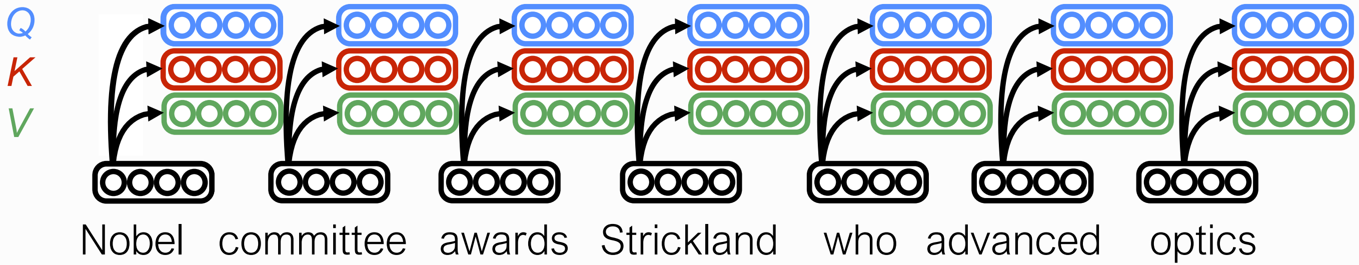
- Given these 3 representation of x_i

$$\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{V}}$$

- To compute the similarity of current element x_i with some element (for self-attention) x_j
- We’ll use dot product between q_i and k_j.
- And instead of summing up x_j, we'll sum up v_j

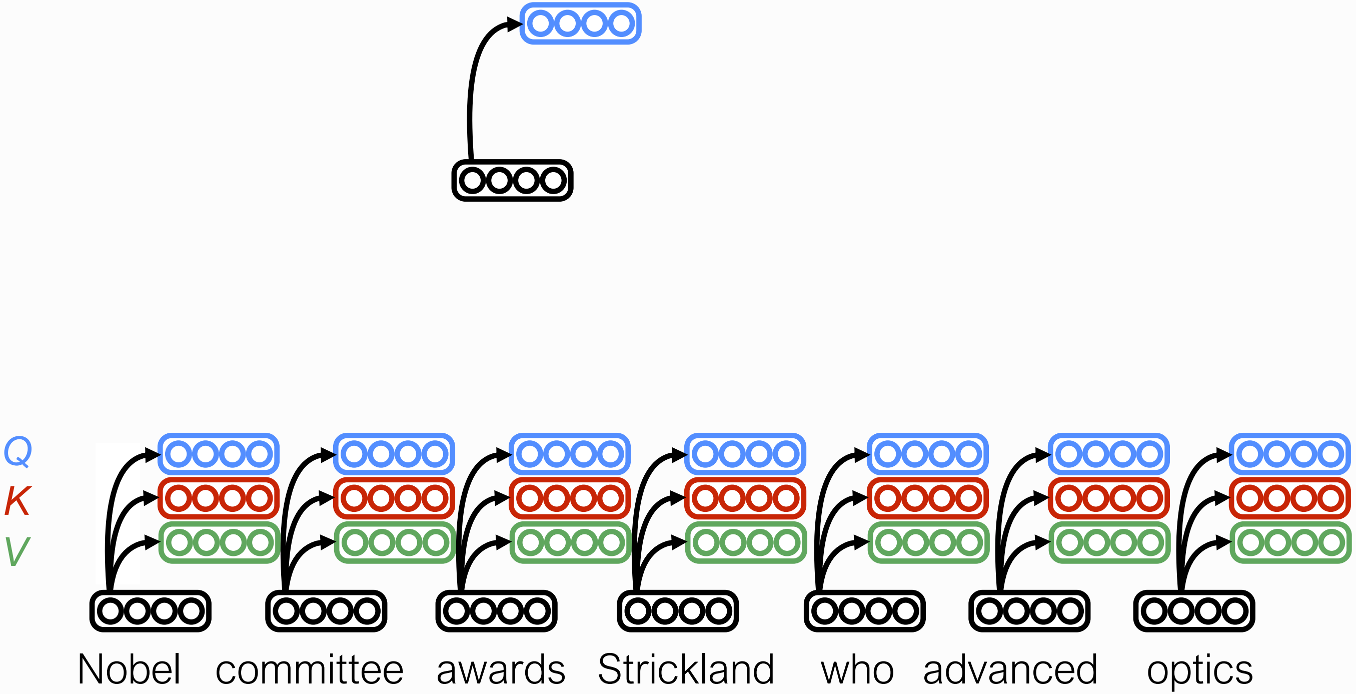
Slide adapted from Jurafsky and Martinc

Transformer self-attention



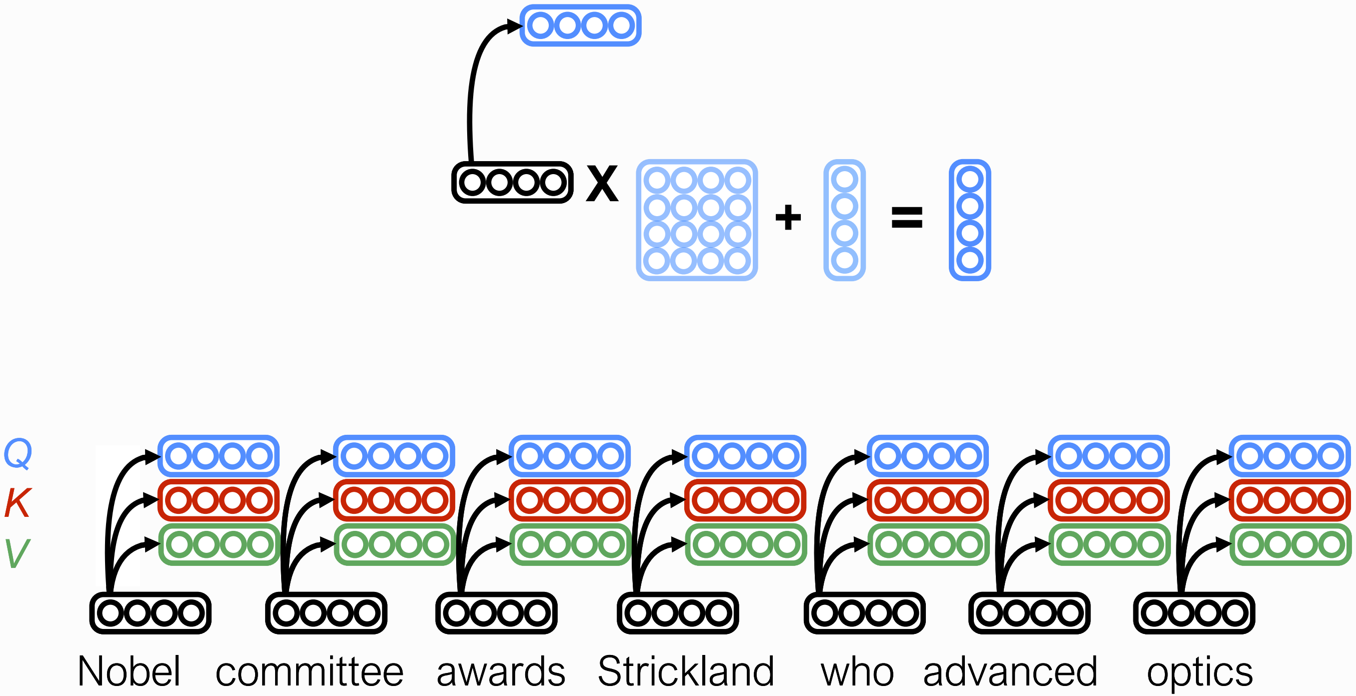
Source: Emma Strubell

Transformer self-attention



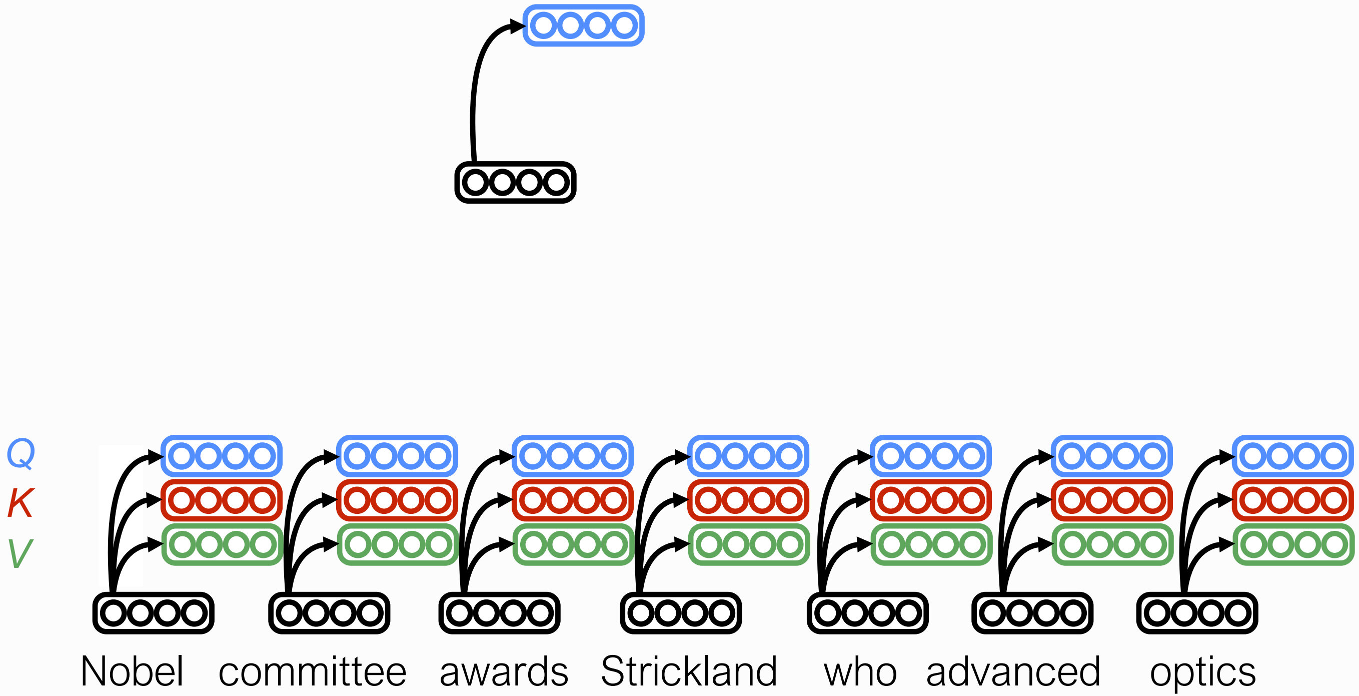
Source: Emma Strubell

Transformer self-attention



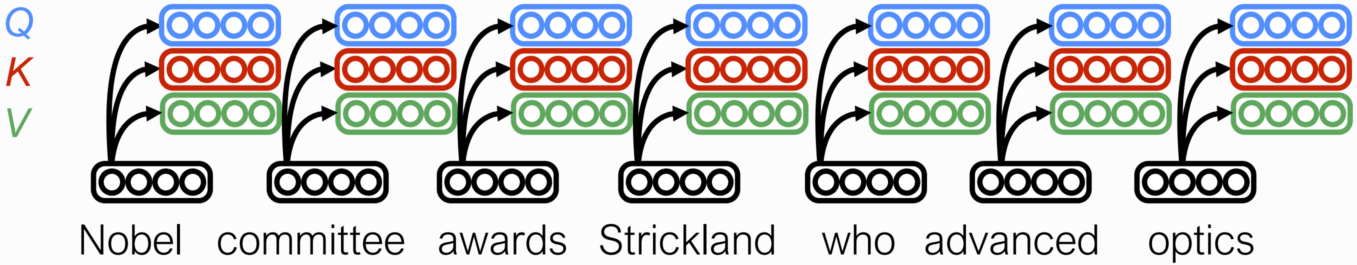
Source: Emma Strubell

Transformer self-attention



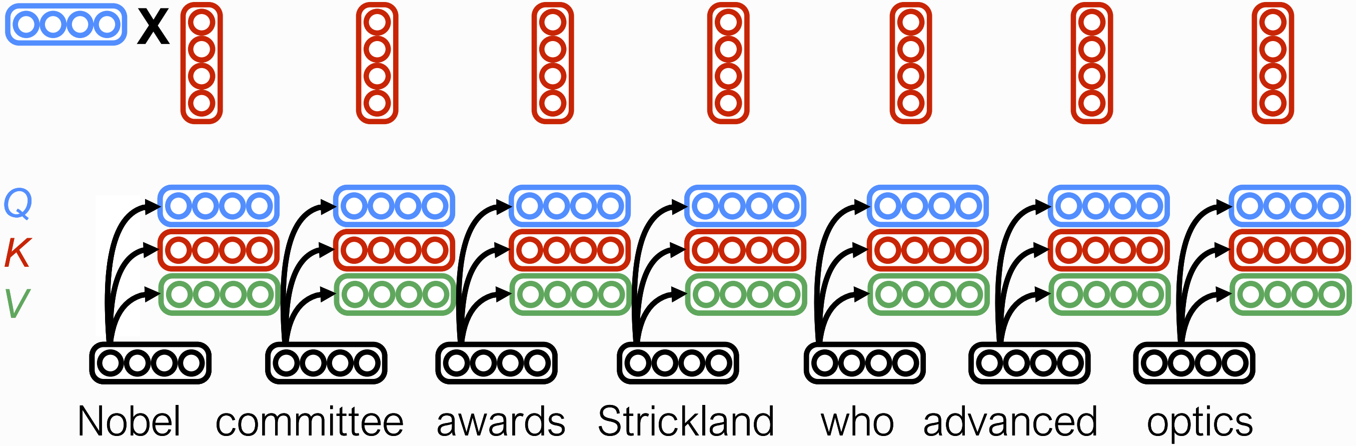
Source: Emma Strubell

Transformer self-attention



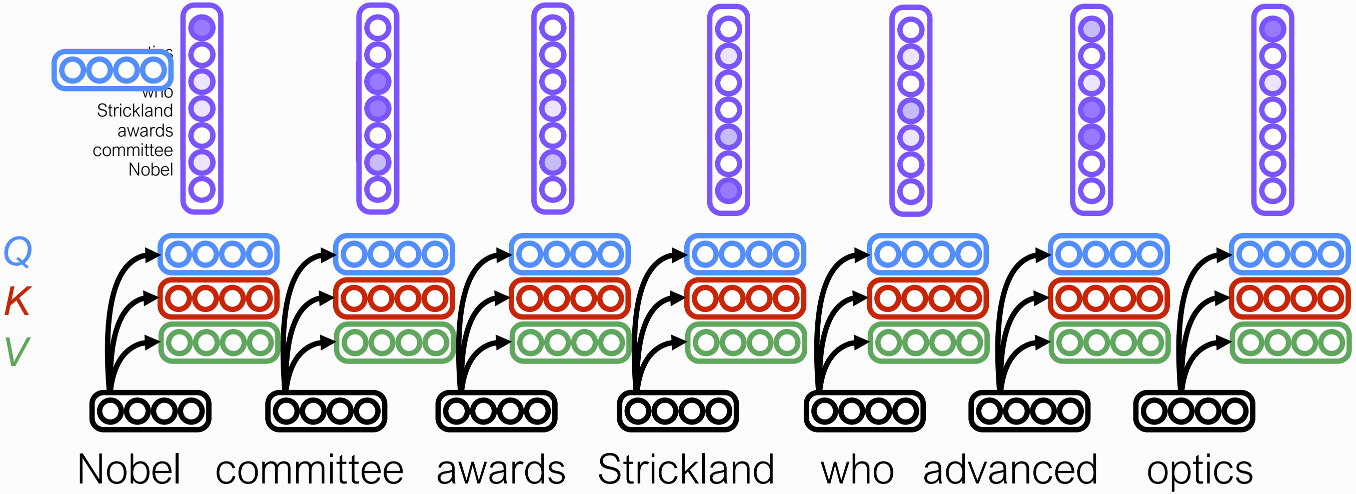
Source: Emma Strubell

Transformer self-attention



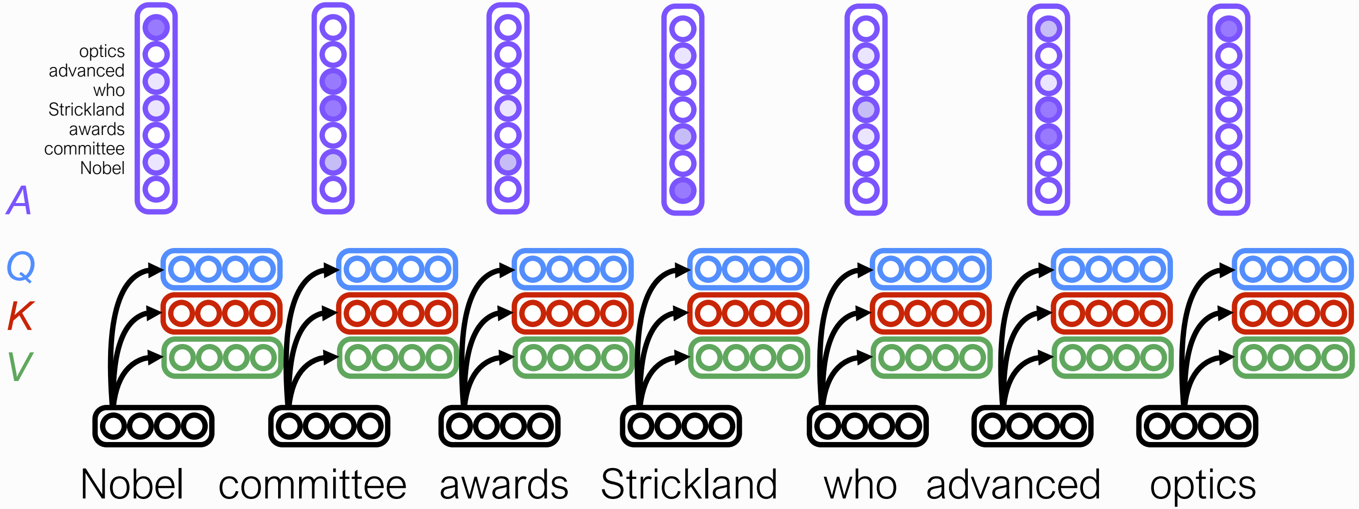
Source: Emma Strubell

Transformer self-attention



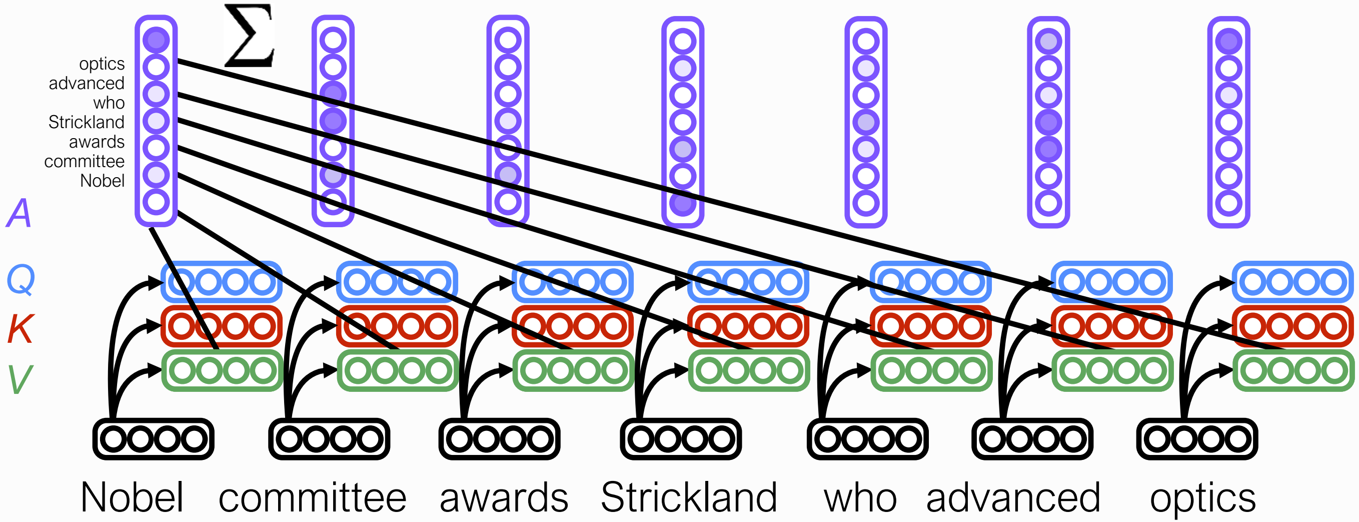
Source: Emma Strubell

Transformer self-attention



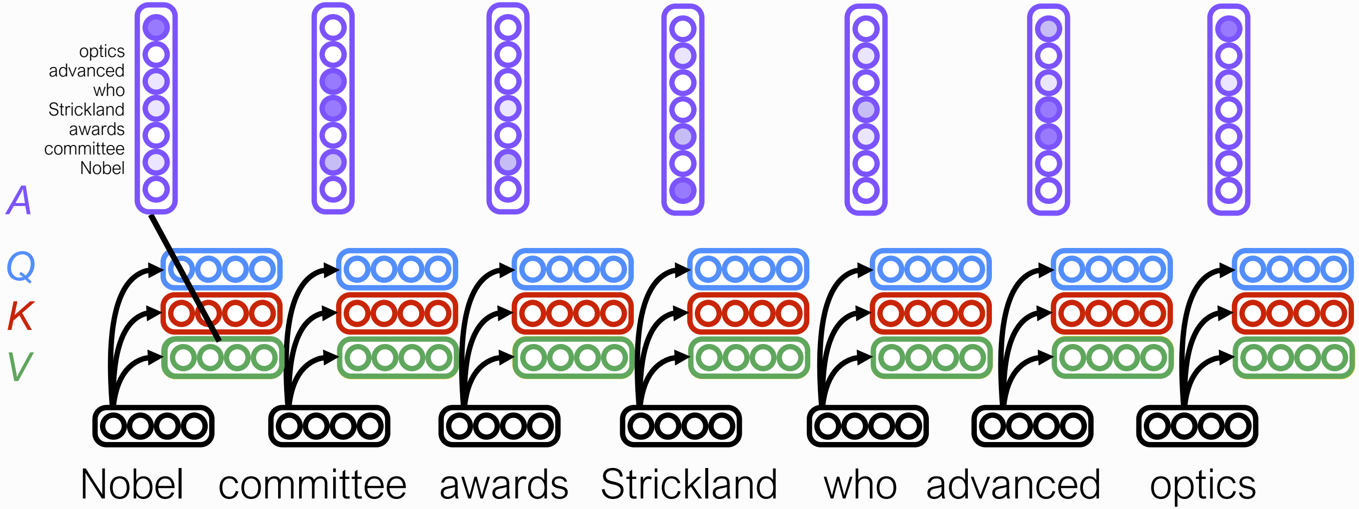
Source: Emma Strubell

Transformer self-attention



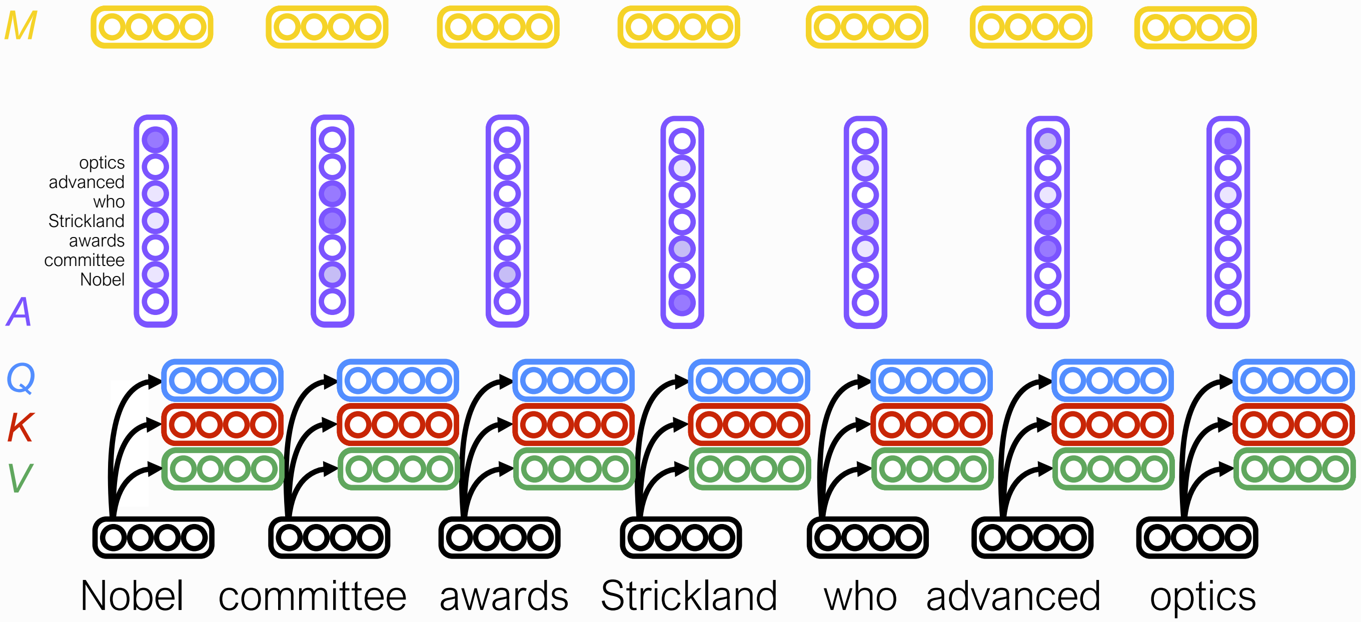
Source: Emma Strubell

Transformer self-attention



Source: Emma Strubell

Transformer self-attention



Source: Emma Strubell

Activity: work through self-attention

Calculate transformed output for one input word

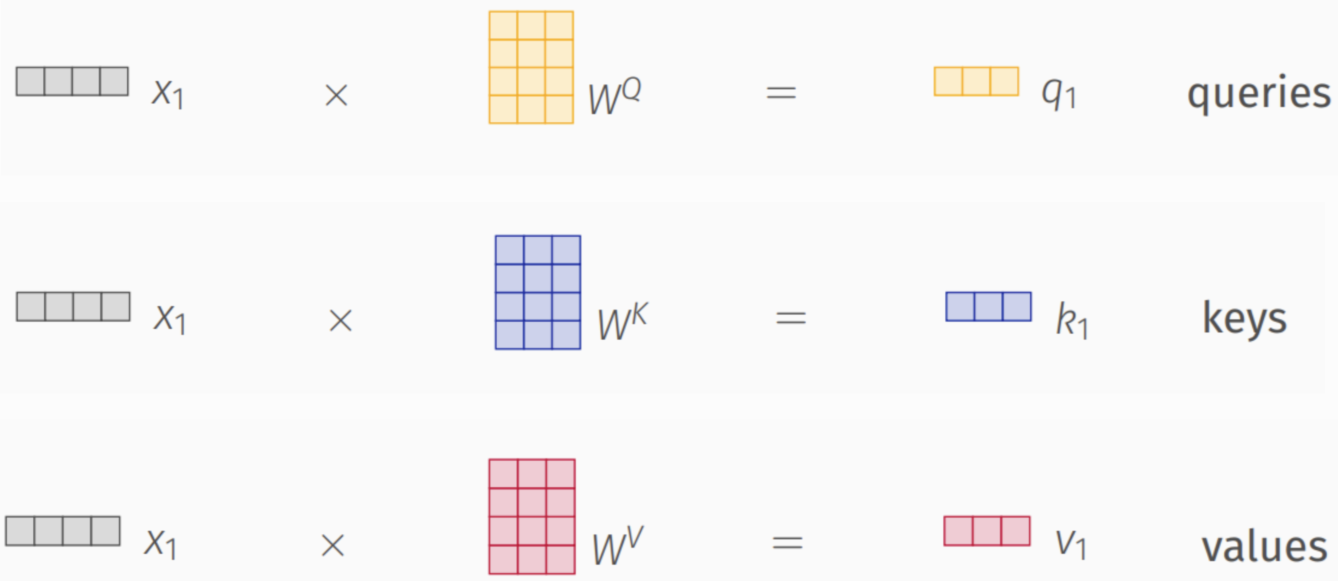
- Example sentence: “we wash our cats” (don’t ask)
- Let’s just calculate the vector output, for one input word: “we”
- High-level points to remember before you get buried in the math:
  - Each token will have an output vector that integrates contextual information from other tokens in the sentence
  - Each token can play a role as a query, key, and value
- Parameters (learned through backpropagation) are assumed given:
  - W^Q, W^K, W^V

Computing Self-Attention, Step One: Compute Key, Query, and Value Vectors

d_x-dimensional embeddings

$d_k \times d_x$-dimensional Weight Matrices

d_k-dimensional vectors



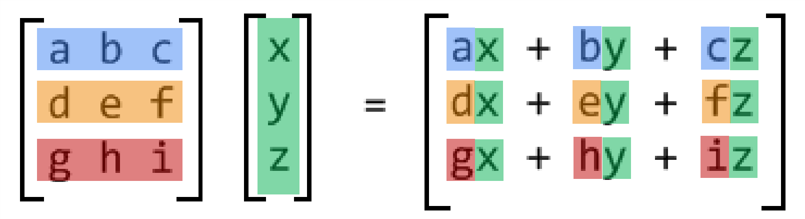
$x_1 \times W^Q = q_1$ queries

$x_1 \times W^K = k_1$ keys

$x_1 \times W^V = v_1$ values

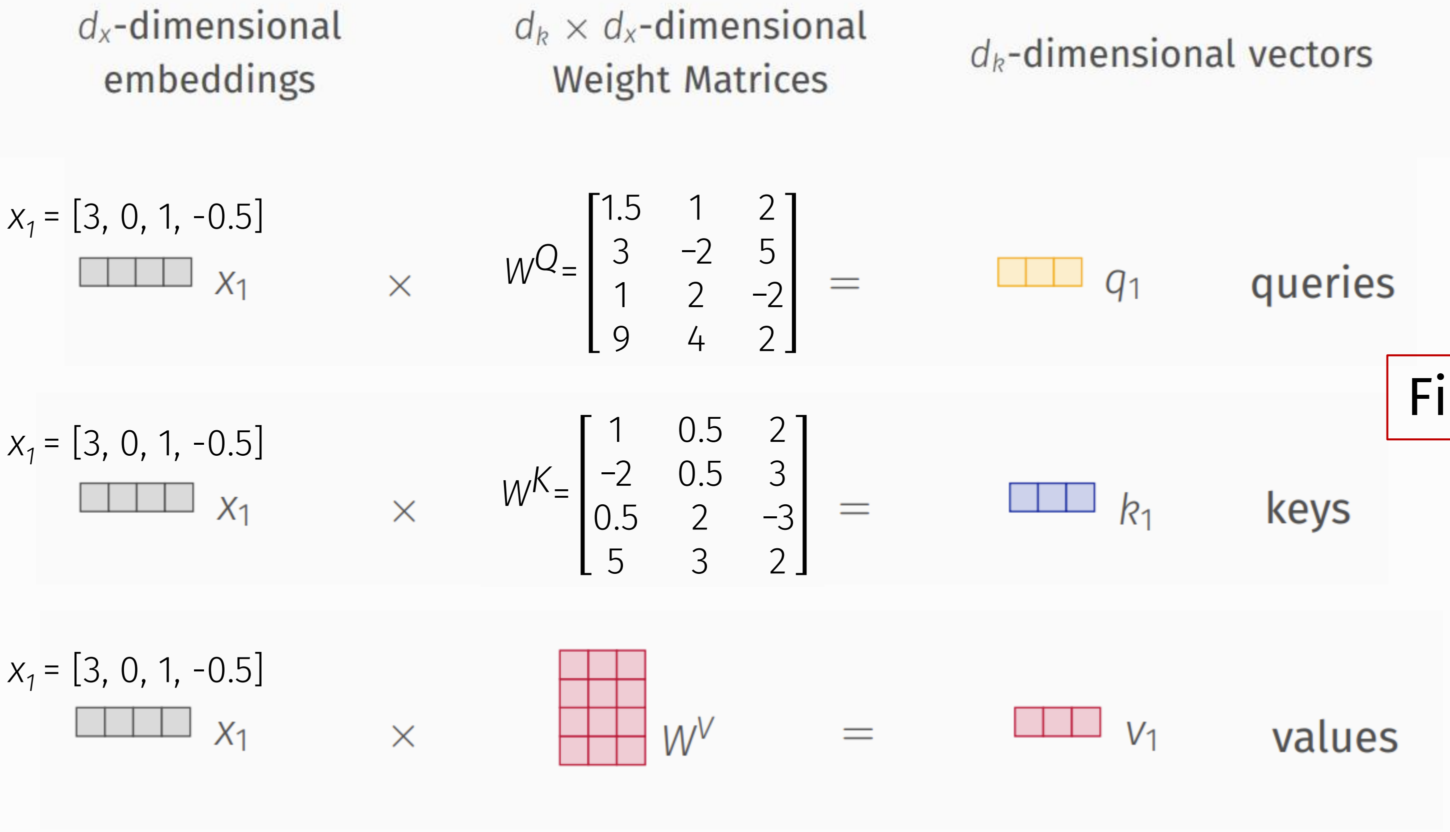
Slide credit: David Mortensen

Dot product: vector · matrix



$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

Computing Self-Attention, Step One: Compute Key, Query, and Value Vectors



d_x-dimensional embeddings

$d_k \times d_x$-dimensional Weight Matrices

d_k-dimensional vectors

x_1 = [3, 0, 1, -0.5]

$$x_1 \times W^Q = \begin{bmatrix} 1.5 & 1 & 2 \\ 3 & -2 & 5 \\ 1 & 2 & -2 \\ 9 & 4 & 2 \end{bmatrix} = q_1$$

queries

x_1 = [3, 0, 1, -0.5]

$$x_1 \times W^K = \begin{bmatrix} 1 & 0.5 & 2 \\ -2 & 0.5 & 3 \\ 0.5 & 2 & -3 \\ 5 & 3 & 2 \end{bmatrix} = k_1$$

keys

x_1 = [3, 0, 1, -0.5]

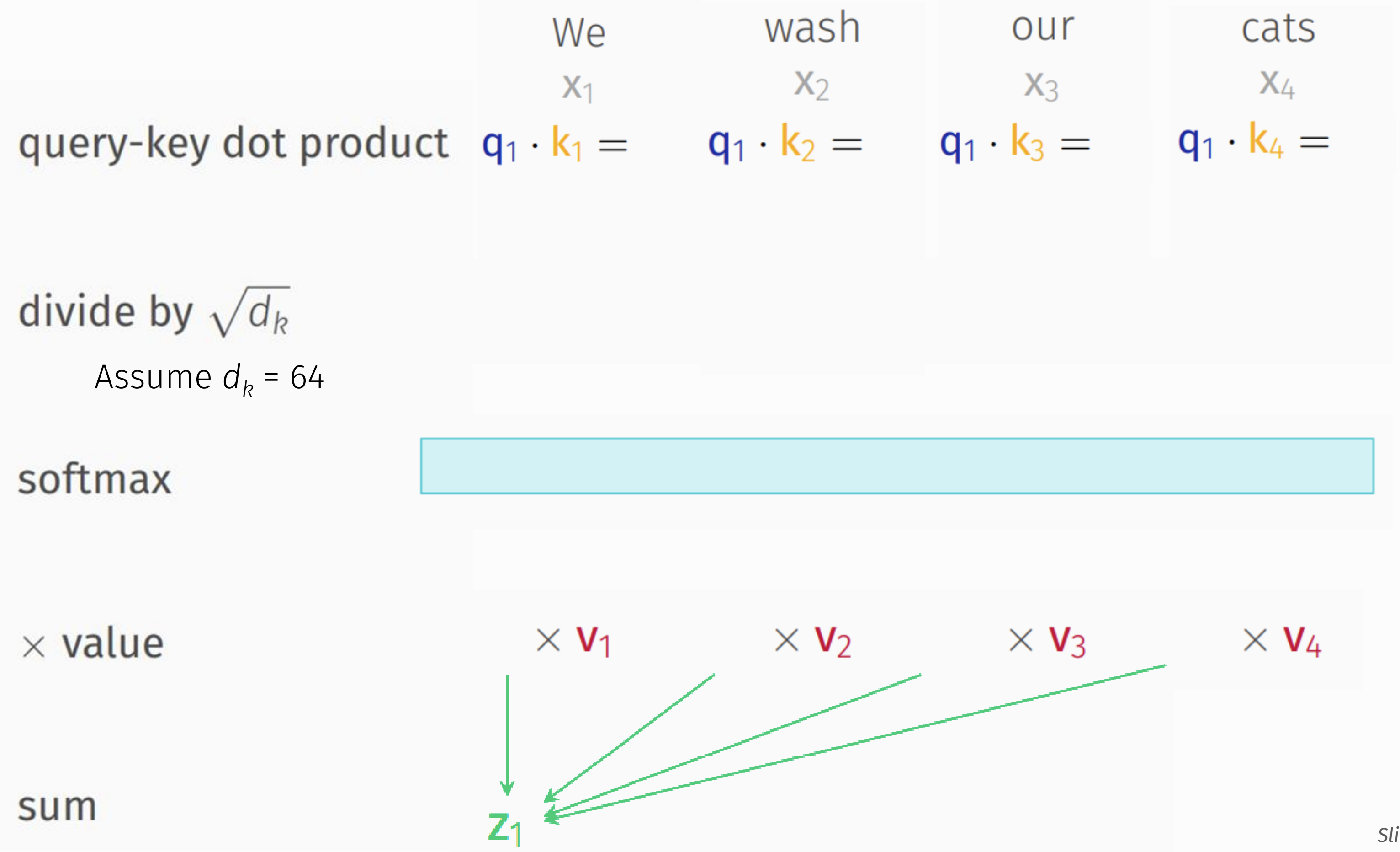
$$x_1 \times W^V = v_1$$

values

Find q_1 and k_1

Slide credit: David Mortensen

Computing Self-Attention, Step Two: Weighted Sum of Value Vectors



	We	wash	our	cats
	x_1	x_2	x_3	x_4
query-key dot product	$q_1 \cdot k_1 =$	$q_1 \cdot k_2 =$	$q_1 \cdot k_3 =$	$q_1 \cdot k_4 =$
divide by $\sqrt{d_k}$ Assume d_k = 64				
softmax				
× value	× v_1	× v_2	× v_3	× v_4
sum	z_1			

$k_2 = [3, 4, 3]$
$k_3 = [5, 2, 3]$
$k_4 = [3, 2, 1]$

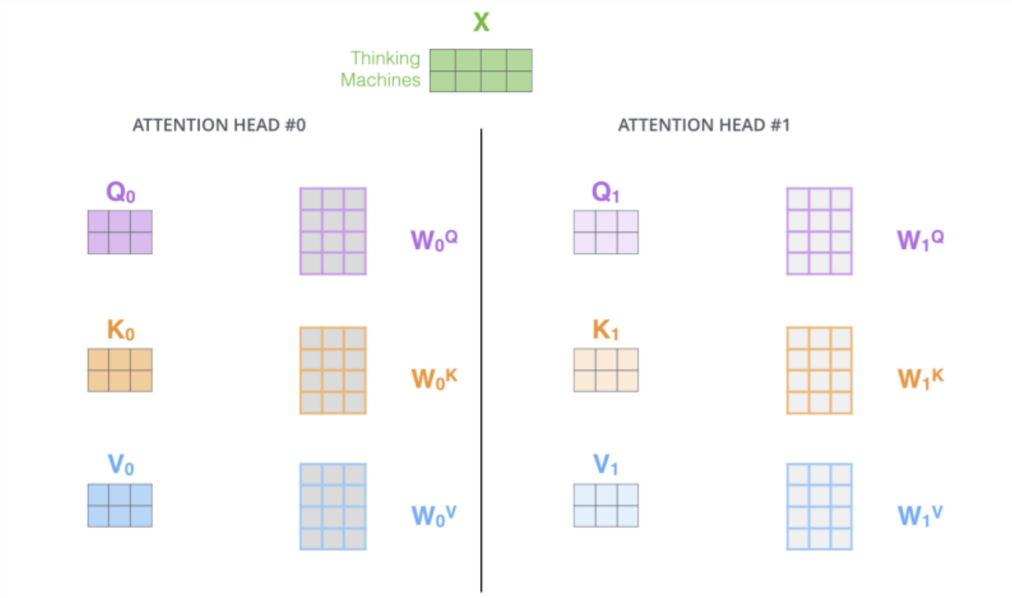
$v_1 = [1, 0.5, -1]$
$v_2 = [4, 5, -2]$
$v_3 = [-3, 2, 2]$
$v_4 = [1, 1, 6]$

Slide credit: David Mortensen

Multi-headed attention

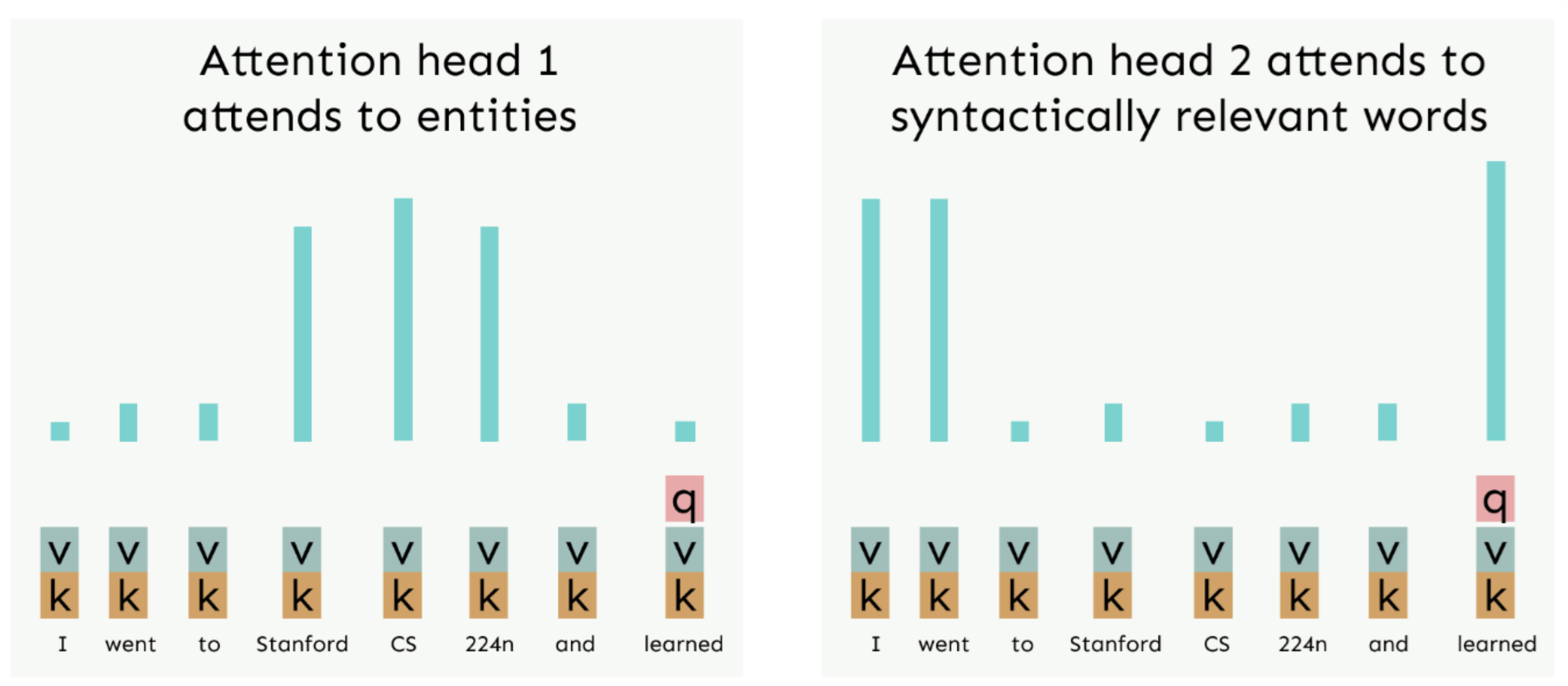
Multi-Headed Attention Expands Transformer Models' Ability to Focus on Different Positions

Maintain distinct weight matrices for each attention head—distinct representational subspaces:



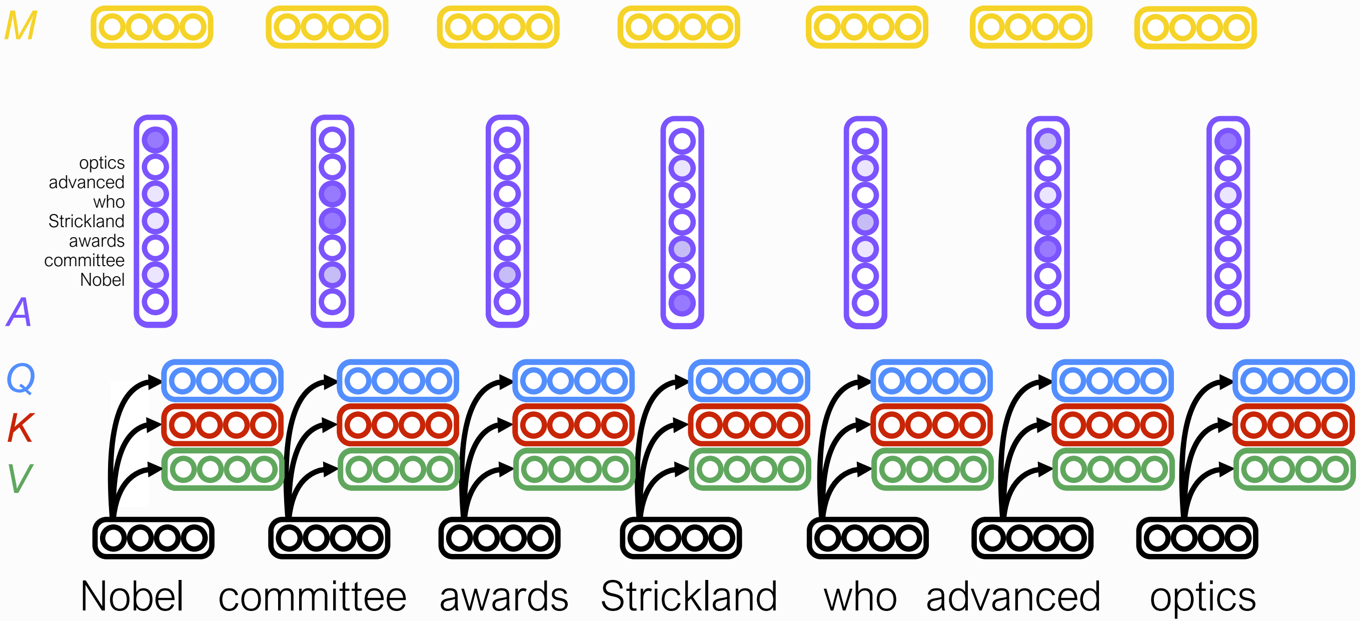
Slide credit: David Mortensen

Hypothetical example of multi-headed attention


Attention head 1 attends to entities
q
v v v v v v v v
k k k k k k k k
I went to Stanford CS 224n and learned
Attention head 2 attends to syntactically relevant words
q
v v v v v v v v
k k k k k k k k
I went to Stanford CS 224n and learned

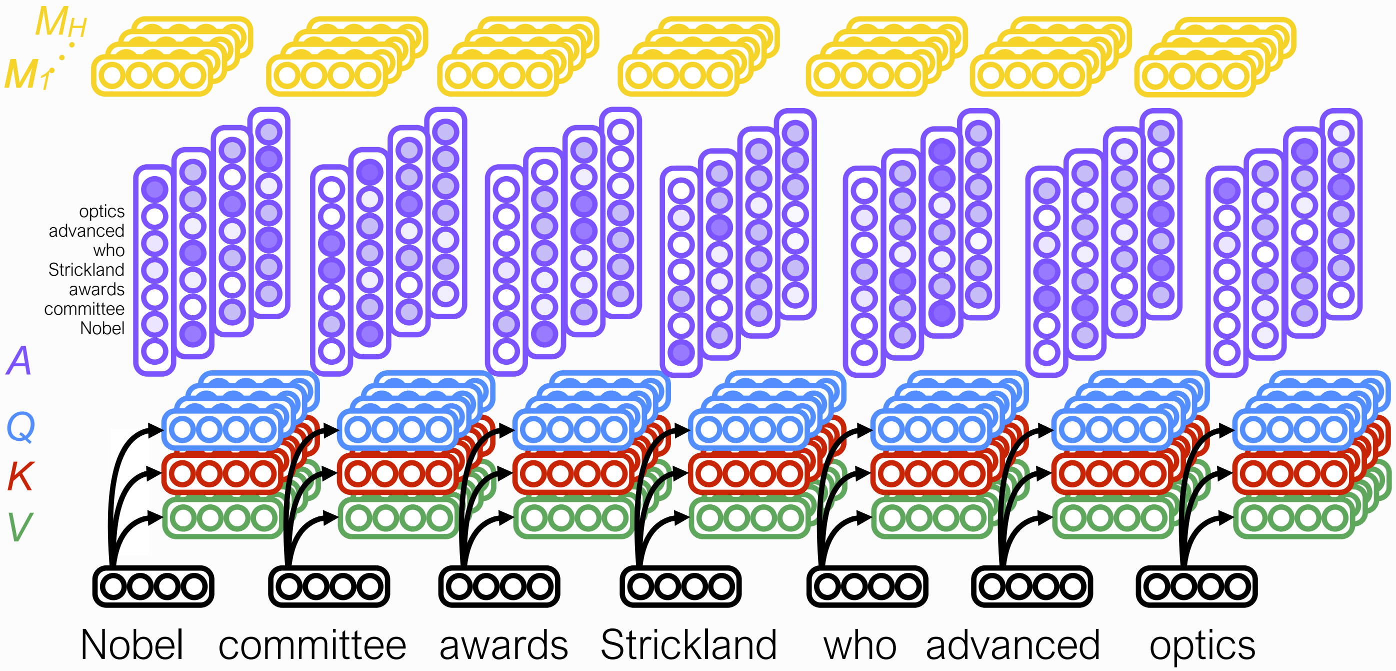
Slide credit: John Hewitt

Transformer self-attention



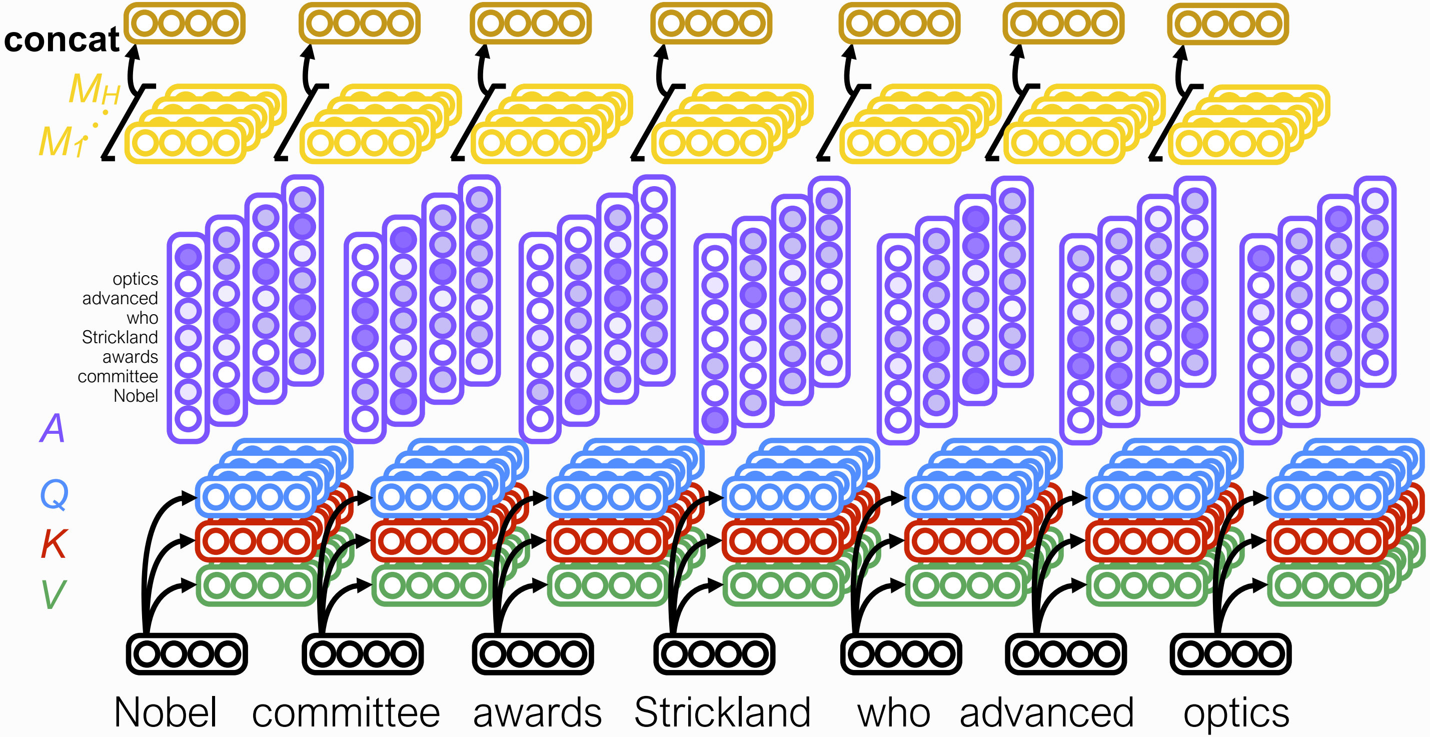
Source: Emma Strubell

Multi-head self-attention



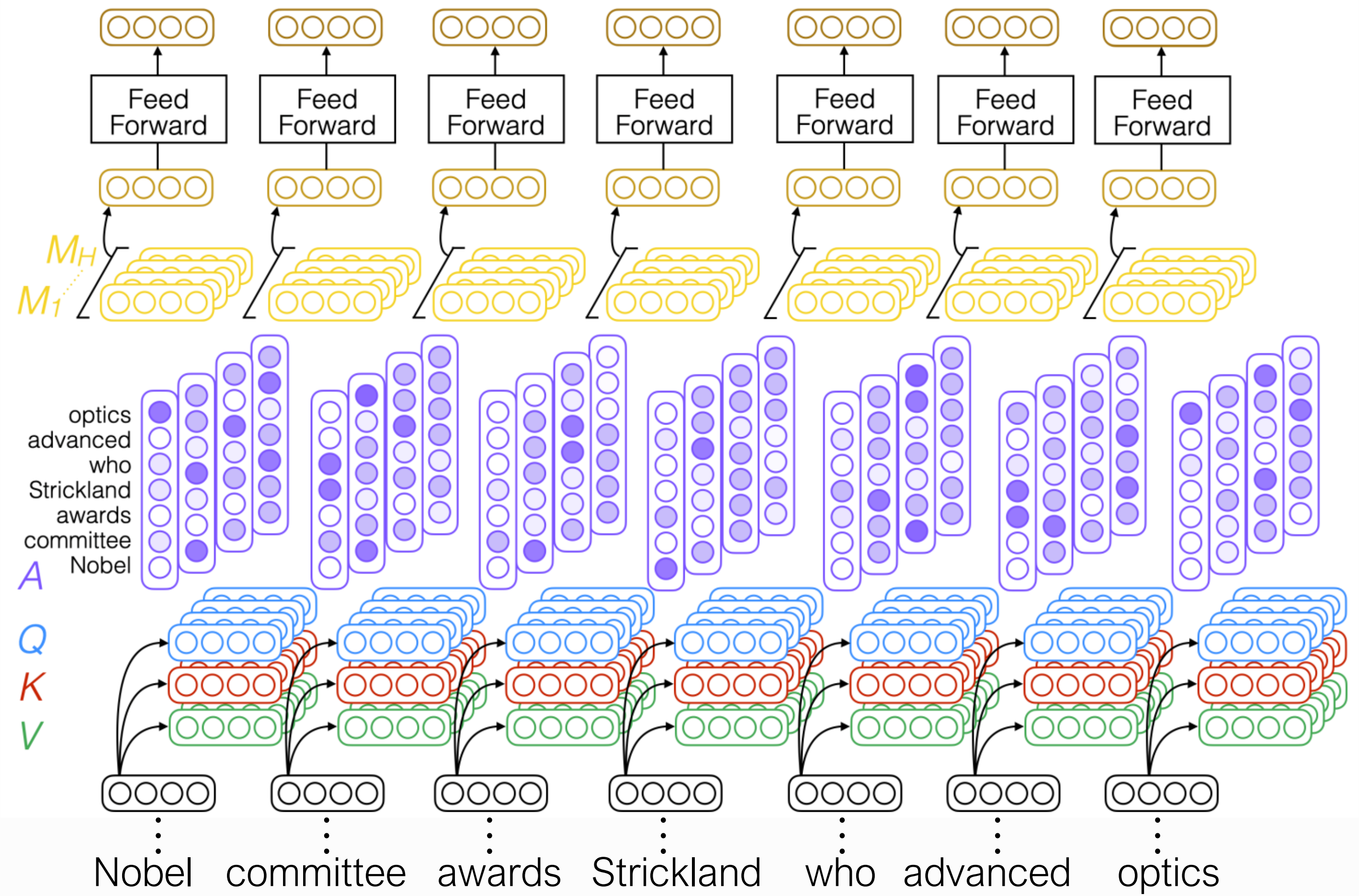
Source: Emma Strubell

Multi-head self-attention



Source: Emma Strubell

Add a feedforward neural transformation for nonlinearity



Source: Emma Strubell

Wrapping up

- Transformers are a high-performing NLP architecture based on self-attention
- Transformers produce one output vector per input token
- Output vectors from transformers integrate information from the surrounding tokens (self-attention)
- Self-attention computation is easily parallelizable

Questions?