04. Transformers

Transformers may not fix all your NLP problems.

.

.

.

But they are worth some attention.

Source: Ramsri Goutham

CS 1671/2071
Human Language Technologies

Session 16: Transformers part 1

Michael Miller Yoder

March 17, 2025

University of Pittsburgh | School of Computing and Information

Course logistics

- I will release the quiz for this week today, will be **due this Thu Mar 20**
- Homework 3 will be released this week, probably Fri Mar 21. Is due Apr 9
- There is a new textbook version, released 2025-01-12. I think it is largely the same as the old one from 2024-08-20 but just with typos fixed
  - If you notice anything strange with the alignment of the readings, let Michael know

Course logistics

- I pushed back the due date for the project progress report, now **due next Thu Mar 27**. Instructions for that are posted on the project website
  - **Part 1:** Data statistics and exploratory data analysis (EDA)
  - **Part 2:** A result from baseline/initial approach
  - **Part 3:** Proposal on how to use LLMs for your task
  - **Part 4:** Open questions and challenges
  - I will set up a Canvas page for submissions soon
  - I am in the process of setting up OpenAI API account to use ($150 for class). In the meantime look into using Gemini free credits or other LLMs

NLP talk this Wed: Anjalie Field

Anjalie Field

Johns Hopkins University

Time: 03/19/2025, 4:30 - 5:30 PM (EST)

Place: In-person (SQ 5317)

Bio: Anjalie Field is an Assistant Professor in the Computer Science Department at Johns Hopkins University. She is also affiliated with the Center for Language and Speech Processing (CLSP) and the new Data Science and AI Institute. Her research focuses on the ethics and social science aspects of natural language processing, which includes developing models to address societal issues like discrimination and propaganda, as well as critically assessing and improving ethics in AI pipelines. Her work has been published in NLP and interdisciplinary venues, like ACL and PNAS, and in 2024 she was named an AI2050 Early Career Fellow by Schmidt Futures. Prior to joining JHU, she was a postdoctoral researcher at Stanford, and she completed her PhD at the Language Technologies Institute at Carnegie Mellon University.

Title: Fairness and Privacy in High-Stakes NLP

Abstract: Practitioners are increasingly using algorithmic tools in high-stakes settings, like healthcare, social services, policing, and education with particular recent interest in natural language processing (NLP). These domains raise a number of challenges, including preserving data privacy, ensuring model reliability, and developing approaches that can mitigate, rather than exacerbate historical bias. In this talk, I will discuss our recent work investigating risks of racial bias in NLP child protective services and ways we aim to better preserve privacy for these types of audits in the future. Time permitting, I will also discuss, our development of speech processing tools for policy body camera footage, which aims to improve police accountability. Both domains involve challenges in working with messy minimally processed data containing sensitive information and domain-specific language. This work emphasizes how NLP has potential to advance social justice goals, like police accountability, but also risks causing direct harm by perpetuating bias and increasing power imbalances.

Lecture overview: Transformers part 1

- Self-attention
- Multi-headed attention
- Transformer blocks
- Activity: work through self-attention

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

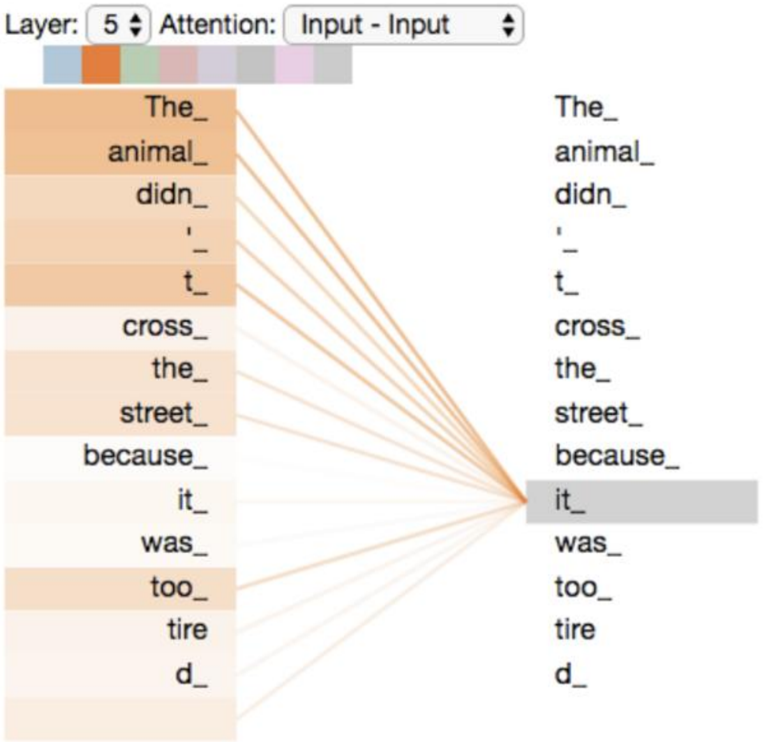
Slide adapted from Jurafsky and Martin

Intuition of attention

- Build up the contextual embedding from a word by selectively integrating information from all the neighboring words
- We say that a word "attends to" some neighboring words more than others

Self-attention

Self-attention illustrated



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: instead of using vectors (like x_i and x_4) directly, we'll represent 3 separate roles each vector $\mathbf{x}_i$ plays:

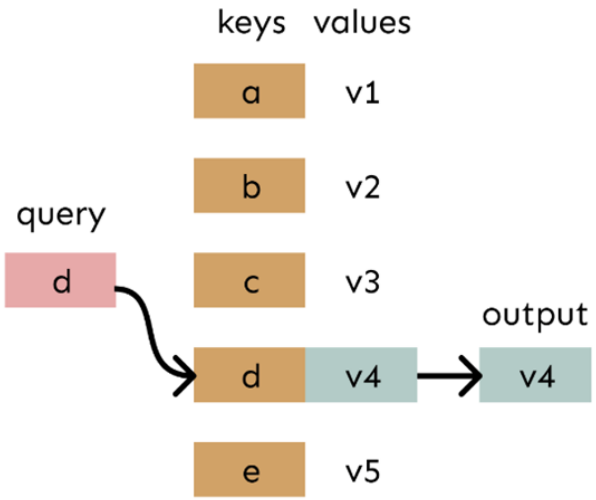
- **query:** As *the current element* being compared to the other inputs.
- **key:** as *an input* that is being compared to the current element to determine a similarity
- **value:** a value of a preceding element that gets weighted and summed

Slide adapted from Jurafsky and Martinc

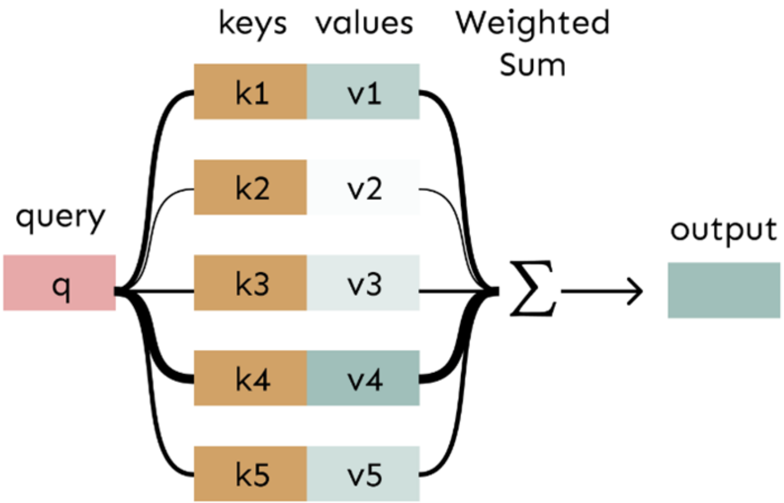
Attention as a soft, averaging lookup table

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.



9

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 Martinc

An Actual Attention Head: slightly more complicated

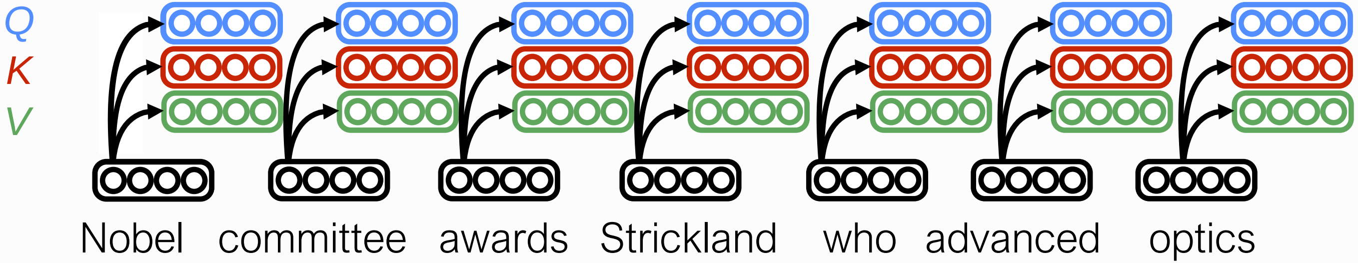
- Given these 3 representation of x_i

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

- 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

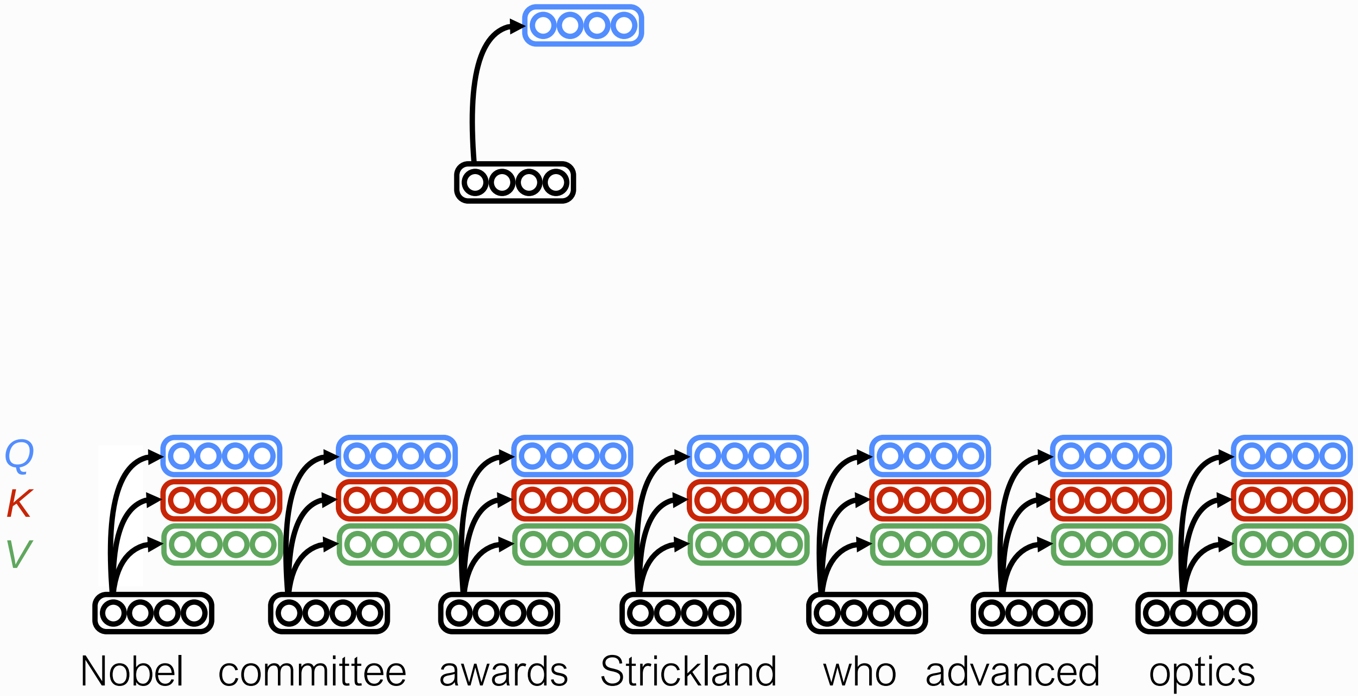
Slide adapted from Jurafsky and Martinc

Transformer self-attention



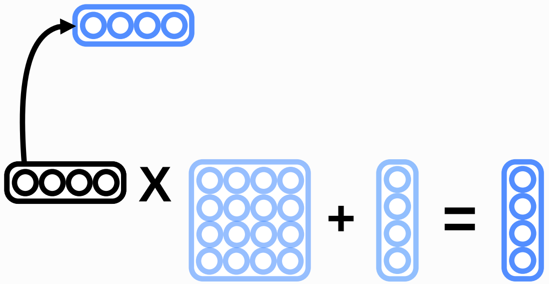
Source: Emma Strubell

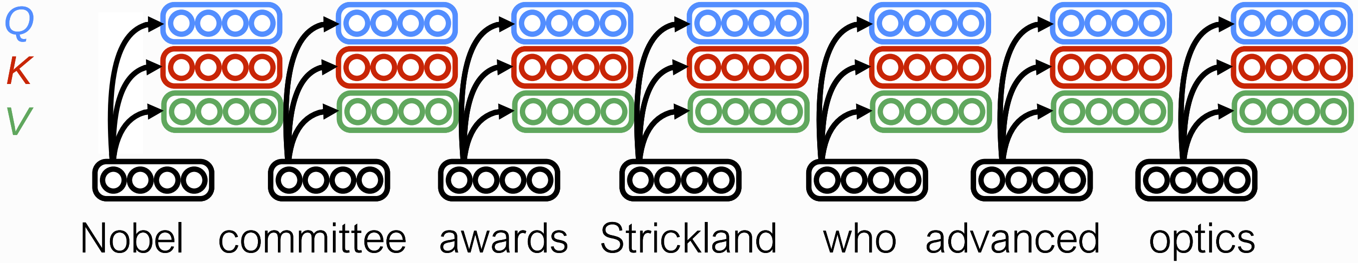
Transformer self-attention



Source: Emma Strubell

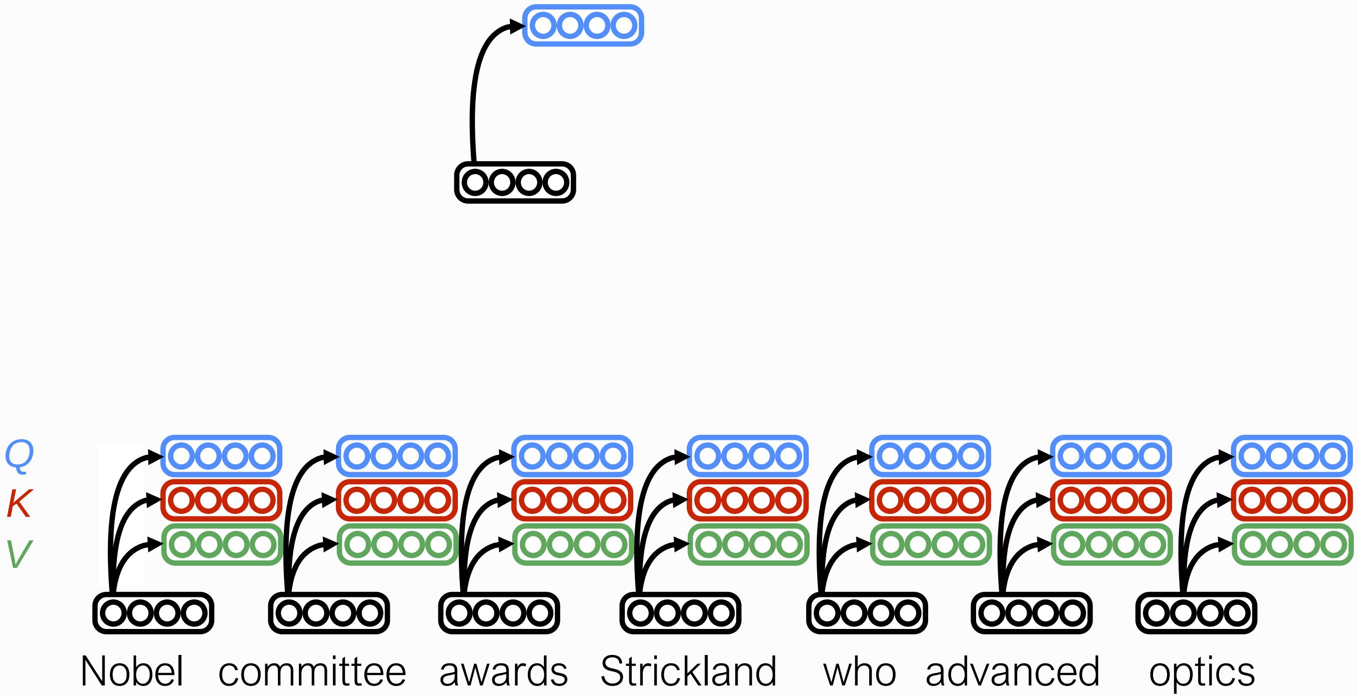
Transformer self-attention





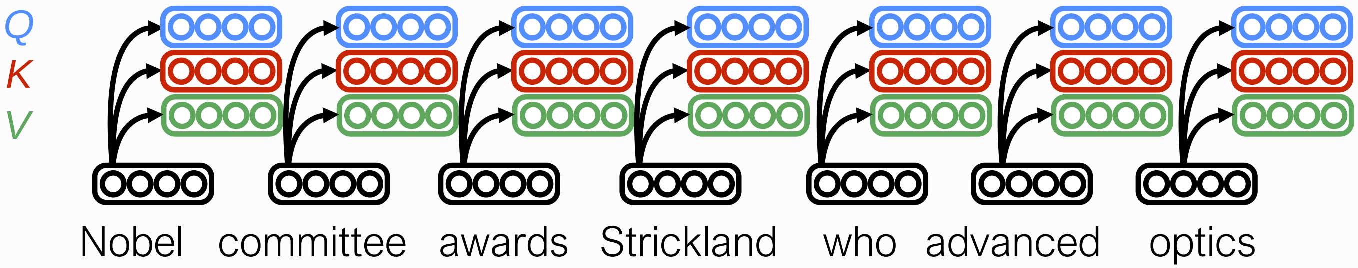
Source: Emma Strubell

Transformer self-attention



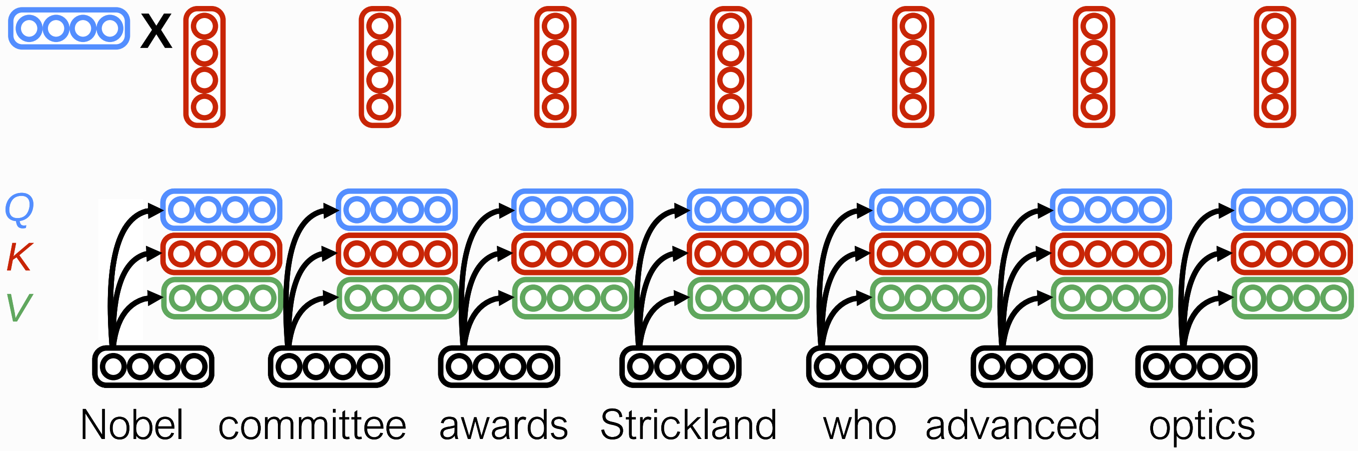
Source: *Emma Strubell*

Transformer self-attention



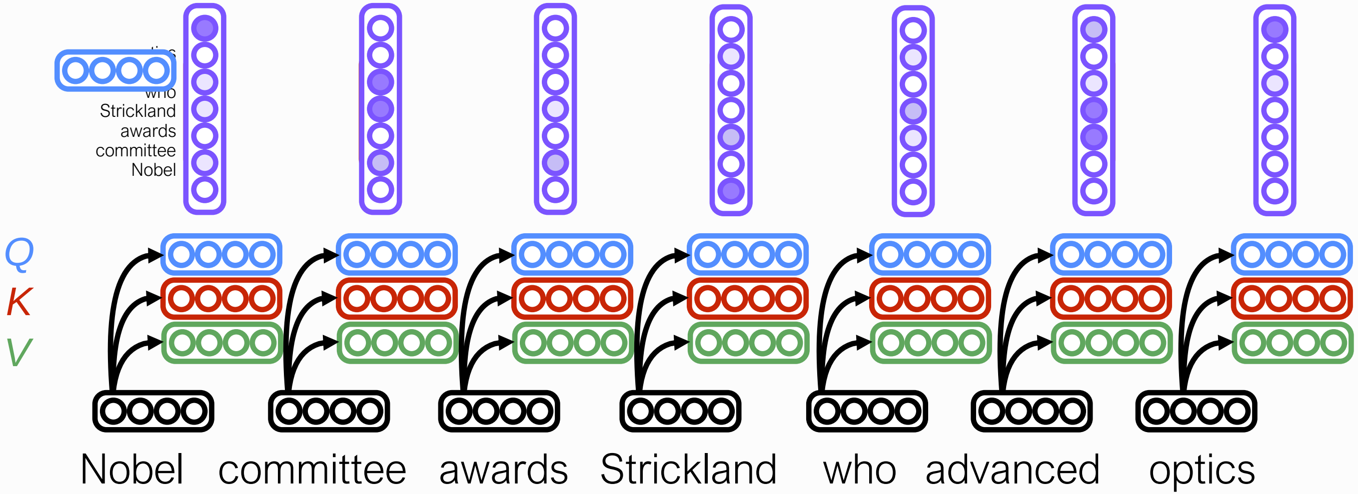
Source: *Emma Strubell*

Transformer self-attention



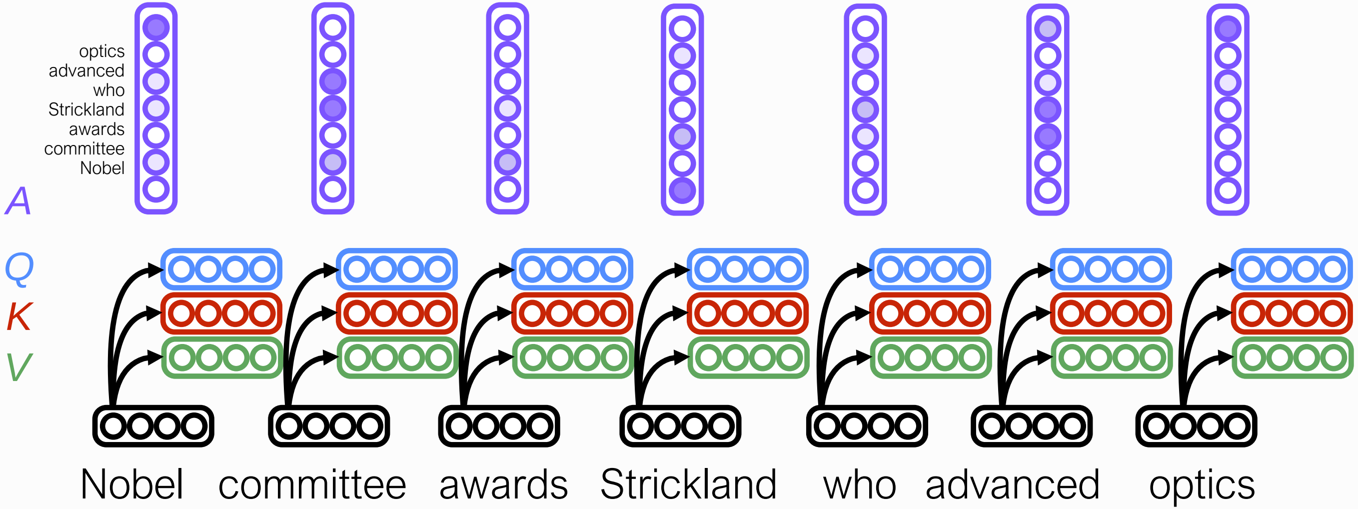
Source: Emma Strubell

Transformer self-attention



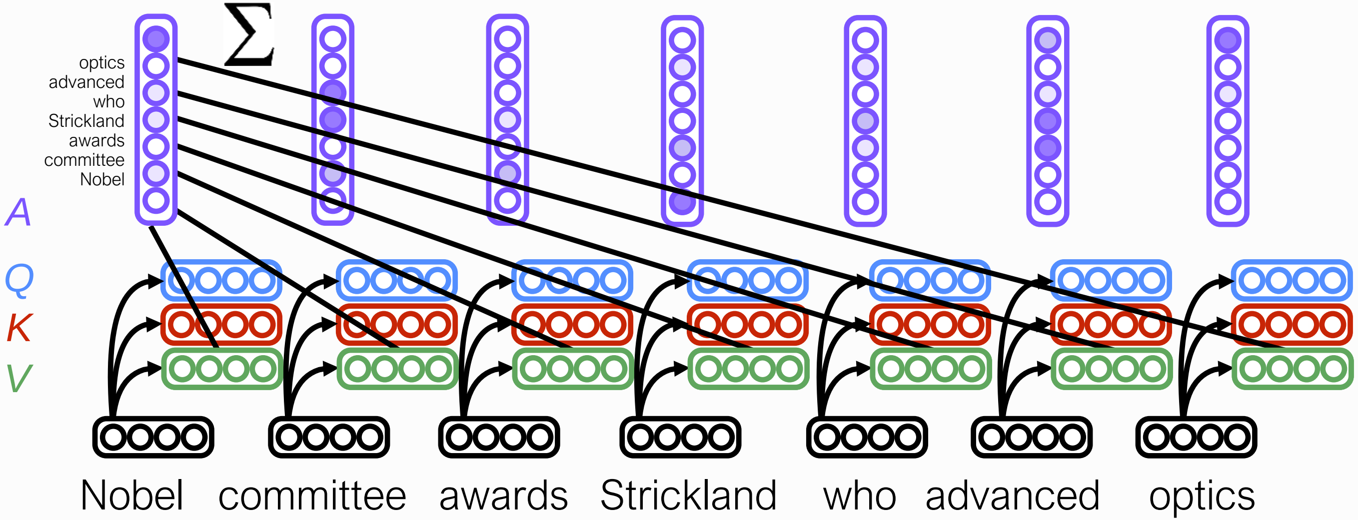
Source: Emma Strubell

Transformer self-attention



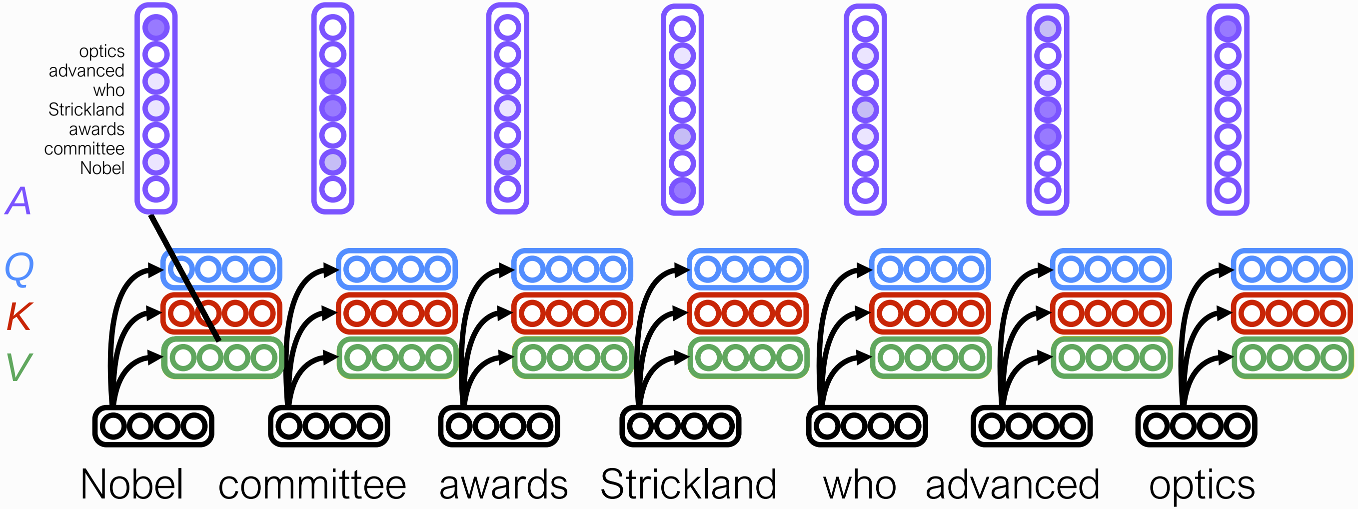
Source: Emma Strubell

Transformer self-attention



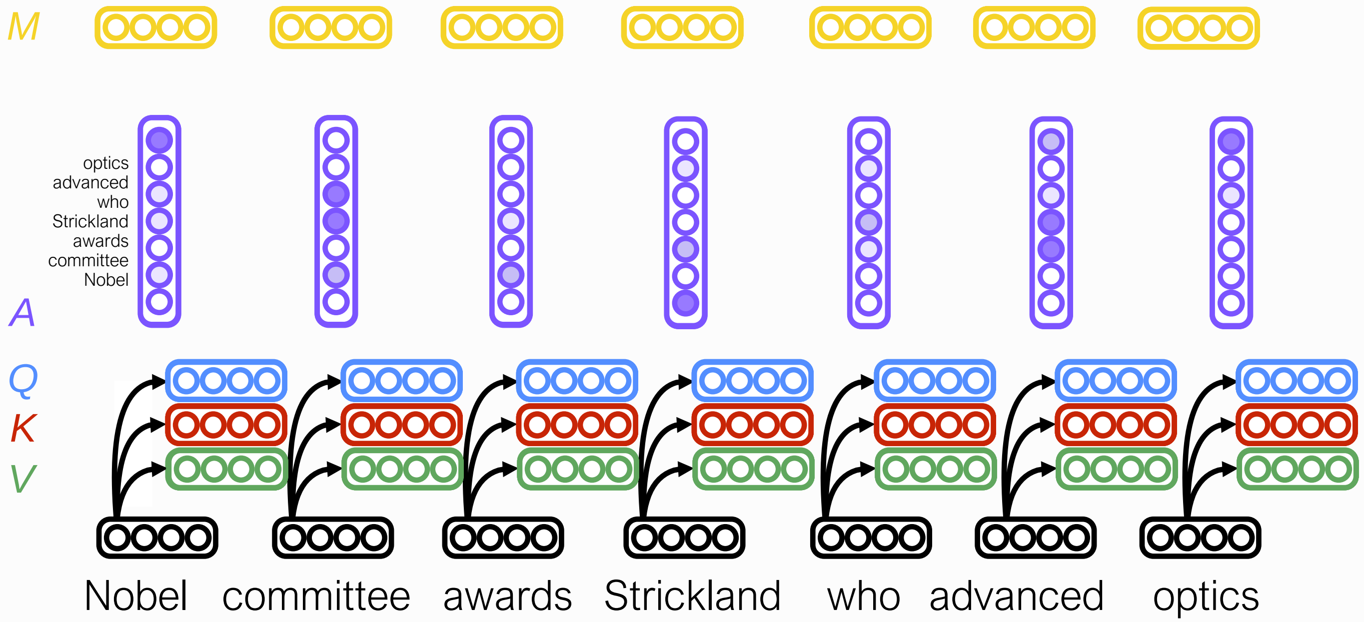
Source: Emma Strubell

Transformer self-attention



Source: Emma Strubell

Transformer self-attention

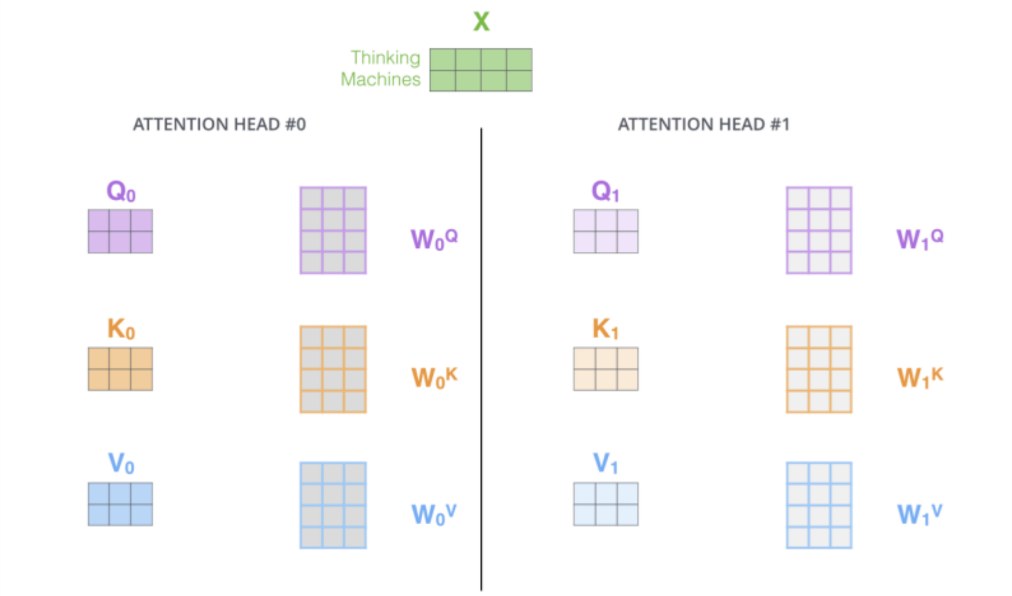


Source: Emma Strubell

Multi-headed attention

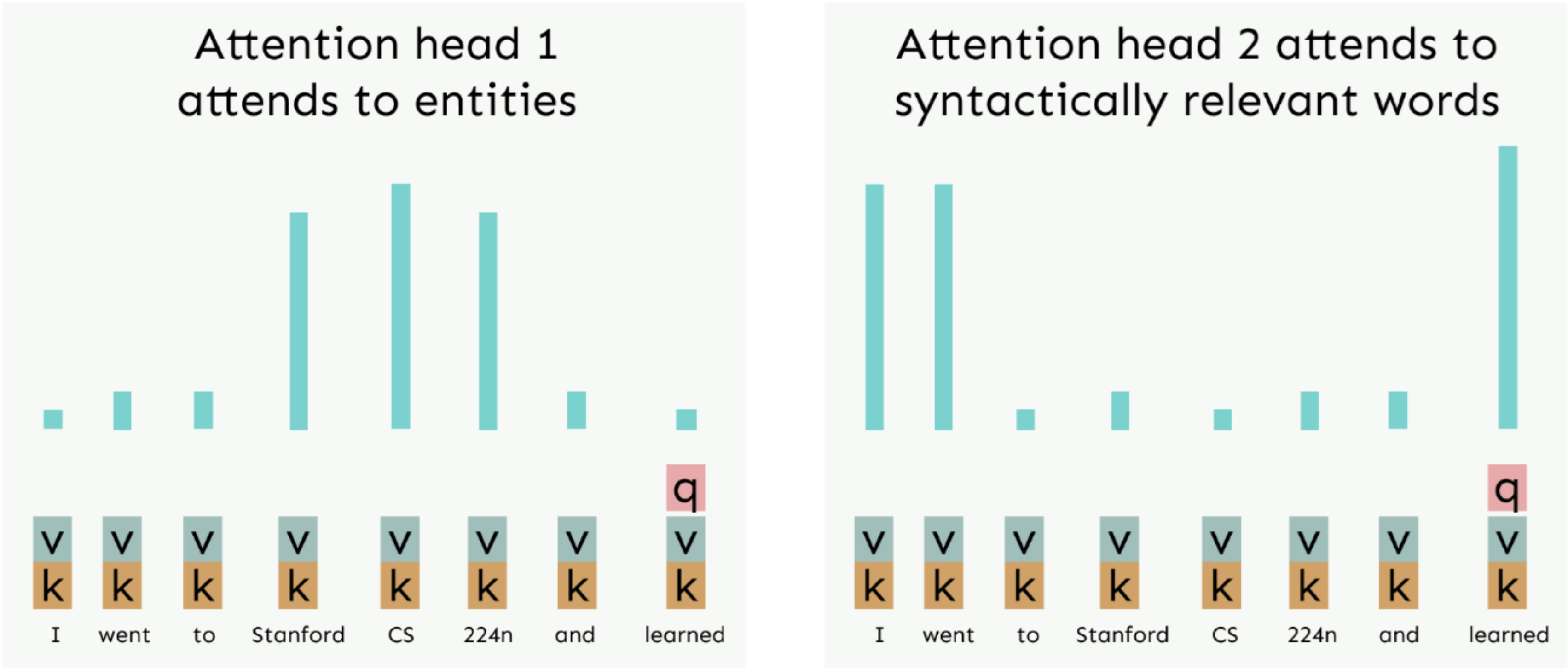
Multi-Headed Attention Expands Transformer Models' Ability to Focus on Different Positions

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



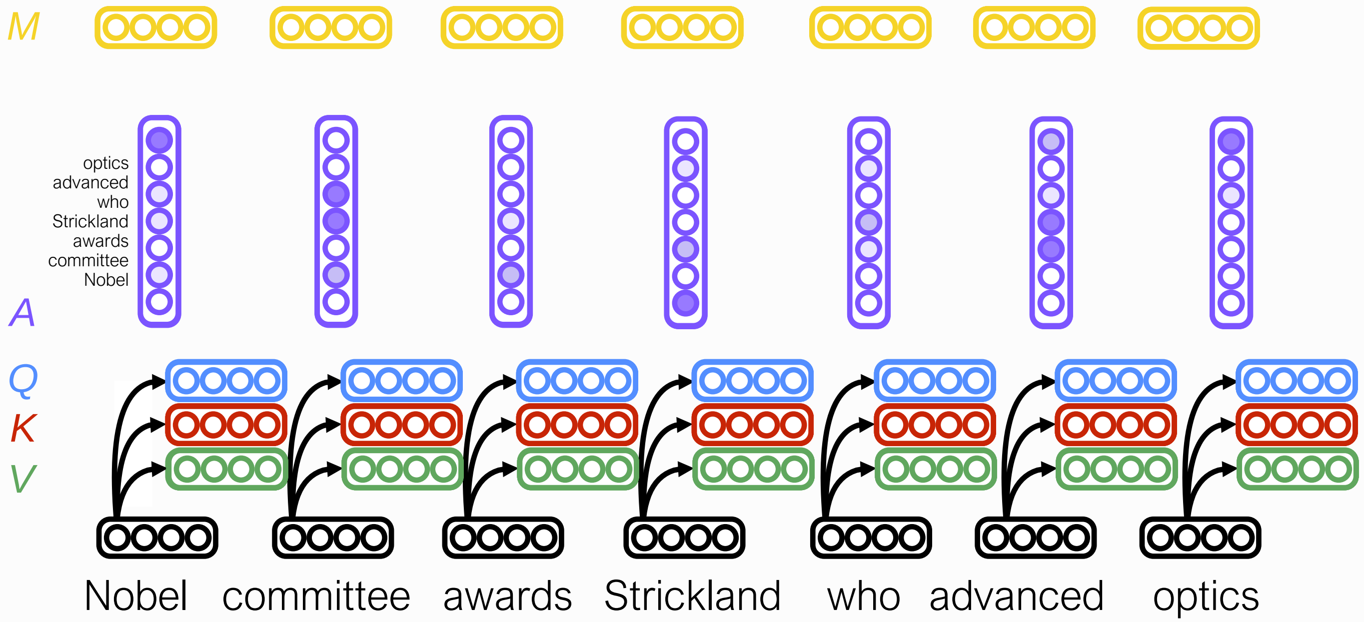
Slide credit: David Mortensen

Hypothetical example of multi-headed attention



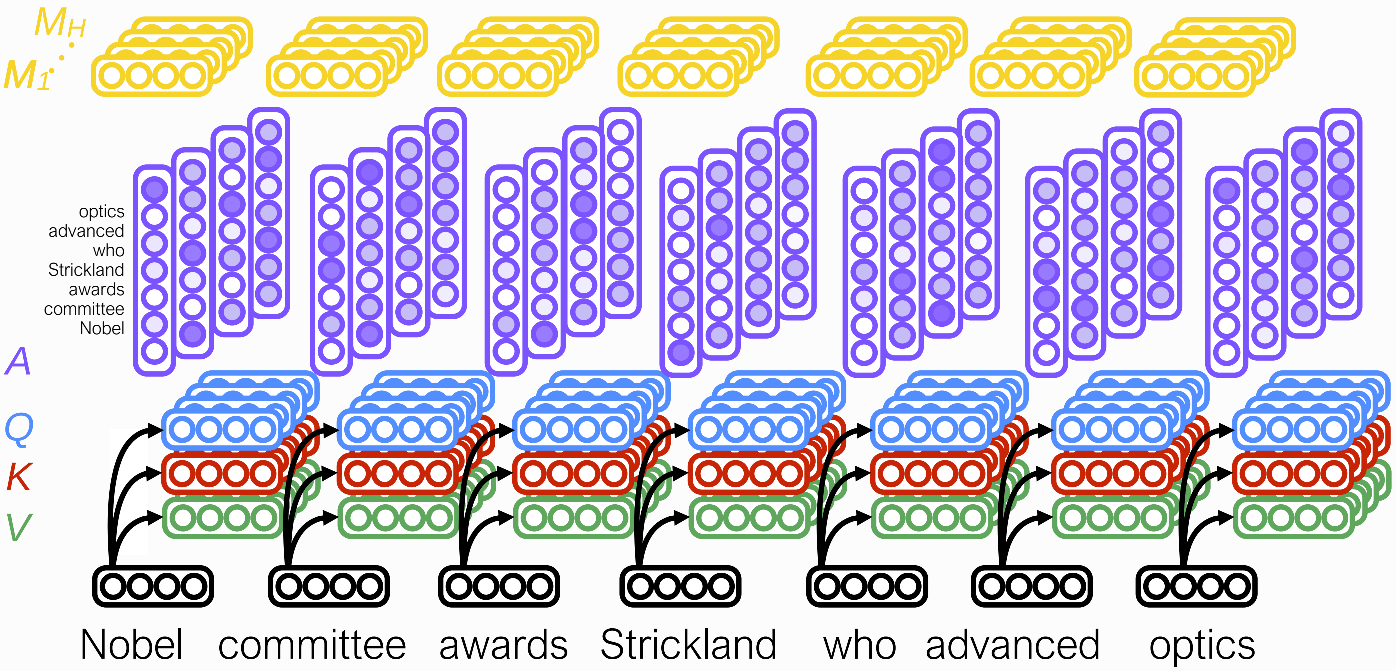
Slide credit: John Hewitt

Transformer self-attention



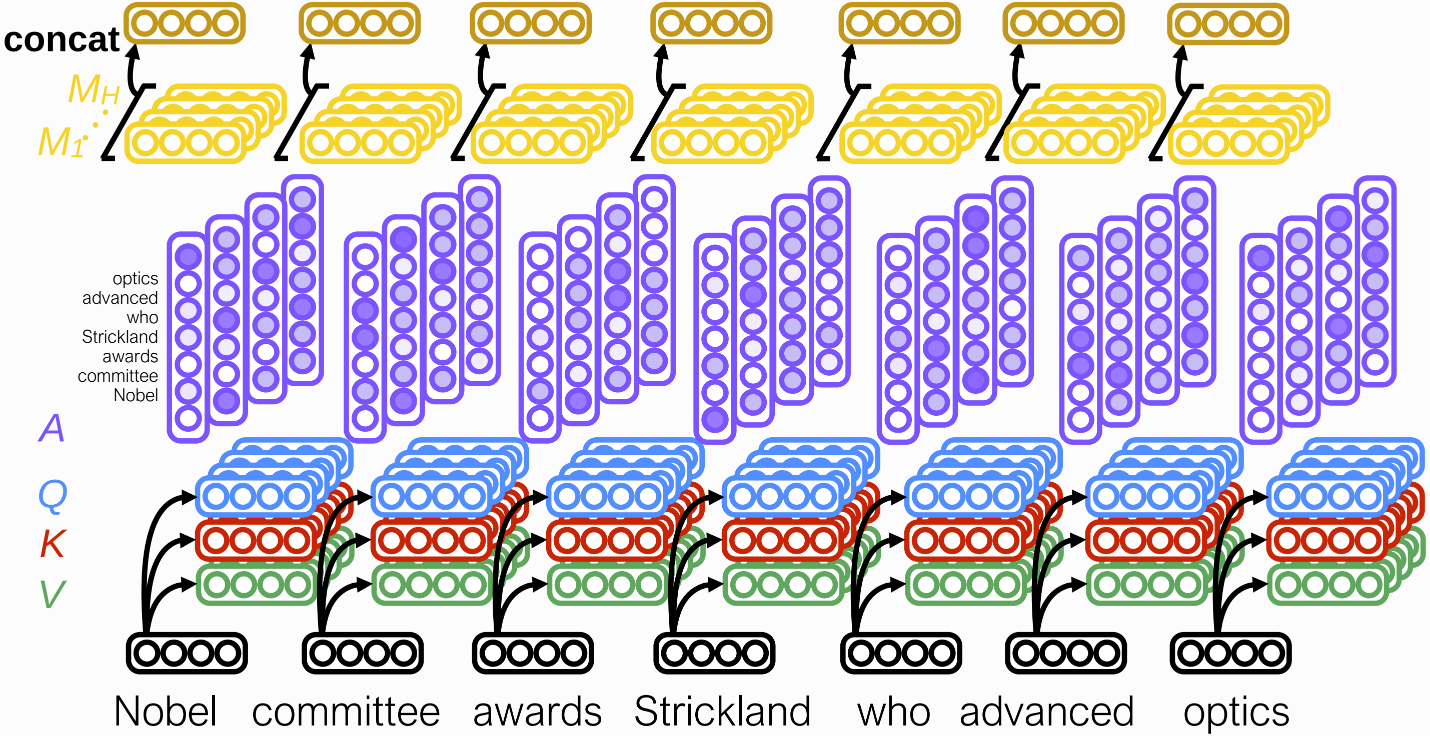
Source: Emma Strubell

Multi-head self-attention



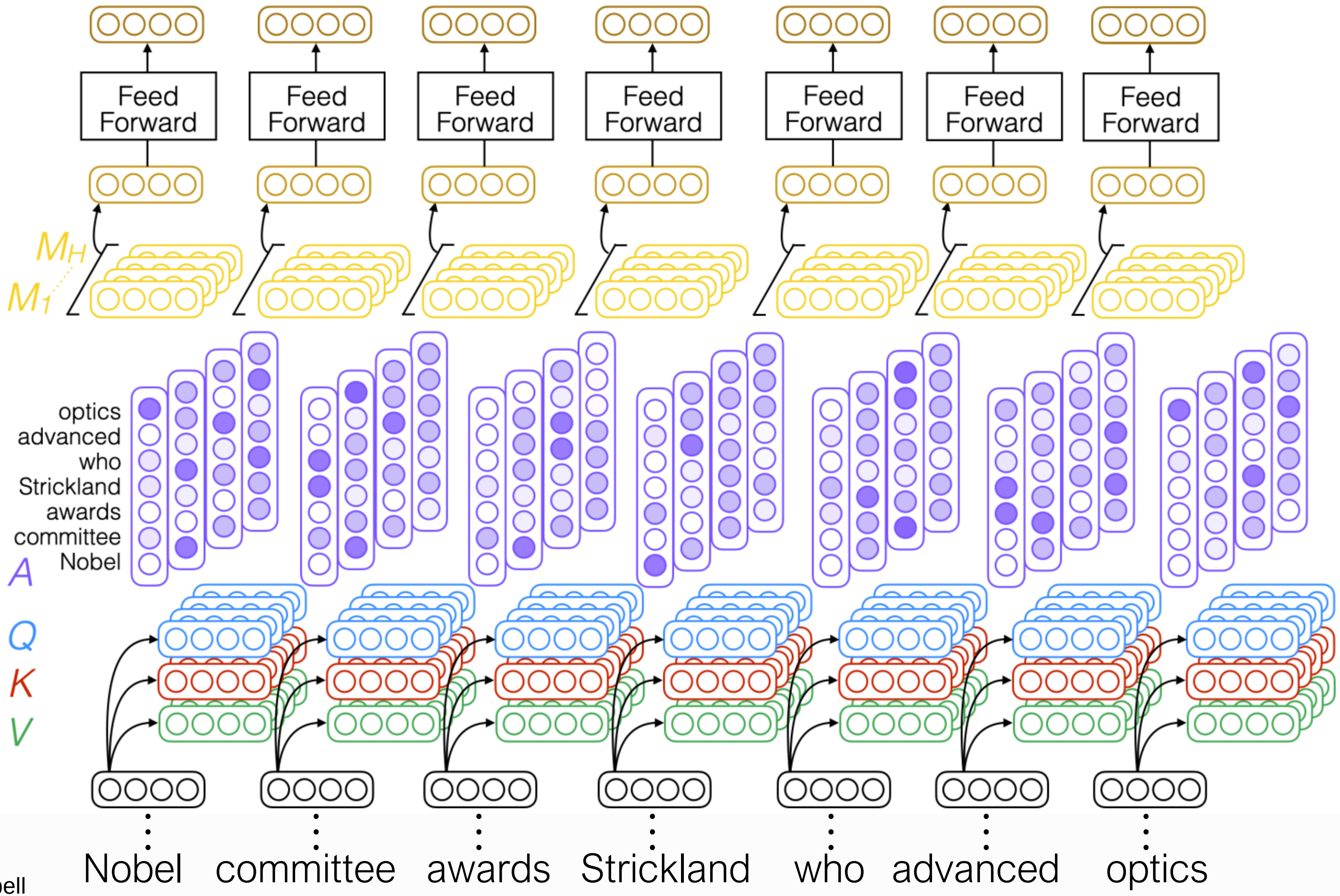
Source: Emma Strubell

Multi-head self-attention



Source: Emma Strubell

Add a feedforward neural transformation for nonlinearity



Source: Emma Strubell

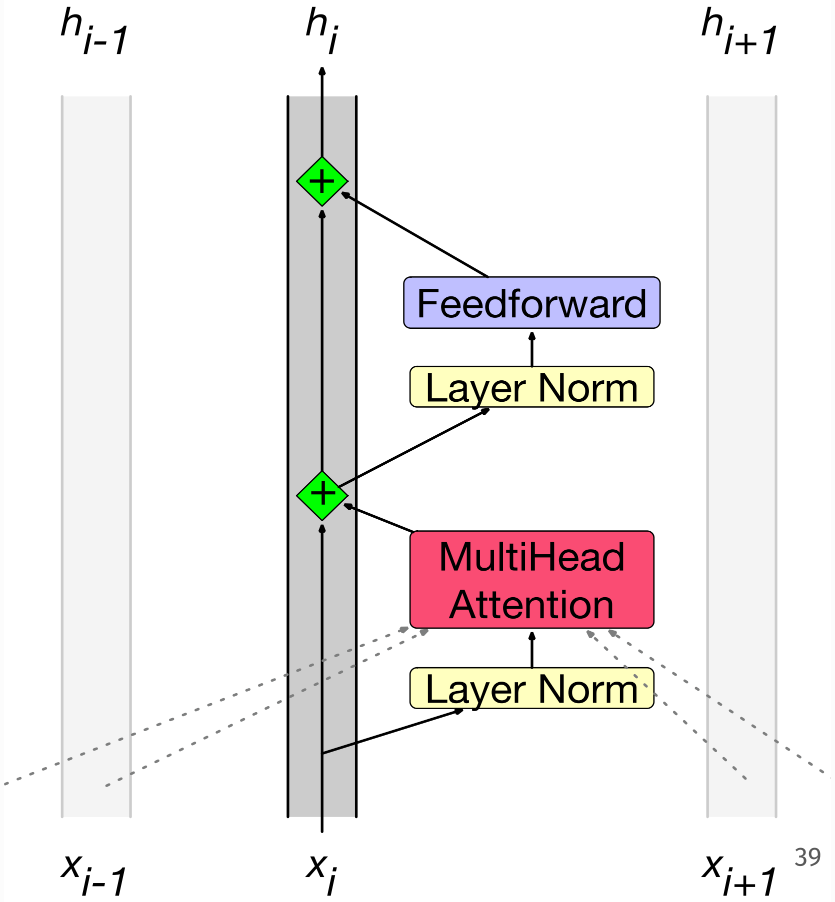
Transformer blocks

Transformer blocks

Each block consists of:

- Self-attention
- Layer normalization and residual connections: tricks to optimize learning
- Feedforward neural network

Output: 1 vector for every input token



Slide adapted from Jurafsky and Martin, John Hewitt

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	$\times$	$W^K = \begin{bmatrix} 1 & 0.5 & 2 \\ -2 & 0.5 & 3 \\ .5 & 2 & -3 \\ 5 & 3 & 2 \end{bmatrix}$	$=$	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				
$\times$ value	$\times$ v_1	$\times$ v_2	$\times$ v_3	$\times$ 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

Wrapping up

- Transformers are a high-performing NLP architecture based on self-attention
- Transformer blocks perfom a number of transformations on vectors for input tokens, including integrating information from the surrounding tokens (self-attention)
- Transformer blocks produce one output vector per each input token, which is contextual, i.e. varies depending on what words surround the token
- Self-attention computation is easily parallelizable

Questions?