Quiz

- Go to **Quizzes > Quiz 10-08** on Canvas
- You have until **2:40pm** to complete it
- Allowed resources
  - Textbook
  - Your notes (on a computer or physical)
  - Course slides and website
- Resources not allowed
  - Generative AI
  - Internet searches

04. Transformers

Transformers may not fix all your NLP problems.

.

.

.

But they are worth some attention.

Source: Ramsri Goutham

CS 2731
Introduction to Natural Language Processing

Session 13: Transformers part 1

Michael Miller Yoder

October 8, 2025

School of Computing and Information

Course logistics: homework

- Homework 2 is **due tomorrow, Thu Oct 9**

Course logistics: homework and project

- Next project deliverable: project proposal due **next Thu Oct 16**
  - Will include plans for **task, data, methods, evaluation**
  - Compare multiple approaches, including an LLM-based approach
  - Literature review of at least 3 related papers
    - Baselines to compare your approach to
  - Feel free to email or book office hours with Michael to discuss
- We have $150 total as a class to use on OpenAI LLM credits
- Access to open-source LLM set up on School of Computing and Information servers for API access is coming soon
  - Gemma, LLaMa, Deepseek

Structure of this course

MODULE 1	Introduction and text processing	text normalization, machine learning, NLP tasks	
	Approaches	How text is represented	NLP tasks
MODULE 2	statistical machine learning	n-grams	language modeling text classification
MODULE 3	neural networks	static word vectors	text classification
MODULE 4	transformers and LLMs	contextual word vectors	language modeling text classification
MODULE 5	Sequence labeling and parsing		
MODULE 6	NLP applications and ethics		

Midterm OMET feedback

6 responses out of 17 students (35%)

- What's been helping students learn
  - Lectures, discussions
  - Notebooks, including going through them as a class
  - Quizzes to keep in check with previous classes' concepts
- Changes
  - More real-life examples instead of just formulas
  - Some quiz questions could be clearer

Lecture overview: Transformers part 1

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

Lecture overview: Transformers part 1

- Self-attention
- Multi-headed attention
- Transformer blocks
- Activity: work through self-attention
- From Vaswani et al. 2017, “Attention is all you need” paper

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

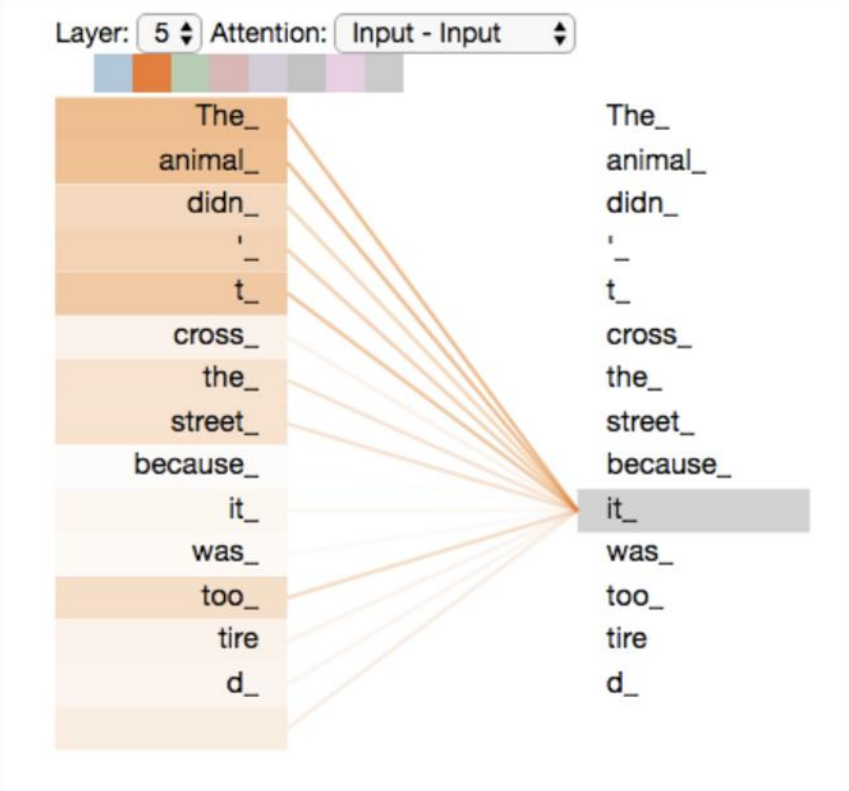
Self-attention

Intuition of self-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

Slide adapted from Jurafsky and Martin

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

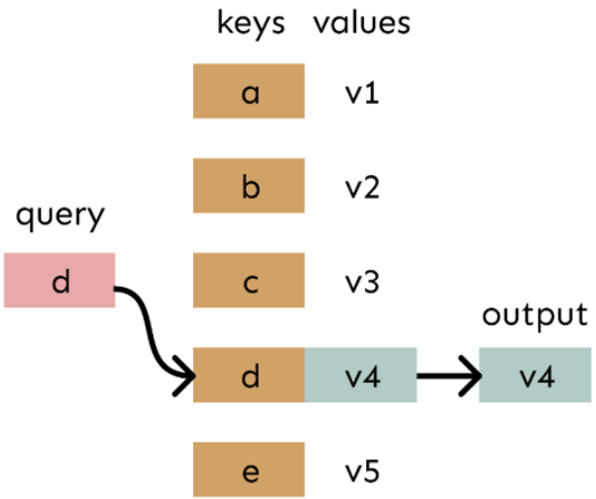
- **query:** As *the current element* being compared to the other inputs.
- **key:** as *an input* that is being compared to the current element to determine a similarity
- **value:** a value of a preceding element that gets weighted and summed

Slide adapted from Jurafsky and Martinc

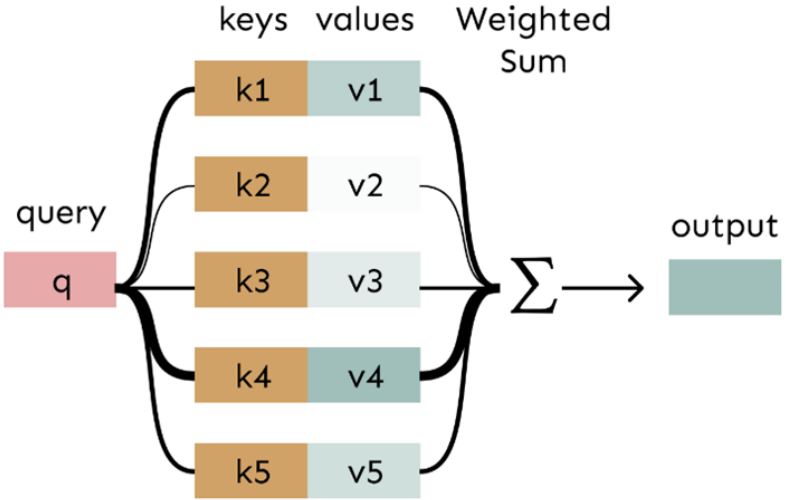
Attention as a soft, averaging lookup table

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 $\mathbf{x}_i$ 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

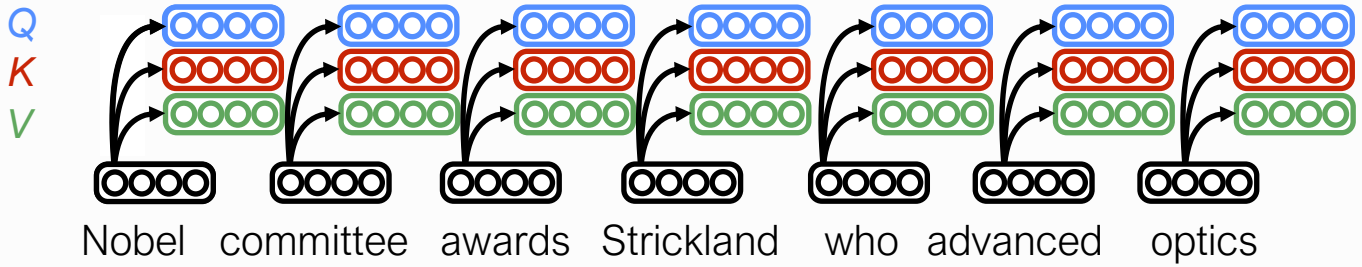
- Given these 3 representation of $\mathbf{x}_i$

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

- To compute the similarity of current element $\mathbf{x}_i$ with some element (for self-attention) $\mathbf{x}_j$
- We'll use dot product between $\mathbf{q}_i$ and $\mathbf{k}_j$.
- And instead of summing up $\mathbf{x}_j$, we'll sum up $\mathbf{v}_j$

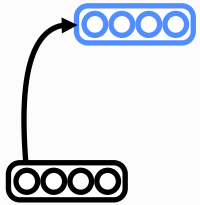
Slide adapted from Jurafsky and Martinc

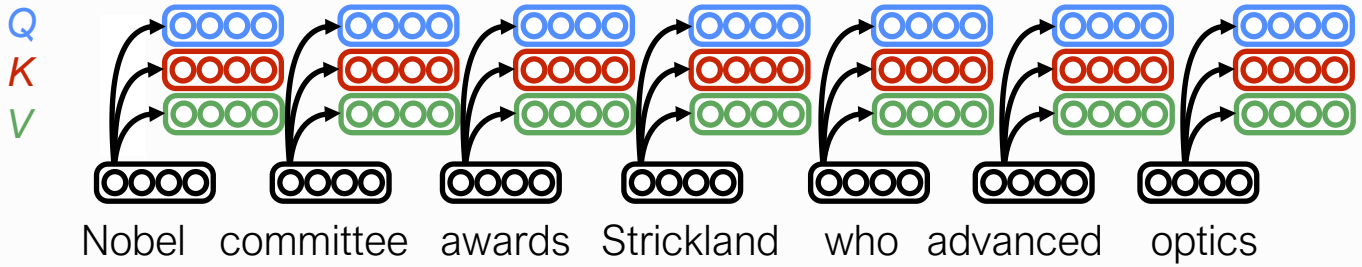
Transformer self-attention



Source: Emma Strubell

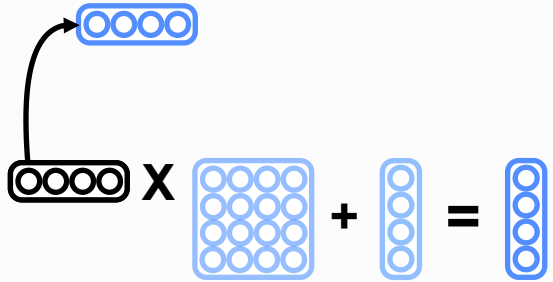
Transformer self-attention

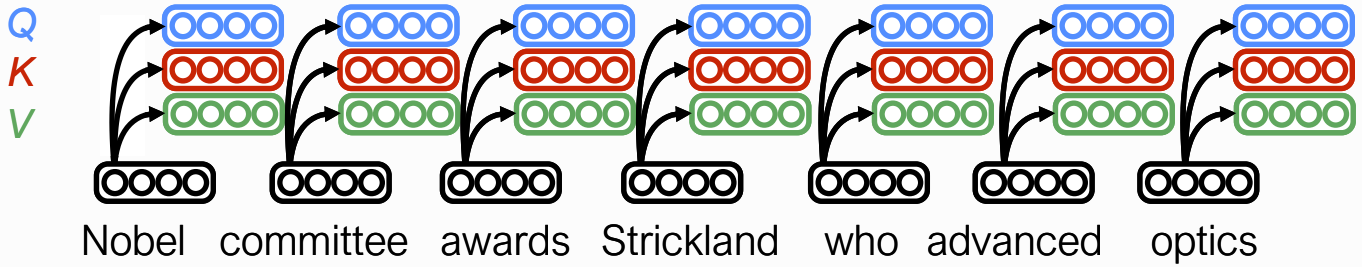




Source: Emma Strubell

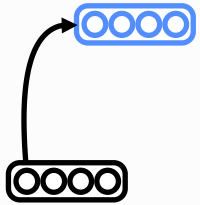
Transformer self-attention

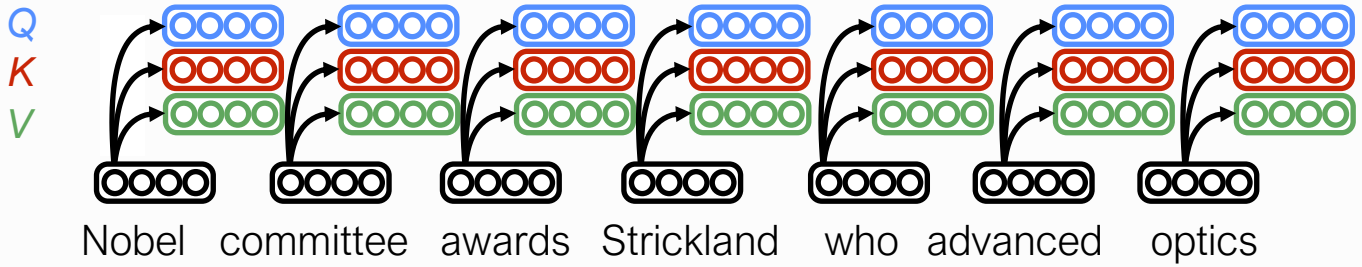




Source: Emma Strubell

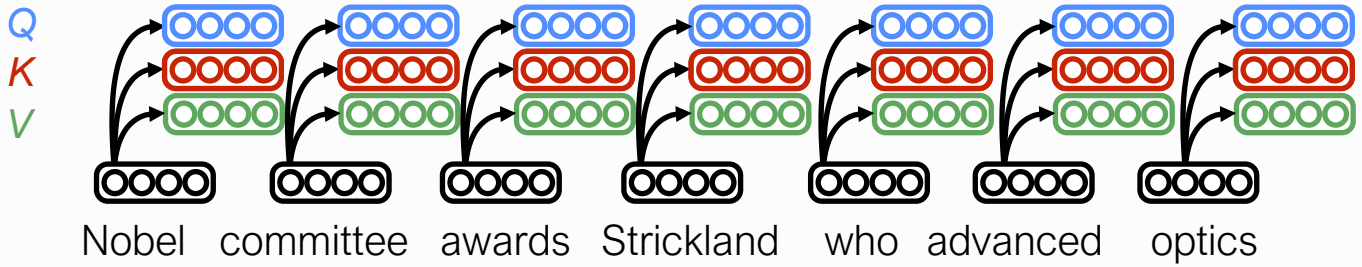
Transformer self-attention





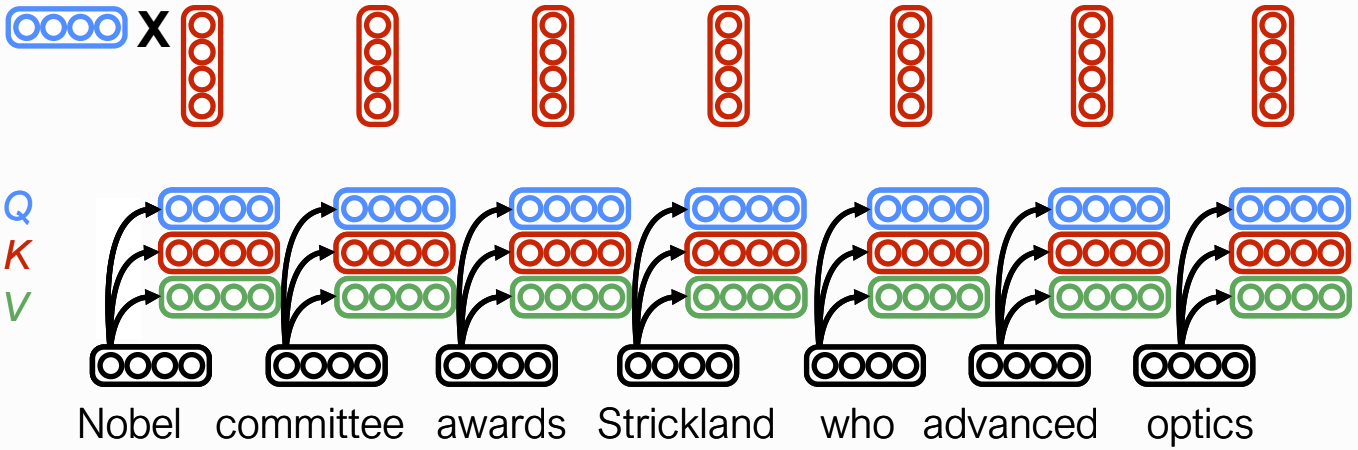
Source: Emma Strubell

Transformer self-attention



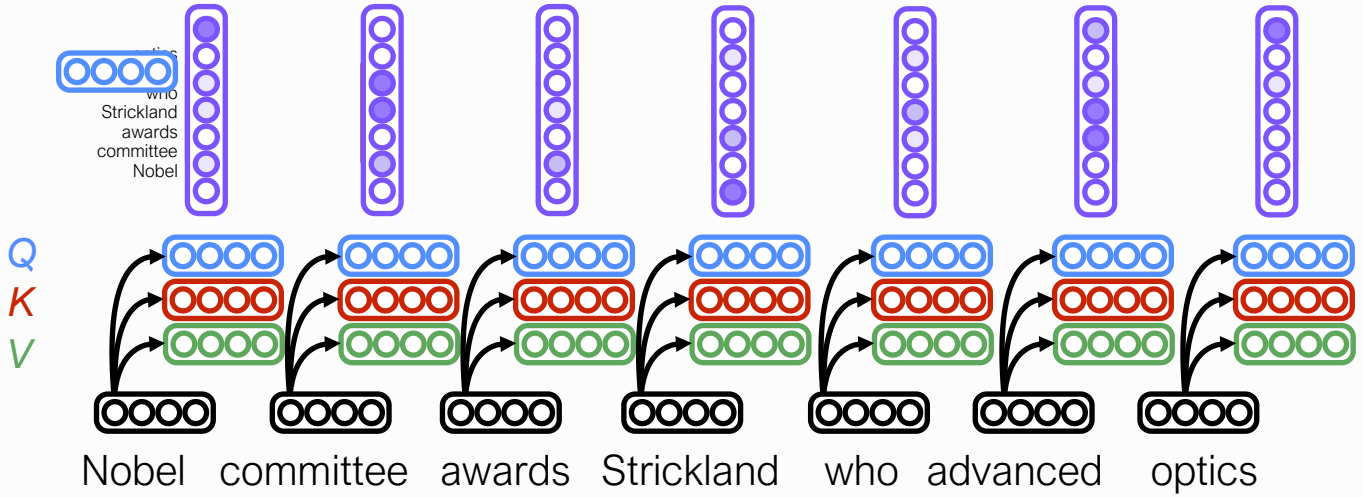
Source: Emma Strubell

Transformer self-attention



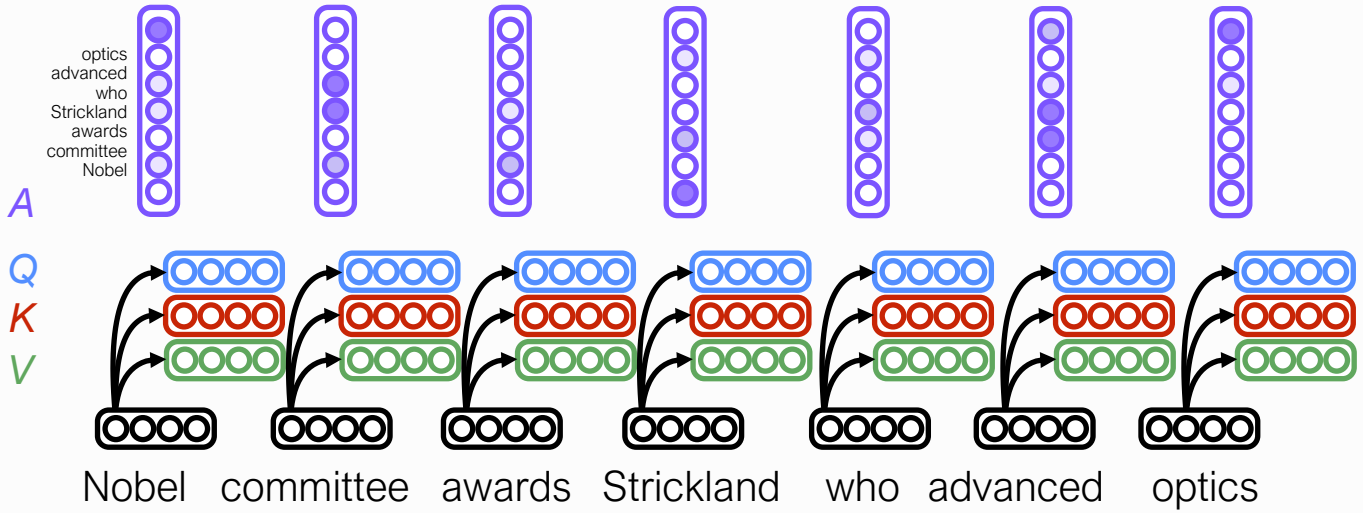
Source: Emma Strubell

Transformer self-attention



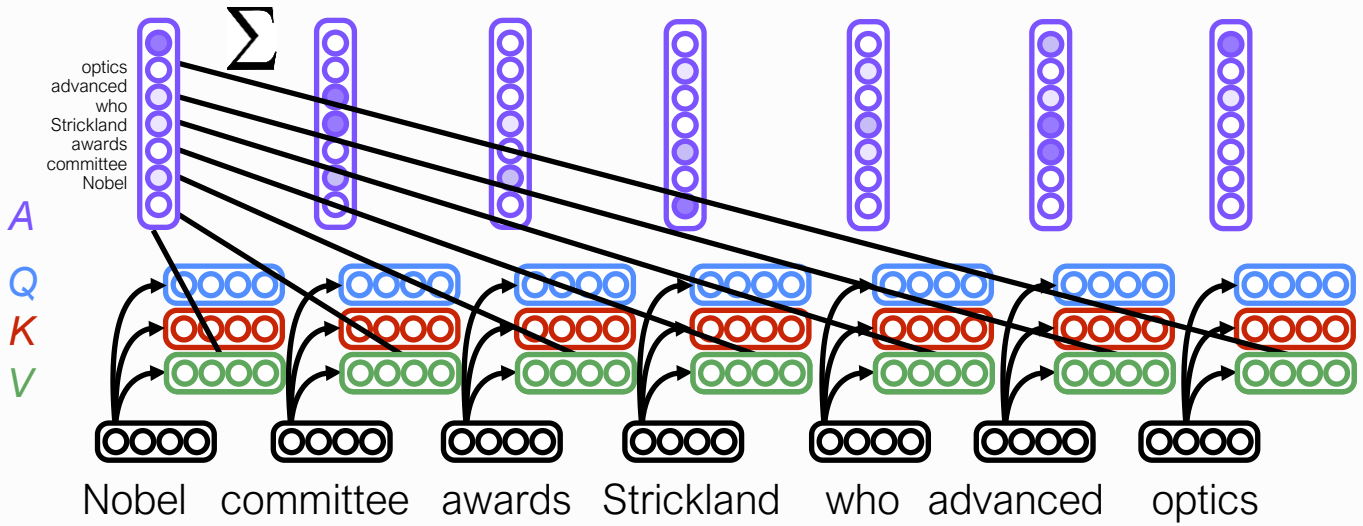
Source: Emma Strubell

Transformer self-attention



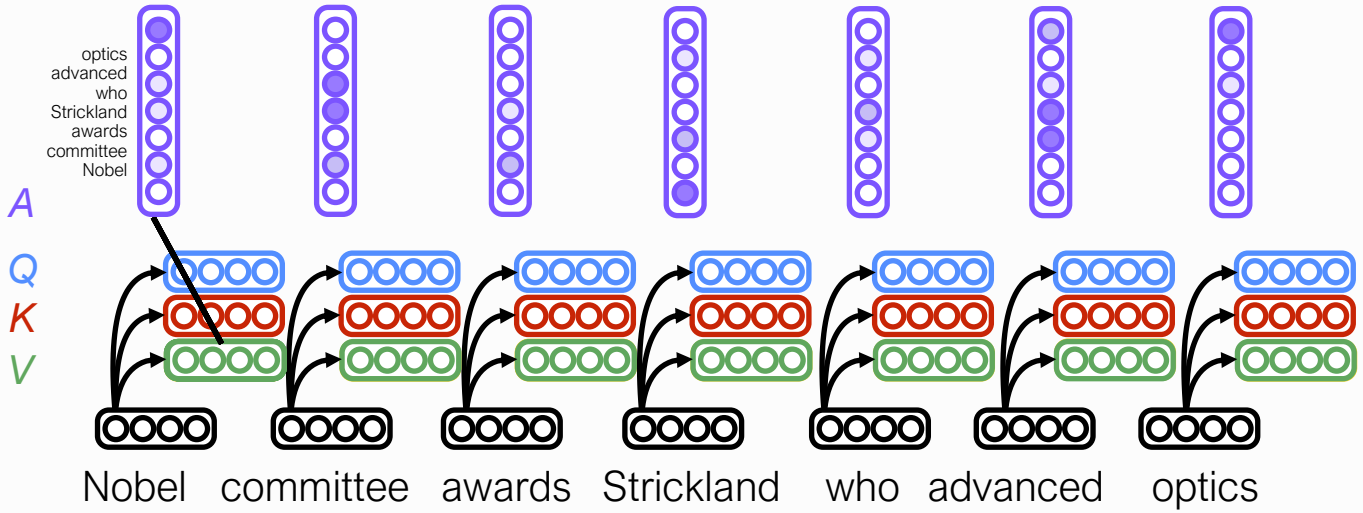
Source: Emma Strubell

Transformer self-attention



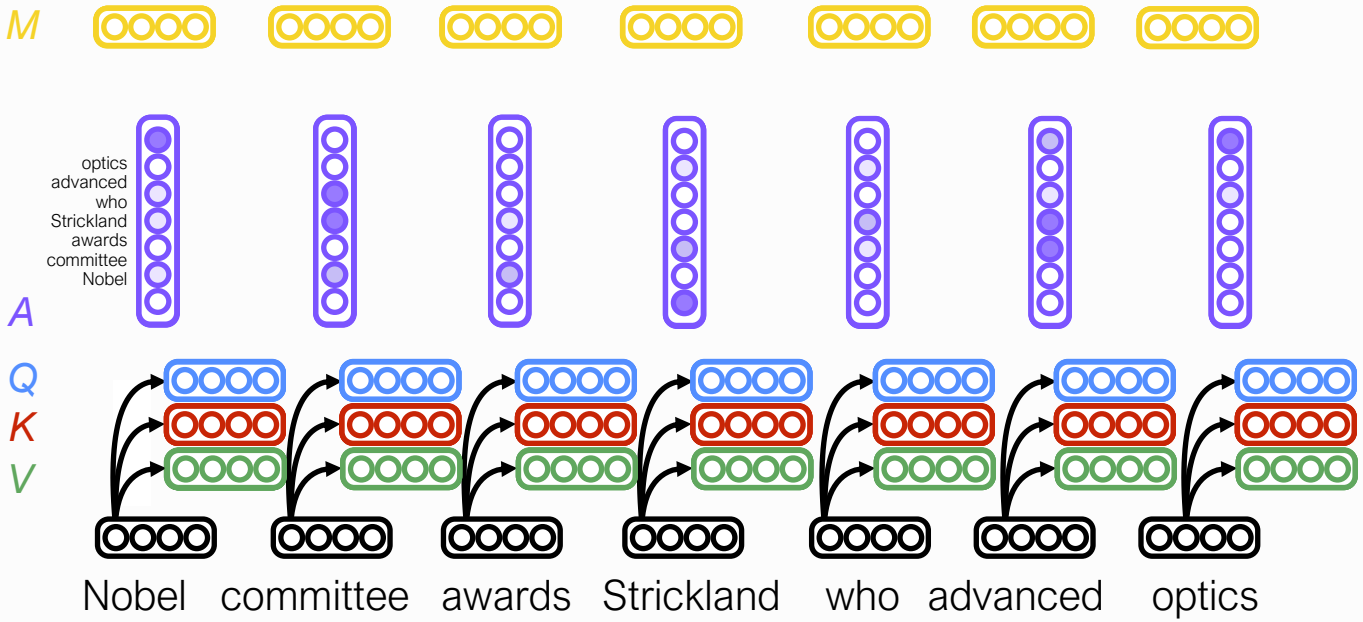
Source: Emma Strubell

Transformer self-attention



Source: Emma Strubell

Transformer self-attention

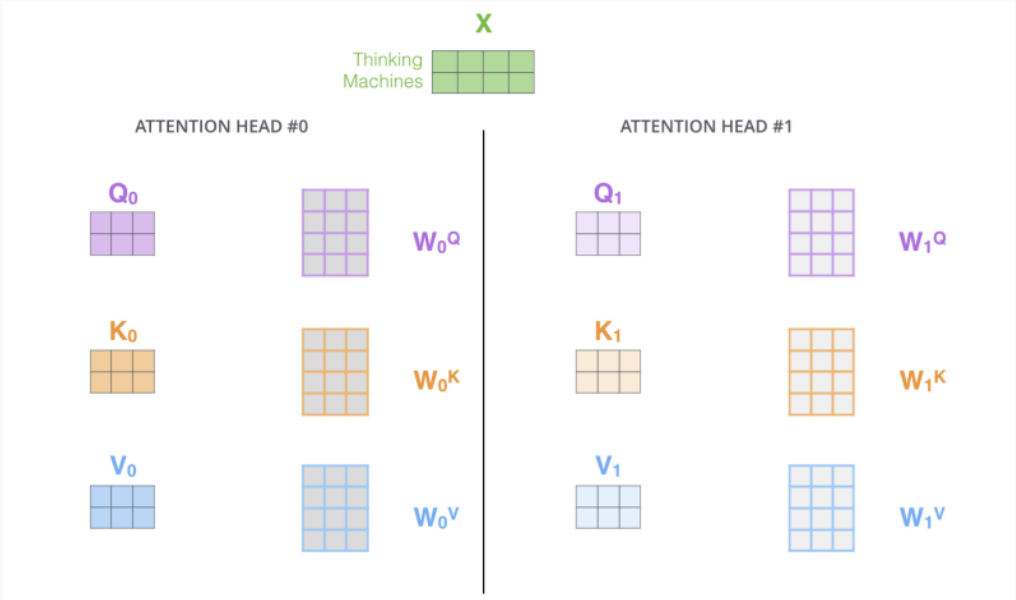


Source: Emma Strubell

Multi-headed attention

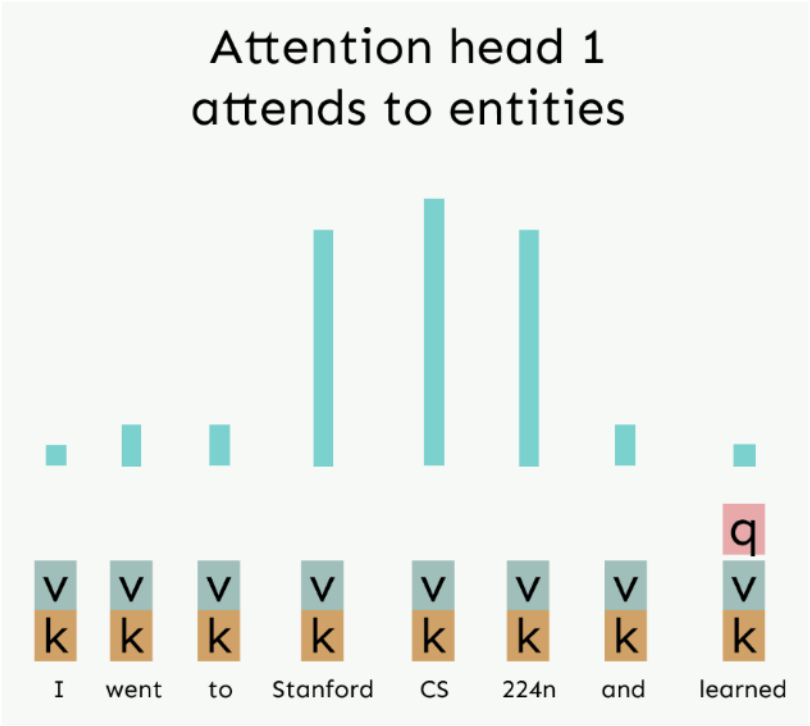
Multi-Headed Attention Expands Transformer Models' Ability to Focus on Different Positions

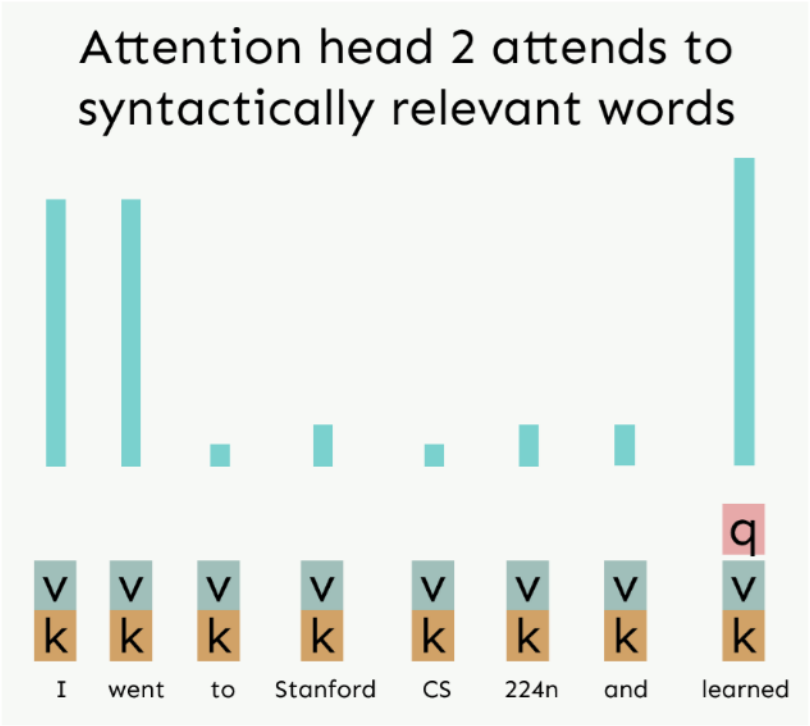
Maintain distinct weight matrices for each attention head—distinct representational subspaces:



Slide credit: David Mortensen

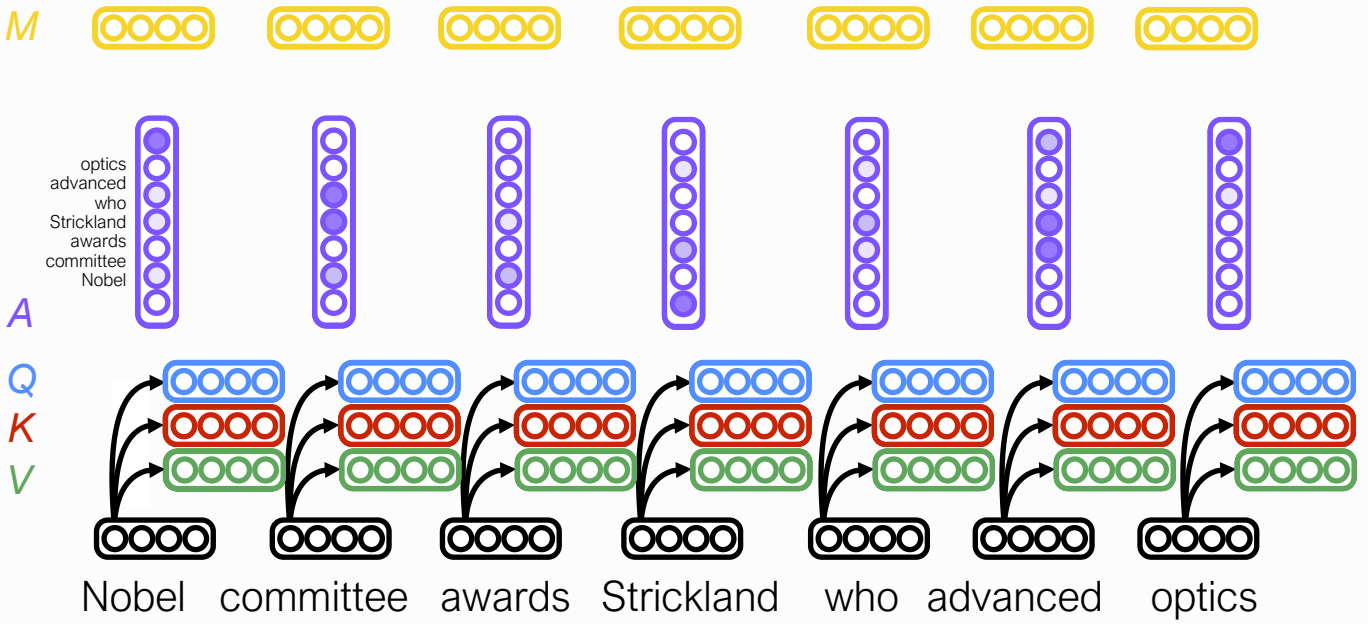
Hypothetical example of multi-headed attention





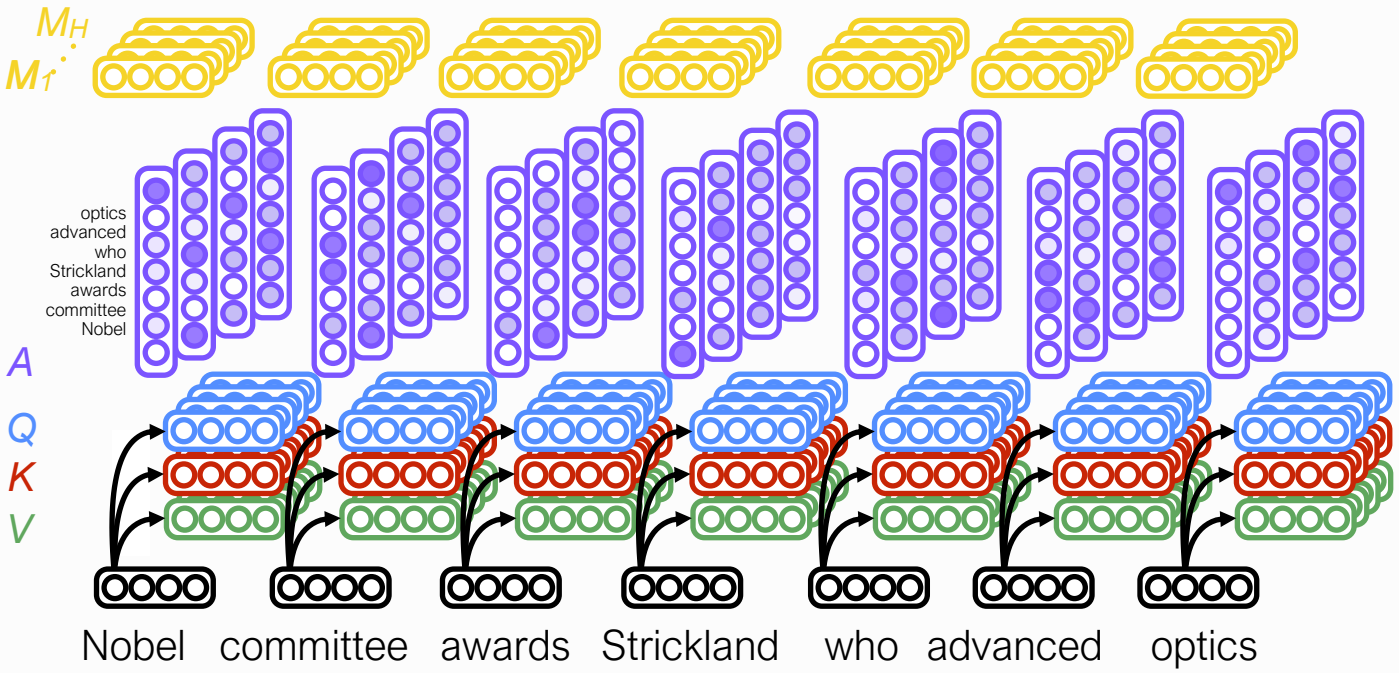
Slide credit: John Hewitt

Transformer self-attention



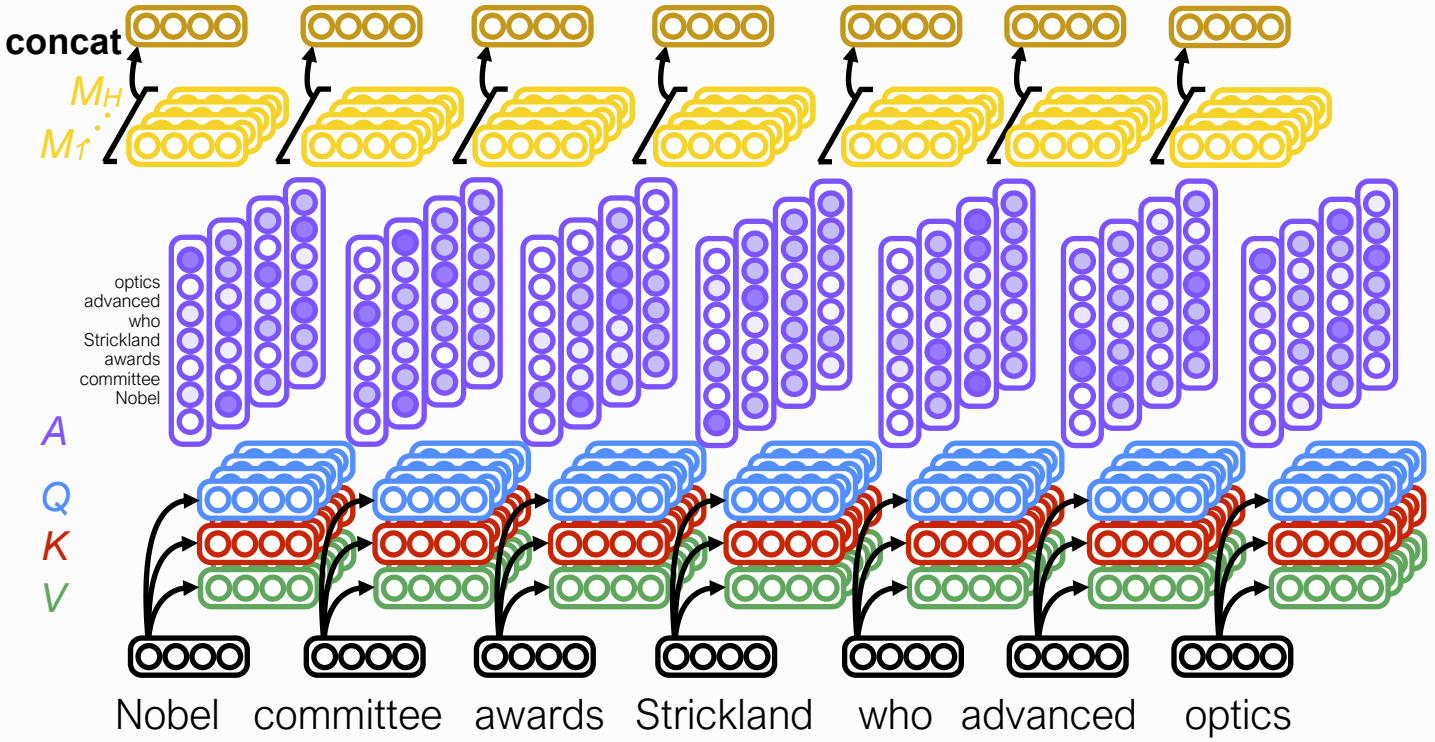
Source: Emma Strubell

Multi-head self-attention



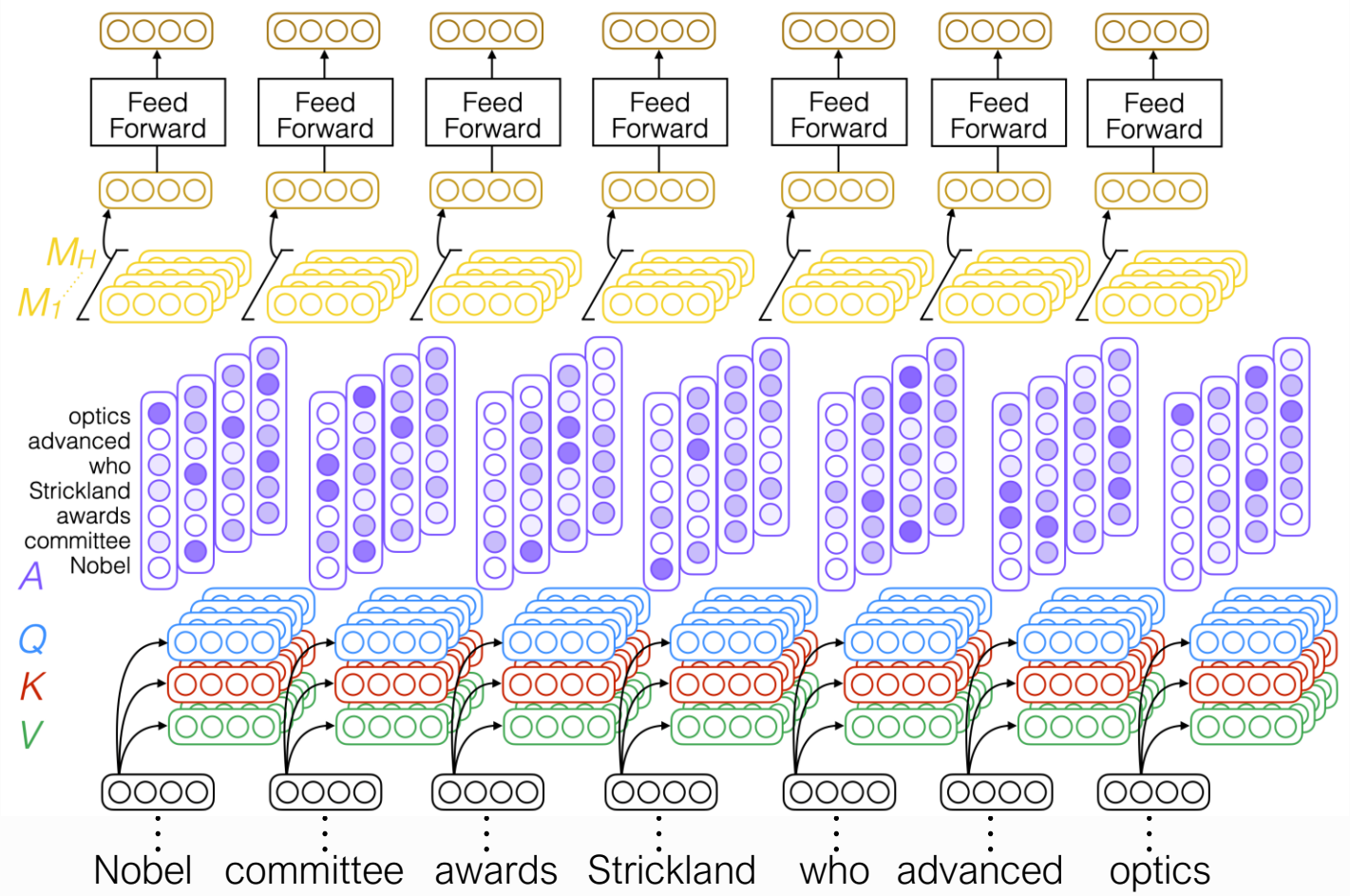
Source: Emma Strubell

Multi-head self-attention



Source: Emma Strubell

Add a feedforward neural transformation for nonlinearity



Source: Emma Strubell

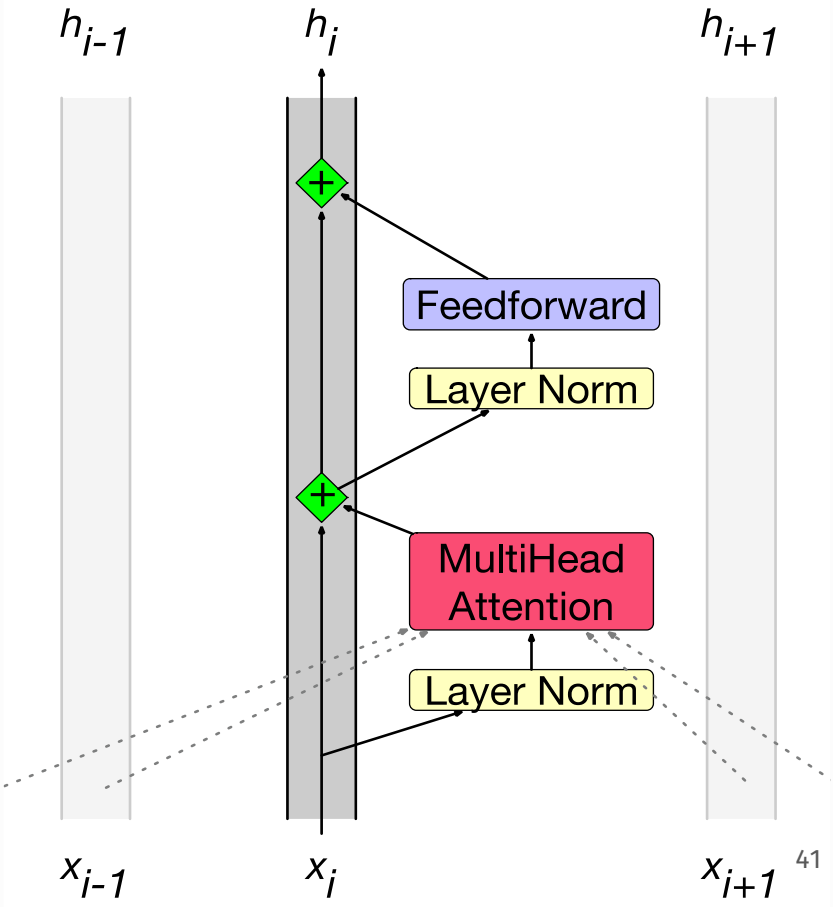
Transformer blocks

Transformer blocks

Each block consists of:

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

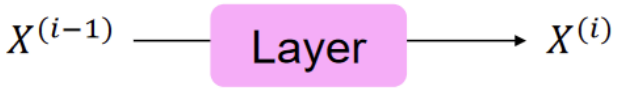
Output: 1 vector for every input token



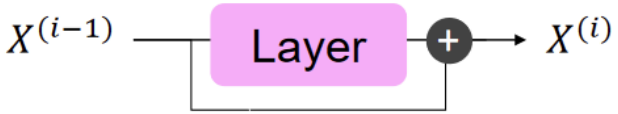
Slide adapted from Jurafsky and Martin, John Hewitt

Residual connections [He et al. 2016]

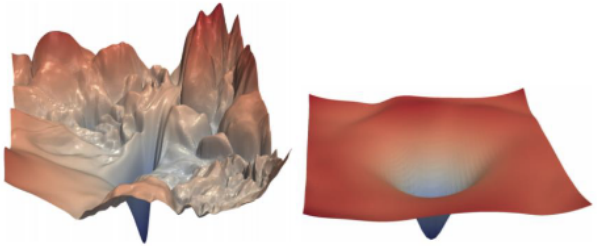
- **Residual connections** are a trick to help models train better.
  - Instead of $X^{(i)} = \text{Layer}(X^{(i-1)})$ (where i represents the layer)



  - We let $X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$ (so we only have to learn "the residual" from the previous layer)



  - Gradient is **great** through the residual connection; it's 1!
  - Bias towards the identity function!



[no residuals] [residuals]

[Loss landscape visualization, Li et al., 2018, on a ResNet]

Slide credit: John Hewitt

Layer normalization [Ba et al. 2016]

- **Layer normalization** is a trick to help models train faster.
- Idea: cut down on uninformative variation in hidden vector values by normalizing to unit mean and standard deviation **within each layer**.
  - LayerNorm's success may be due to its normalizing gradients [Xu et al., 2019]
- Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.
- Let $\mu = \sum_{j=1}^{d} x_j$; this is the mean; $\mu \in \mathbb{R}$.
- Let $\sigma = \sqrt{\frac{1}{d} \sum_{j=1}^{d} (x_j - \mu)^2}$; this is the standard deviation; $\sigma \in \mathbb{R}$.
- Let $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)
- Then layer normalization computes:

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma + \epsilon}} * \gamma + \beta$$

Normalize by scalar mean and variance

Modulate by learned elementwise gain and bias

Slide credit: John Hewitt

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?