Transformer Language Models

Qijia He, PhD student
Time: 11:00am – 12:00am, June 25, 2026

Slide Credits

1. *Generative AI*, 10-423/10-623, by Pat Virtue and Matt Gormley, Carnegie Mellon University: https://www.cs.cmu.edu/~mgormley/courses/10423/

2. *Speech and Language Processing*, by Dan Jurafsky and James H. Martin, Stanford University: https://web.stanford.edu/~jurafsky/slp3/

Index

- Part I: (Attention) Embedding Mechanism
- Part II: Transformer Language Models

Part I: Embedding Mechanism

Autoregressive model

In each forward process, the model returns $\Pr(y_t|\mathrm{x}, y_{<t})$

How could we encode the prompt and context into the language model?

First, We embed each token into a vector
(e.g. through one-hot encoder or a well-trained embedding model)

One naïve idea:
Let's concatenate those vectors and put them to LLMs!

Neural networks usually supports fixed dimension inputs

Then let's take average of the history embedding (Bag of words, $v_{\text{sentence}} = \frac{1}{T}\sum_{t=1}^{T} e_{w_t}$)

Looks better, but it does not capture the sequence information

Recurrent Neural Networks (RNNs)

Let $w_{t-1}, \dots, w_1$ be the embedded prefix

Key idea:

1) Convert all previous words to a fixed length vector (embedding)
2) Define distribution $p(w_t | f_\theta(w_{t-1}, \dots, w_1))$ that conditions on the vector $h_t = f_\theta(w_{t-1}, \dots, w_1) = \phi\left(W_h h_{t-1} + W_x e_{w_{t-1}} + b\right)$, where ϕ is the activation function, $e_{w_{t-1}}$ is embedding of the new word w_t

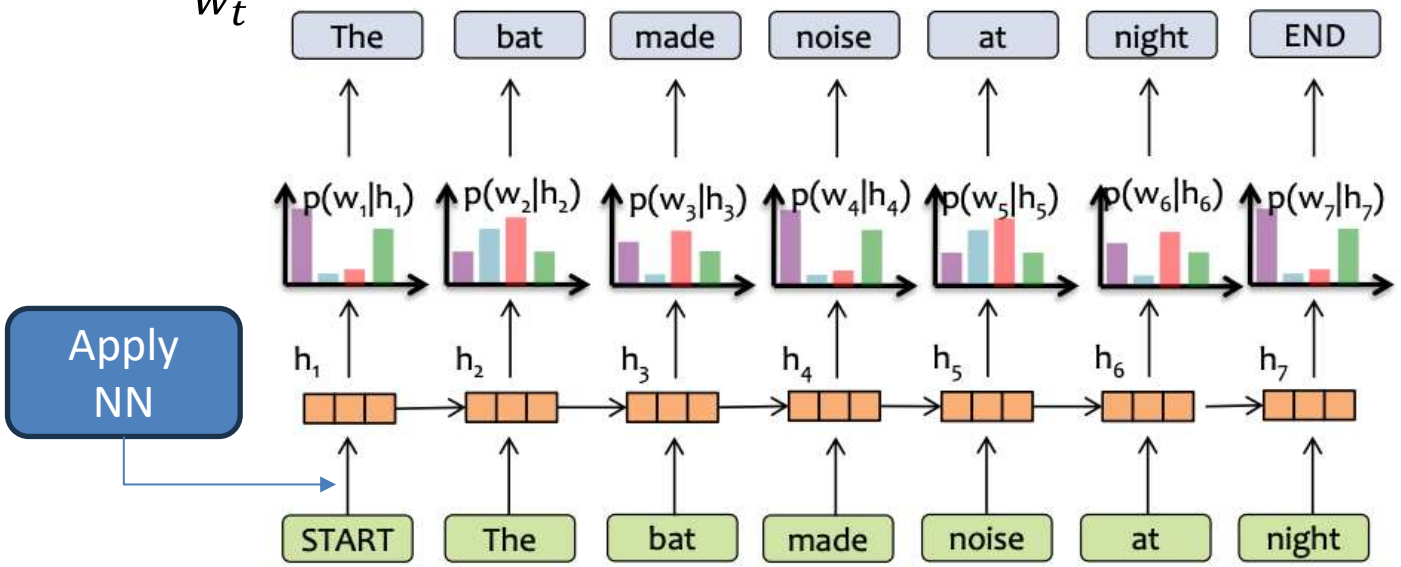


Figure from: Virtue and Gormley (2025)

Problem with Static Embeddings (word2vec)

They are static!

The embedding for a word doesn't reflect how its meaning changes in text.

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

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

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
- The hidden state of RNN is contextual embedding, but does not capture structural information for a word (like what “it” refers to)
- How to compute contextual embeddings that could capture those structural information?
    - **Attention**

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**

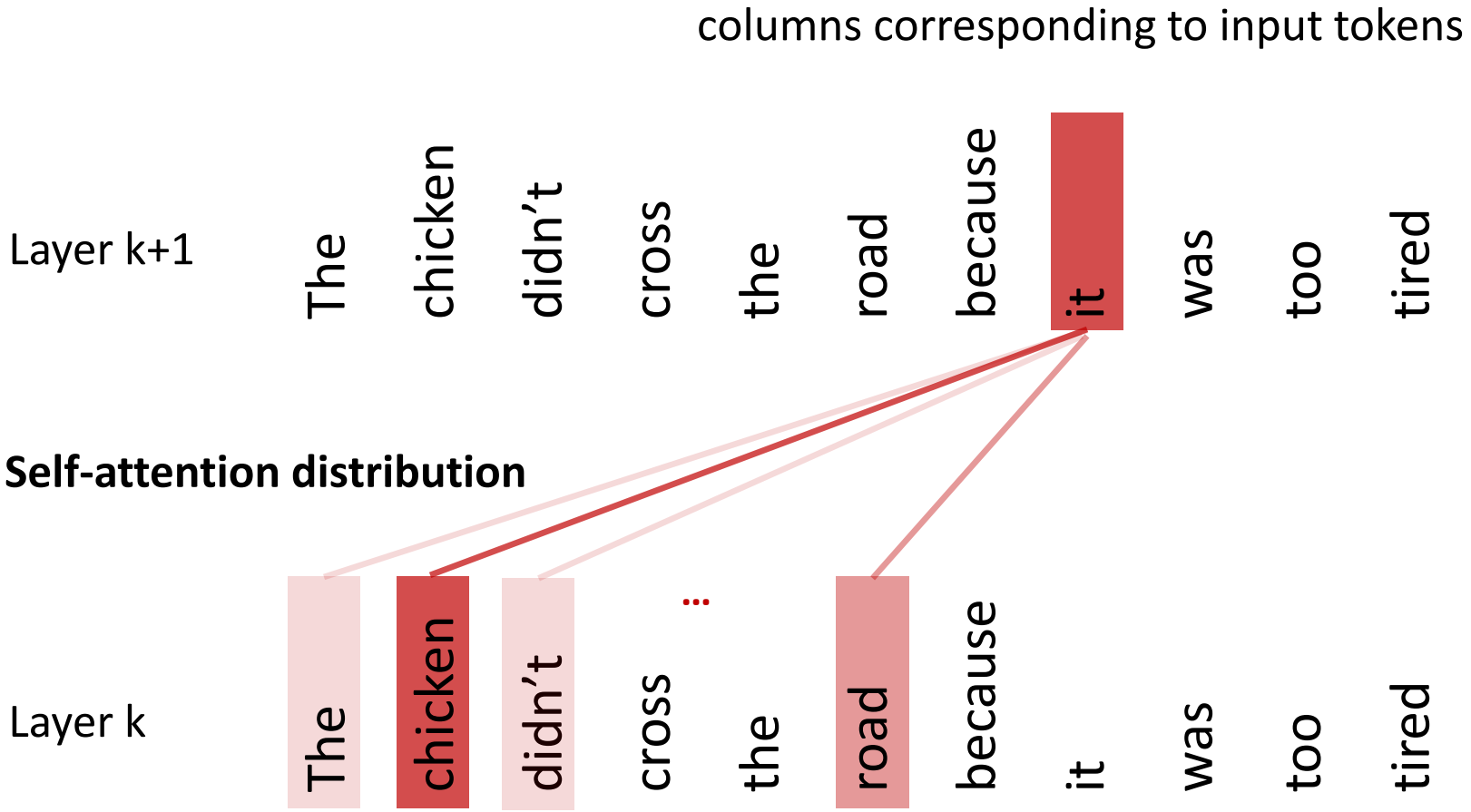
At this point in the sentence, it's probably referring to either the chicken or the road

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

Intuition of Attention



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.

Simplified Version of Attention

- Given a sequence of token embeddings:
  - $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_i$
- Produce: a_i =a weighted sum of x_1 through x_7 (and x_i)
- Weighted by their **similarity to x_i**

$$\text{score}(x_i, x_j) = x_i \cdot x_j$$

$$\alpha_{ij} = \text{softmax}\left(\text{score}(x_i, x_j)\right), \ \forall j \le i$$

$$\boldsymbol{a_i} = \sum_{j \le i} \alpha_{ij} \, x_j$$

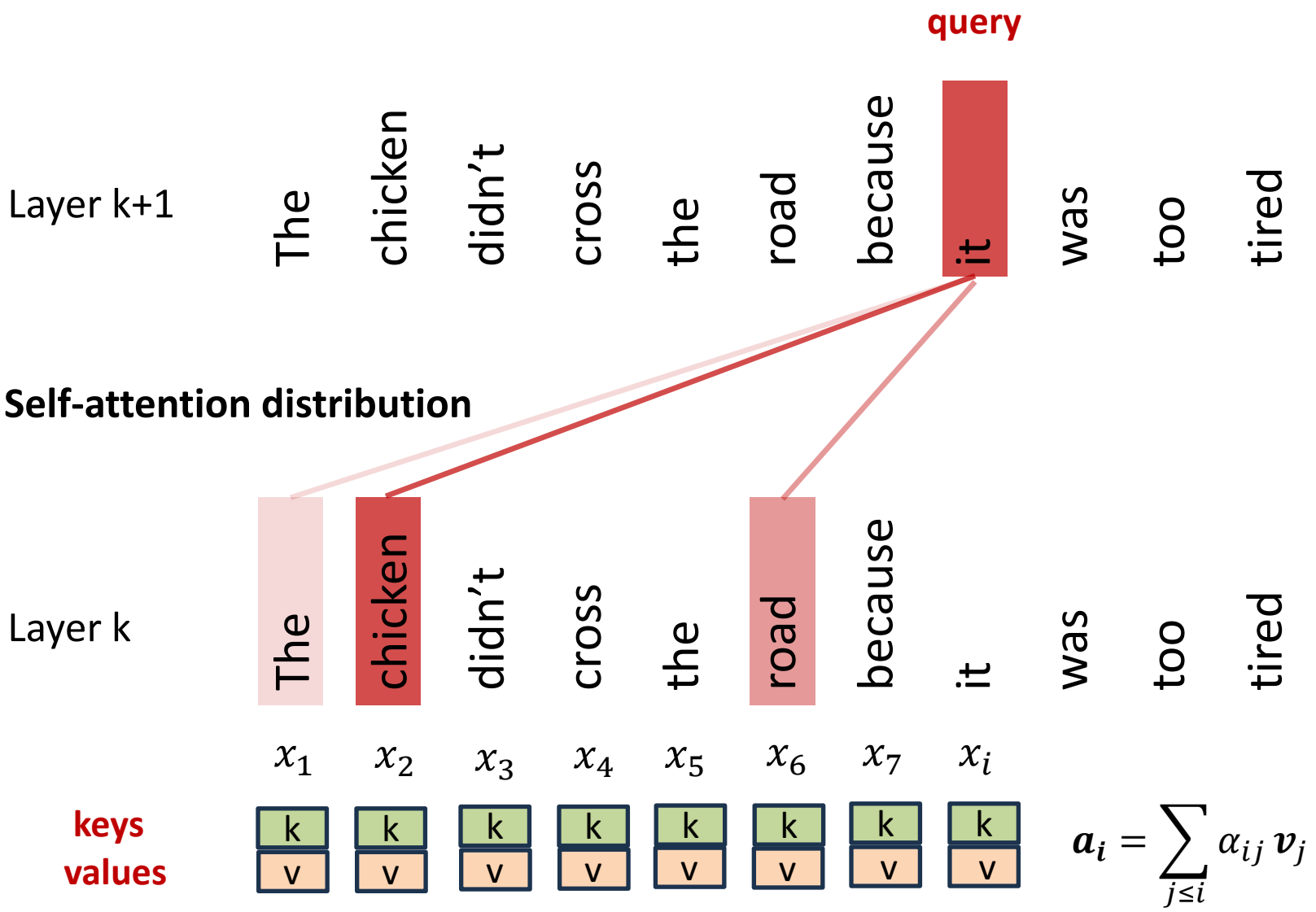
Given a vector $\mathbf{z} = [z_1, z_2, \ldots, z_n]$, the softmax is: $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$.

This is applied **element-wise** to produce: $\text{softmax}(\boldsymbol{z}) = \left[\frac{e^{z_1}}{\sum_j e^{z_j}}, \frac{e^{z_2}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_n}}{\sum_j e^{z_j}}\right]$

An Actual Attention Head

- High-level idea: The "simplified Attention mechanism" looks great, but "Similarity" doesn't mean the word is important for the token's context. (e.g. "this" is close to "those", but I don't want to focus on "those" when inferring the word "this"). By induction, it also means that sometimes in a sentence, "A" wants to focus on "B", but "B" doesn't want to focus on "A" To capture this phenomenon, we need at least 3 features for one token.
- Instead of using vectors (like x_i and x_4) directly, we'll represent 3 separate roles each vector x_i players:
  - **Query:** As the current element being compared to the preceding inputs.
  - **Key:** As a preceding 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.

An Actual Attention Head



An Actual Attention Head

- We will use matrices to project each vector x_i into a representation of its role as query, key, value:
  - **Query**: $\boldsymbol{W}^Q$
  - **Key**: $\boldsymbol{W}^K$
  - **Value**: $\boldsymbol{W}^V$

$$\boldsymbol{q}_i = x_i \boldsymbol{W}^Q; \qquad \boldsymbol{k}_i = x_i \boldsymbol{W}^K; \qquad \boldsymbol{v}_i = x_i \boldsymbol{W}^V$$

An Actual Attention Head

Given these 3 representation of x_i

$$\boldsymbol{q}_i = x_i \boldsymbol{W}^Q; \qquad \boldsymbol{k}_i = x_i \boldsymbol{W}^K; \qquad \boldsymbol{v}_i = x_i \boldsymbol{W}^V$$

To compute similarity of current element x_i with some prior element x_j, we will use dot product between q_i and k_j.

Instead of summing up x_j, we will sum up v_j.

Q: Why?

A: Because we don't want to directly mix the **raw inputs x_j**. We want to mix **task-specific transformed representations**— that's what the **values v_j** are.

An Actual Attention Head: Final Equations

$$\boldsymbol{q}_i = x_i \boldsymbol{W}^Q; \qquad \boldsymbol{k}_j = x_j \boldsymbol{W}^K; \qquad \boldsymbol{v}_j = x_j \boldsymbol{W}^V$$

$$\text{score}(x_i, x_j) = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}\left(\text{score}(x_i, x_j)\right), \qquad \forall j \le i$$

$$\boldsymbol{a_i} = \sum_{j \le i} \alpha_{ij} \boldsymbol{v}_j$$

Calculate the Value of a_3

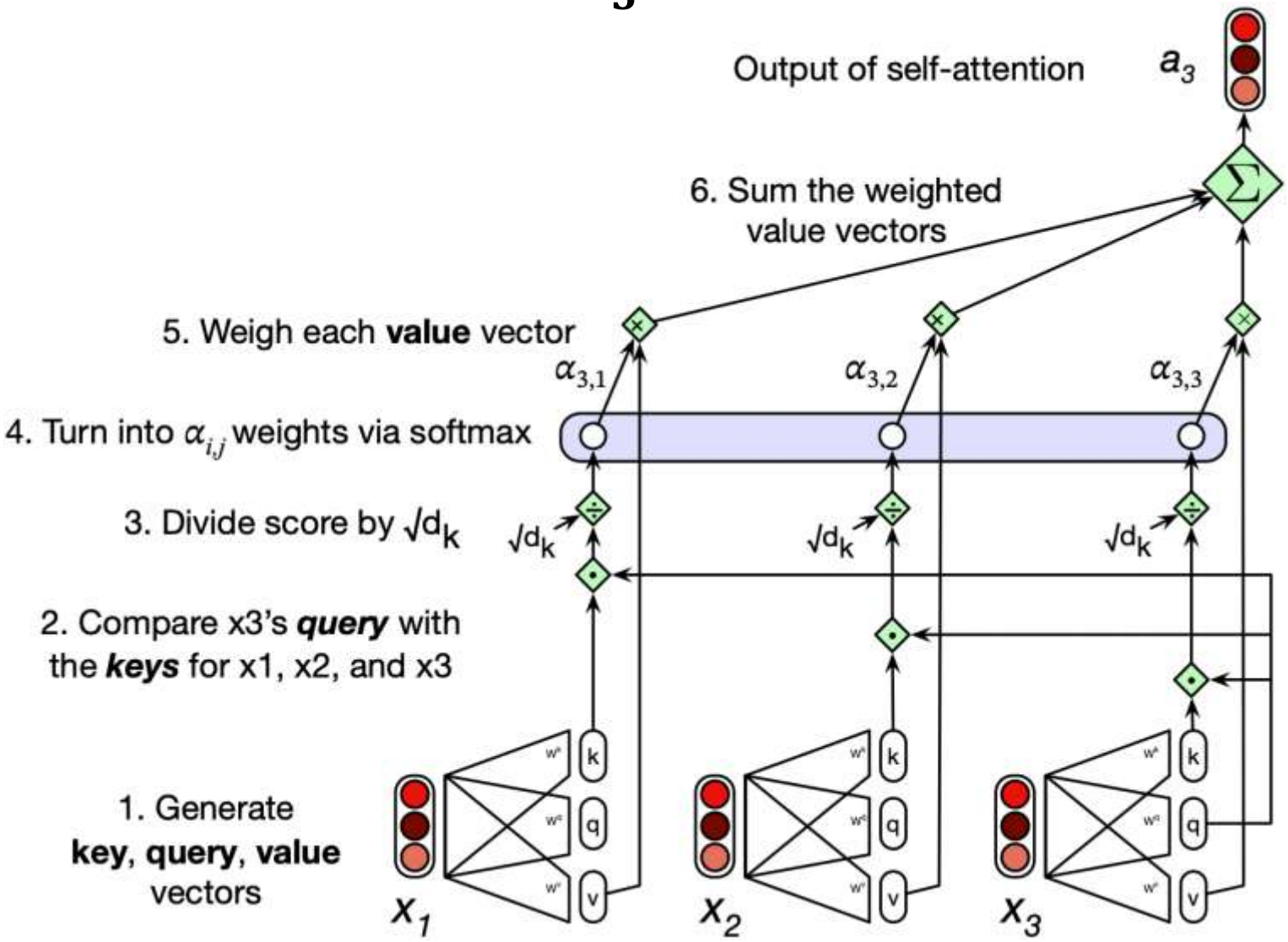


Figure from: Jurafsky and Martin (2024)

Multi-Head Attention

- Instead of one attention head, we'll have lots of them!
- Intuition: each head might be attending to the context for different purposes
  - Different linguistic relationships or patterns in the context

$$\boldsymbol{q}_i^c = x_i \boldsymbol{W}^{\boldsymbol{Q}_c}; \quad \boldsymbol{k}_j^c = x_j \boldsymbol{W}^{\boldsymbol{K}_c}; \quad \boldsymbol{v}_j^c = x_j \boldsymbol{W}^{\boldsymbol{V}_c}; \ \forall c \in \{1, \ldots, h\}$$

$$\text{score}^c(x_i, x_j) = \frac{\boldsymbol{q}_i^c \cdot \boldsymbol{k}_j^c}{\sqrt{d_k}}$$

$$\alpha_{ij}^c = \text{softmax}\left(\text{score}^c(x_i, x_j)\right), \qquad \forall j \le i$$

$$\mathbf{head}_i^c = \sum_{j \le i} \alpha_{ij}^c \boldsymbol{v}_j^c$$

$$\boldsymbol{a_i} = (\mathbf{head}^1 \oplus \mathbf{head}^2 \oplus \ldots \oplus \mathbf{head}^h)\boldsymbol{W}^{\boldsymbol{O}}$$

$$\text{MultiHeadAttention}(x_j, [x_1, \ldots, x_N]) = \boldsymbol{a_i}$$

Part II: Transformer Language Models

Transformer Language Model

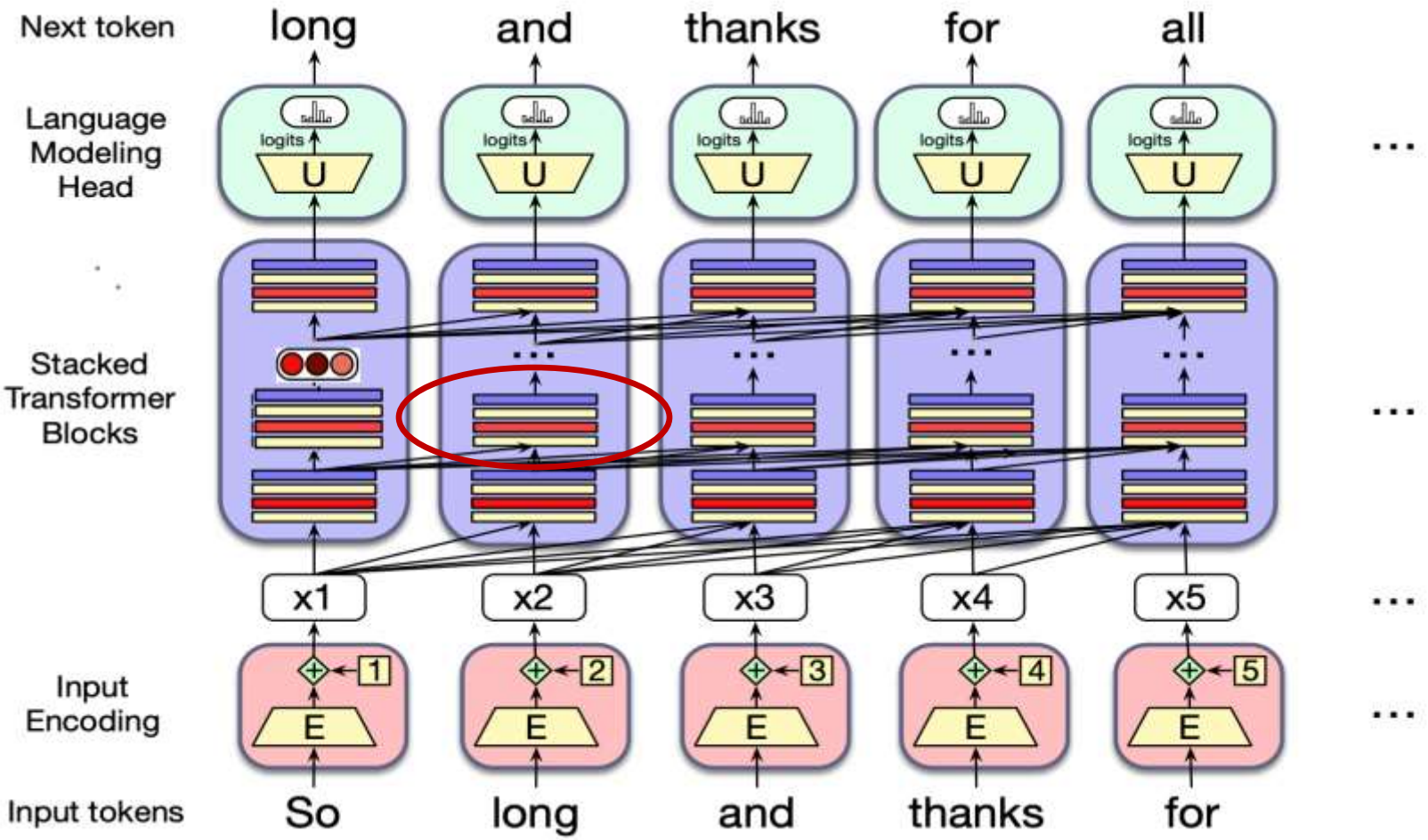


Figure from: Jurafsky and Martin (2024)

Transformer Language Model

- The residual stream: each token gets passed up and modified

Q: Why is residual stream important?

...

1. It prevents **vanishing gradients** during training (deep networks).
2. It lets each layer **refine** the representation rather than overwrite it.

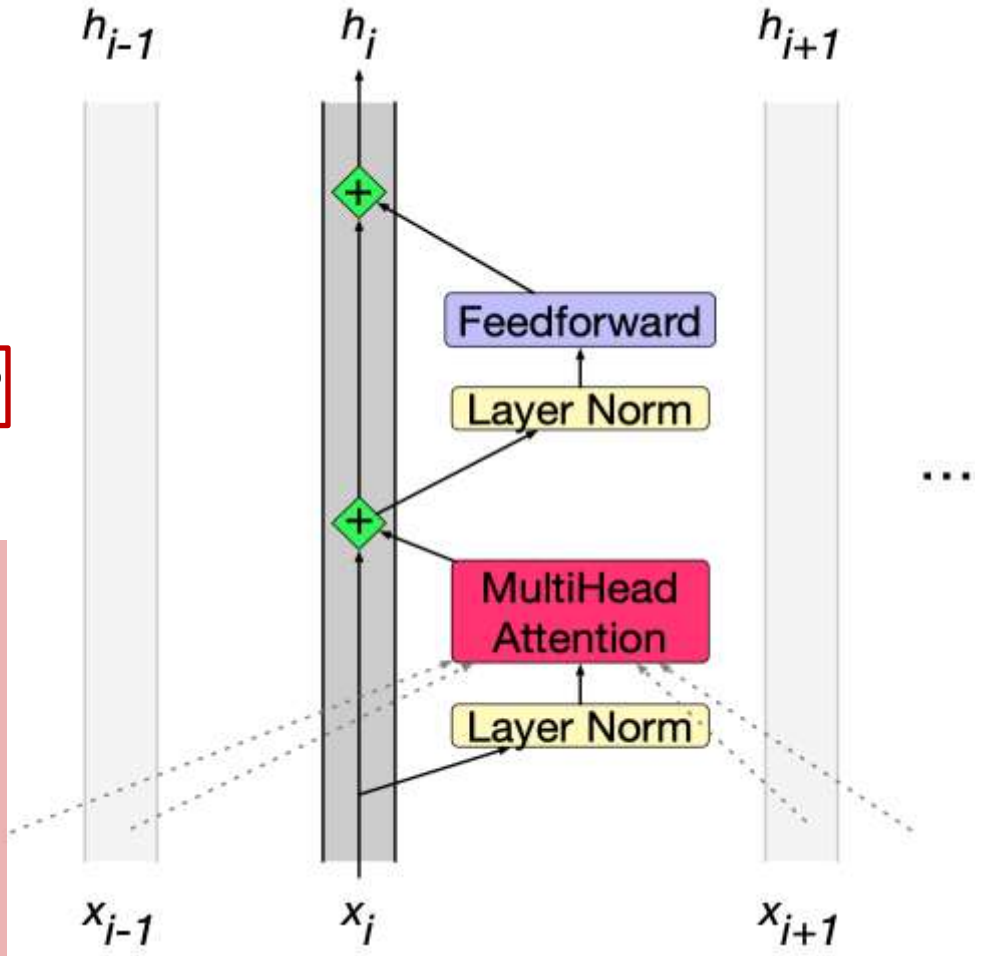


Figure from: Jurafsky and Martin (2024)

Transformer Language Model

- The residual stream: each token gets passed up and modified
- We'll need non-linearities, so a feedforward layer:

$$\text{FFN}(x_i) = \text{ReLU}(x_i \boldsymbol{W}_1 + b_1)\boldsymbol{W}_2 + b_2$$

- Layer norm: the vector x_i is normalized twice

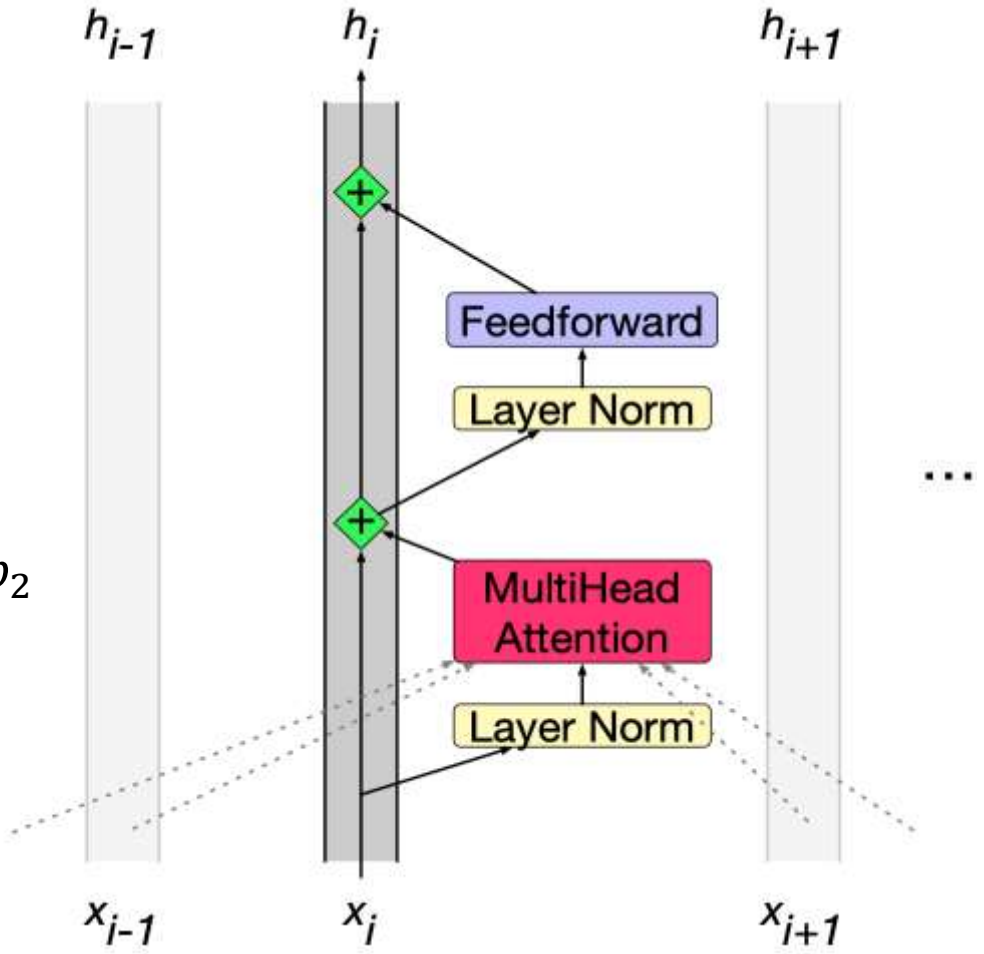


Figure from: Jurafsky and Martin (2024)

Layer Norm

Layer norm is a variation of the z-score from statistics, applied to a single vector in a hidden layer

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$$ mean of the features

$$\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \boldsymbol{\mu})^2}$$ standard deviation

$$\hat{\boldsymbol{x}} = \frac{(\boldsymbol{x} - \boldsymbol{\mu})}{\sigma}$$ z-score (standardization)

$$\text{LayerNorm}(\boldsymbol{x}) = \gamma\frac{(\boldsymbol{x}-\boldsymbol{\mu})}{\sigma} + \beta$$

learnable scale and shift parameters

Q: What's the function of Layer norm?

1. Stabilizes training: it prevents exploding or vanishing gradients.
2. Makes the optimization landscape smoother by keeping layer inputs at a consistent scale.
3. Allows deeper networks (like Transformers) to train faster and more reliably.

Putting Together

$$t_i^1 = \text{LayerNorm}(x_i)$$

$$t_i^2 = \text{MultiHeadAttention}(t_i^1, [x_1^1, \ldots, x_N^1])$$

$$t_i^3 = t_i^2 + x_i$$

$$t_i^4 = \text{LayerNorm}(t_i^3)$$

$$t_i^5 = \text{FFN}(t_i^4)$$

$$h_i = t_i^5 + t_i^3$$

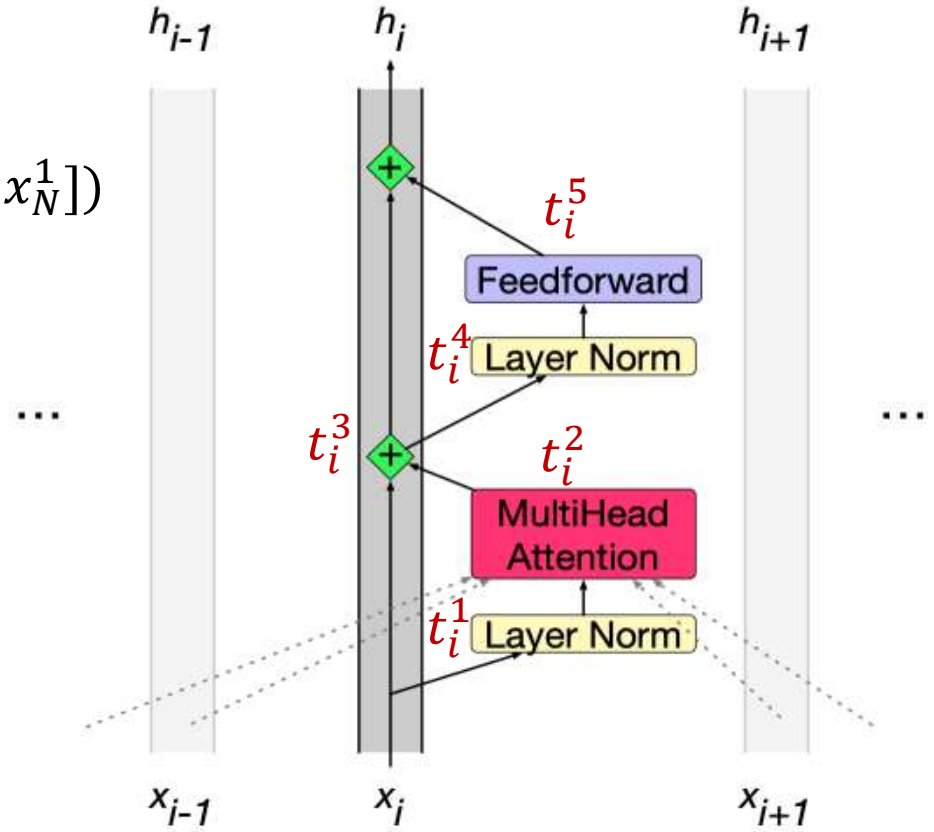


Figure from: Jurafsky and Martin (2024)

Reminder: Next class will be Jun 30 1 pm online