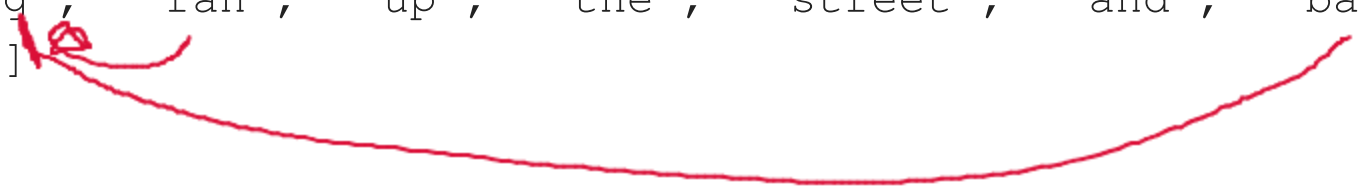
CSCI 1010 Class 12

Profs. Michael Linderman and Phil Chodrow
Department of Computer Science
Middlebury College

Consider the sentence “The dog ran up the street and barked loudly.”, which might be tokenized as follows. What relationships between tokens would be important to capture for a model to generate coherent “next” tokens?

```
['The', ' dog', ' ran', ' up', ' the', ' street', ' and', ' bark', 'ed', ' loudly', '.']
```



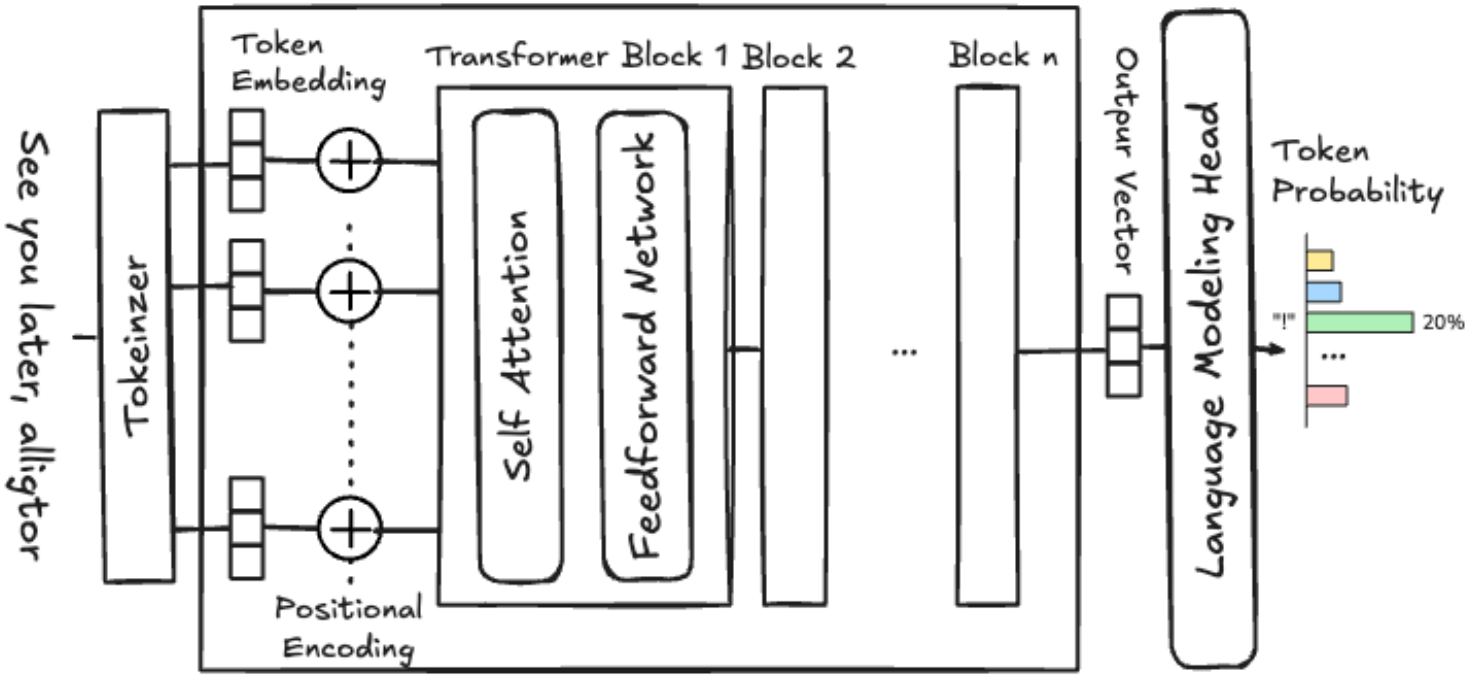
Slide 1 Notes

What is the subject of those two verbs? The dog. What about tense and number? Past tense and singular. Presumably whatever text is generated next should be about a dog, similarly past tense, and agree in number with "dog".

Would a "2nd order" Markov model, i.e., looking at the two previous words, be able to capture that information? Probably not… Although in fairness the output for GPT-2 small is not necessarily great either.

What we observe then is the need capture, potentially "long-range", dependencies between tokens, and do so in a context-dependent way. That latter implies we need some to learn what dependencies are relevant. This is our focus for today.

(Decoder-only) Transformer Architecture



Slide 2 Notes

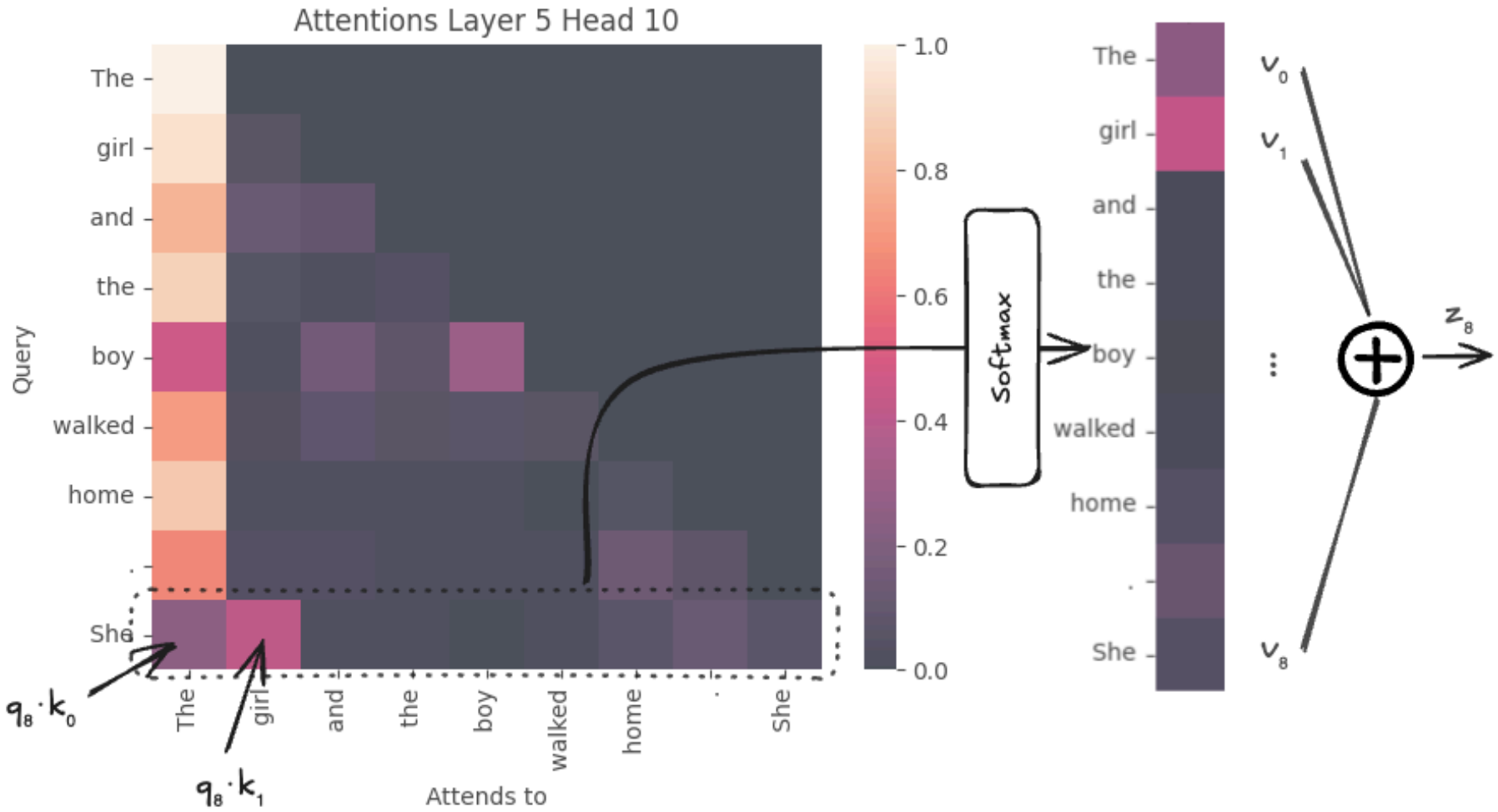
Notice that tokens are processed in parallel, i.e., the entire input sequence is processed at once, rather than sequentially as in RNNs. This is a key feature of Transformer architectures that enables efficient training and inference on modern compute hardware. Let's follow the data flow through the model:

1. The input text is tokenized as we discussed previously,
2. then the tokens are converted to embeddings (dense vector representations of size d_{model}), and
3. combined with positional encodings that provide information about the position of each token in the sequence.
4. The resulting representations are then passed through multiple layers of Transformer blocks, each consisting of a multi-head self-attention layer followed by a feed-forward neural network (similar to those we discussed previously).
5. Finally, a linear layer maps the output of the last Transformer block to the vocabulary size to produce logits for predicting the next token.

The positional encoding are required to enable the parallel processing of tokens. The Transformer architecture doesn't inherently encode the order of the tokens (each token undergoes the same transformations) so positional encodings are added to the token embeddings to provide that ordering information. Without positional encodings, the model would treat all tokens as if they were at the same position in the input sequence, i.e., "The dog bit the man" and "The man bit the dog" would be processed identically. Positional encodings can be fixed (e.g., using sinusoidal functions) or learned during training (the case for GPT2).

The self-attention block is where the "mixing" of information across tokens occurs, enabling the model to capture dependencies between tokens regardless of their distance in the sequence.

Tracing the self-attention calculation



Slide 3 Notes

Let's trace through the attention calculation for the token "She" in this attention head. "She" is the query, i.e., q_8. Each element in that corresponding row of the attention weights (row at index 8 in the heatmap) corresponds to the dot product between the query vector for "She" and the key vectors for "She" and all preceding tokens in the sequence (e.g., k_0 for "The", k_1 for "girl", …), scaled and normalized via softmax. The resulting distribution is used to compute a weighted sum of the value vectors to produce z_8, the output representation for "She" from this attention head. Specifically for this head, $z_8 = 0.25v_0 + 0.41v_1 + ...$, i.e., primarily incorporates information from t_0 and t_1, "The" and "girl".

Which of the following best describes the attention weights in self-attention?

a. The frequency of each token in the training data.

b. How important each token is to the overall meaning of the sequence.

c. How similar the semantic meaning of each token is to every other token.

d. How much other tokens are relevant to computing the representation of a given token.

Slide 4 Notes

Answer: D

Recall the attention weights don't represent how important a token is in general. They are the weights used to compute the output for a specific token from the values of all tokens, i.e. "How much other tokens are relevant to computing the representation of a given token." They are not function of token frequency or semantic similarity (although that latter may indirectly influence the weights learned).

In the absence of positional encodings and causal masking, which of the following must be true about the attention weights for the inputs:

1. “The dog bit the man”
2. “The man bit the dog”

a. $\text{Attention}^1_{\text{man}\to\text{dog}} > \text{Attention}^2_{\text{man}\to\text{dog}}$

b. $\text{Attention}^1_{\text{man}\to\text{dog}} = \text{Attention}^2_{\text{man}\to\text{dog}}$

c. $\text{Attention}^1_{\text{man}\to\text{dog}} < \text{Attention}^2_{\text{man}\to\text{dog}}$

Slide 5 Notes

Answer: B

Recall that all tokens are transformed by the same weight matrices, thus in the absence of positional encodings, the Q, K, and V vectors are the same for a given token regardless of its position in the sequence (although the overall order is shuffled). Thus the attention weights between any pair of tokens must be the same regardless of their positions.

Note there is a subtlety here implied in the question. This assumes that model considers the entire sequence. As we will see in a moment, in decoder-only architectures, attention is masked to prevent "looking ahead" at future tokens (termed "causal masking"). In that case, the causal masking introduces ordering information, i.e., "man" can attend to "dog" in sequence 1, but not in sequence 2.

For a sequence length of $n = 1024$ tokens and model dimension of $d_{model} = 768$, what is the size of the attention weight matrix in a self-attention layer?

a. $\mathcal{O}(n)$

b. $\mathcal{O}(d_{model})$

c. $\mathcal{O}(n \times d_{model})$

d. $\mathcal{O}(n^2)$

e. $\mathcal{O}(n^2 \times d_{model})$

Slide 6 Notes

Answer: D

Recall that the attention weight matrix describes how each token attends to every other token in the sequence. Thus it is a square matrix of size $n \times n$, i.e., $\mathcal{O}(n^2)$. The answer does not depend on the model dimension d_{model} or the number of heads (which is a constant factor in this context). What does answer E describe? The computational complexity of computing the attention weights, which involves d_{model}-dimensional dot products for each of the n^2 pairs of tokens.

In the sentence “The doctor asked the nurse a question. She”, which token(s) do you expect “She” to most strongly attend to in Layer 5 Head 10 of GPT2 (the same head we visualized earlier)?

a. “doctor”

b. “nurse”

c. “The”

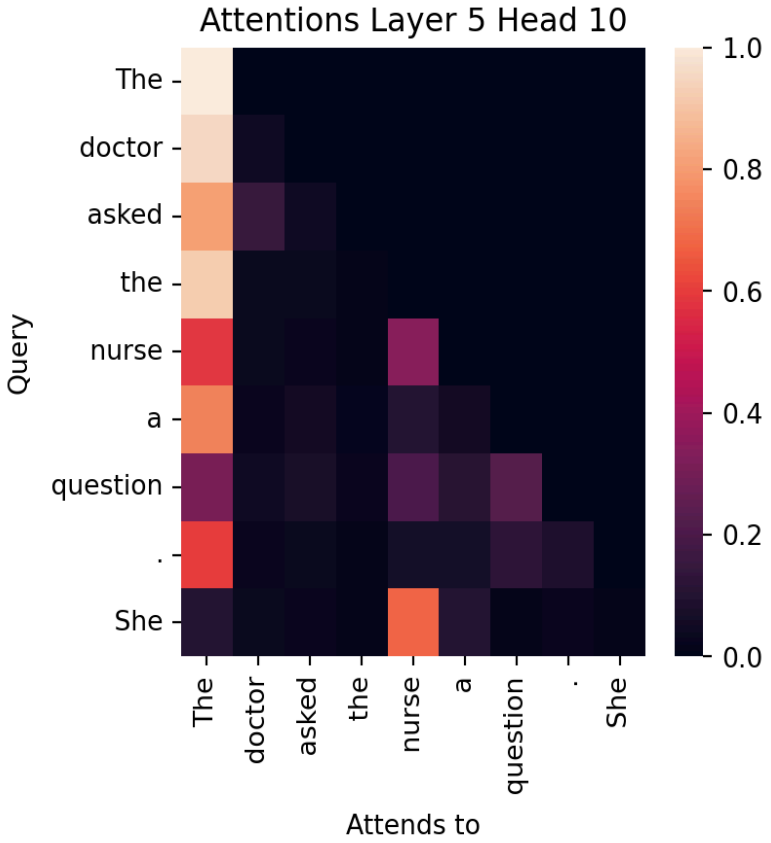
d. “question”

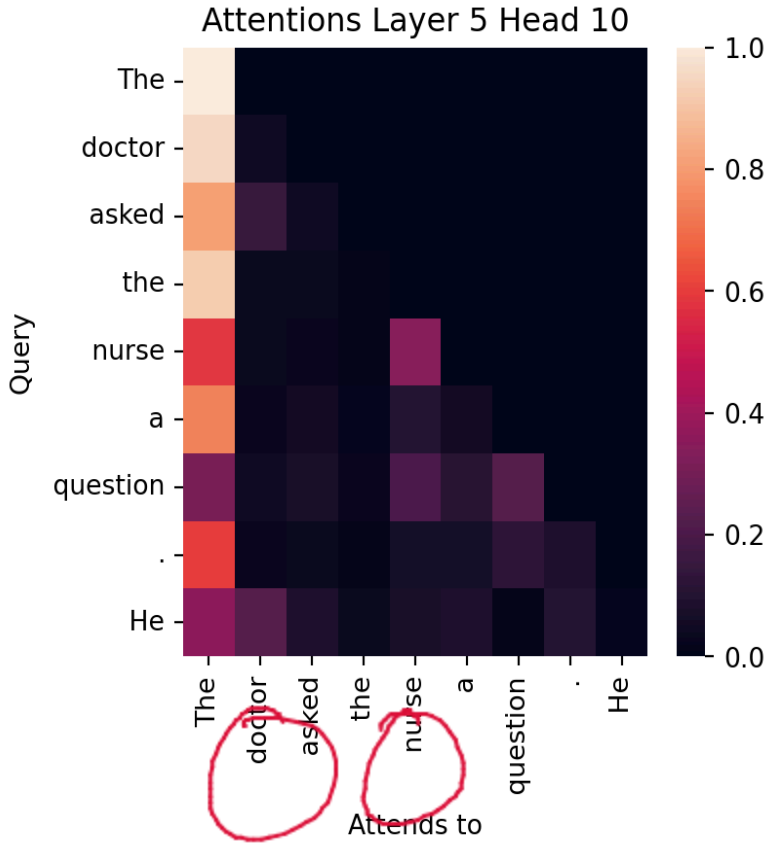
e. “doctor” and “nurse” equally

Slide 7 Notes

Answer: B

This head appears to focus on gendered pronoun resolution, i.e., linking pronouns to their antecedents. In this case, we expect the training data to most commonly describe/imply nurses as female, and thus expect “She” to attend most strongly to “nurse”.


Attentions Layer 5 Head 10
Query
The
doctor
asked
the
nurse
a
question
.
She
Attends to
The
doctor
asked
the
nurse
a
question
.
She
1.0
0.8
0.6
0.4
0.2
0.0


Attentions Layer 5 Head 10
Query
The
doctor
asked
the
nurse
a
question
.
He
Attends to
The
doctor
asked
the
nurse
a
question
.
He
1.0
0.8
0.6
0.4
0.2
0.0

Slide 8 Notes

Notice that the model has learned gendered associations. “She” attends strongly to “nurse”, while “He” attends strongly to “doctor”. It is not that the developer’s explicitly programmed these associations into the model; rather, the model learned these associations from its training data. That training data incorporates past (and current) asymmetric gender distributions in those professions, societal stereotypes, and more as reflected in Internet text. Specifically, we would expect there are many more examples of “doctor” being referred to as “he” and “nurse” being referred to as “she” online. And the model reflects those patterns back to us in the parameters it learns and the text it generates. When we talk about biases in LLMs (our next topic), this is one example of what we are talking about. These models are only as “neutral” as the data they are trained on.