Natural Language Processing

Tutorial 9 (Week 12): Attention / Transformer

Summary and Recap for Week 11

- Attention basics
- Query-Key-Value formulation
- Self attentions
- Transformers

Recap – Attentions

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

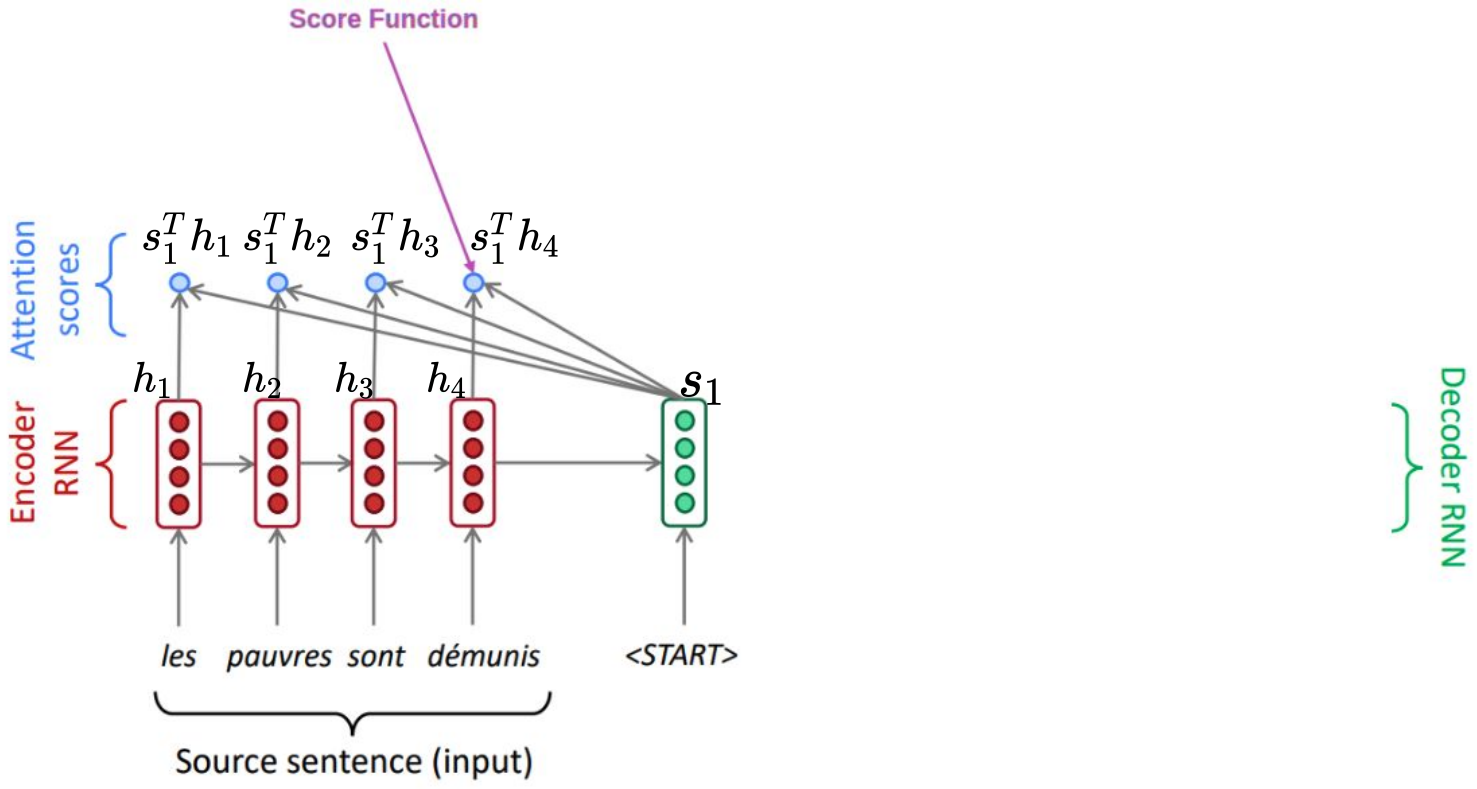
- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

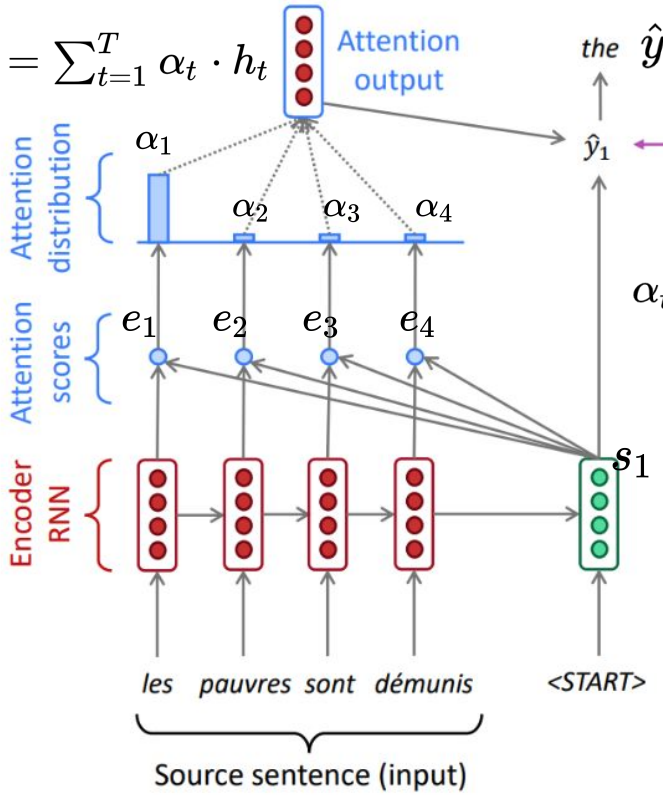
Recap – Attentions



Recap – Attentions

$$a_1 = \sum_{t=1}^{T} \alpha_t \cdot h_t$$

$$e_t = s_1^T h_t$$



the $\hat{y}_t = softmax(W_y \cdot [a_t; s_t] + b_y)$

$\hat{y}_1$

Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

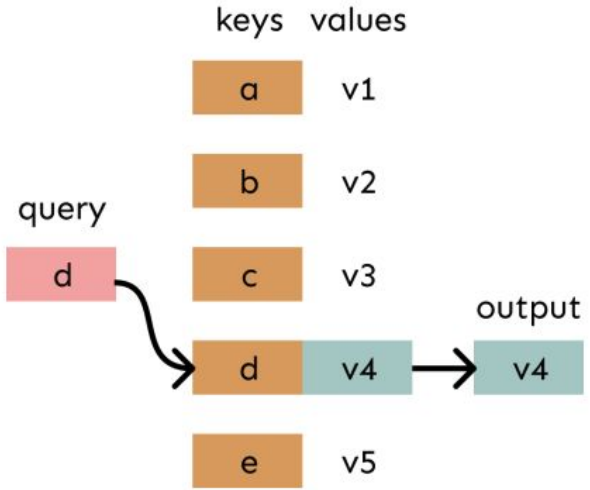
$$\alpha_t = softmax(e_t) = \frac{\exp(e_t)}{\sum_{i=1}^{T} \exp(e_i)}$$

Decoder RNN

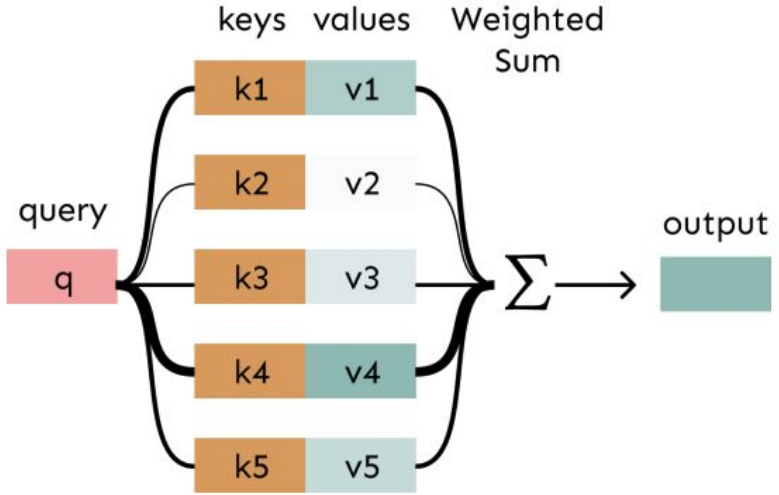
Recap – Attentions as QKV

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.



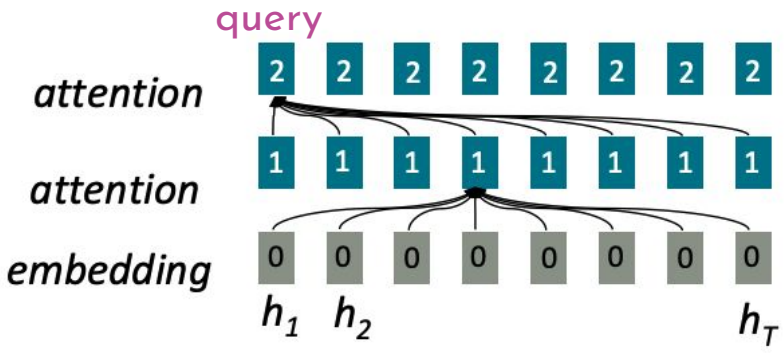
In **attention**, the **query** matches all **keys** *softly*, to a weight between 0 and 1. The keys' **values** are multiplied by the weights and summed.



Recap – Self Attentions

- Treats each word's representation as a query to access and incorporate information from a set of values.
- Easy to parallelize (per layer).
- Maximum interaction distance: O(1), since all words interact at every layer!

Each word can be query, key, value



All words attend to all words in previous layer; most arrows here are omitted

key/value

Recap – Self Attentions

Let $\boldsymbol{w}_{1:n}$ be a sequence of words in vocabulary V, like *Zuko made his uncle tea*.

For each $\boldsymbol{w}_i$, let $\boldsymbol{x}_i = E\boldsymbol{w}_i$, where $E \in \mathbb{R}^{d \times |V|}$ is an embedding matrix.

1. Transform each word embedding with weight matrices $\mathrm{Q}, \mathrm{K}, \mathrm{V}$, each in $\mathbb{R}^{d \times d}$

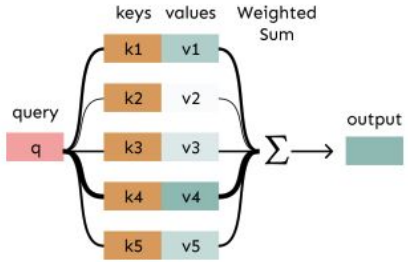
$$\boldsymbol{q}_i = Q\boldsymbol{x}_i \text{ (queries)} \qquad \boldsymbol{k}_i = K\boldsymbol{x}_i \text{ (keys)} \qquad \boldsymbol{v}_i = V\boldsymbol{x}_i \text{ (values)}$$

2. Compute pairwise similarities between keys and queries; normalize with softmax

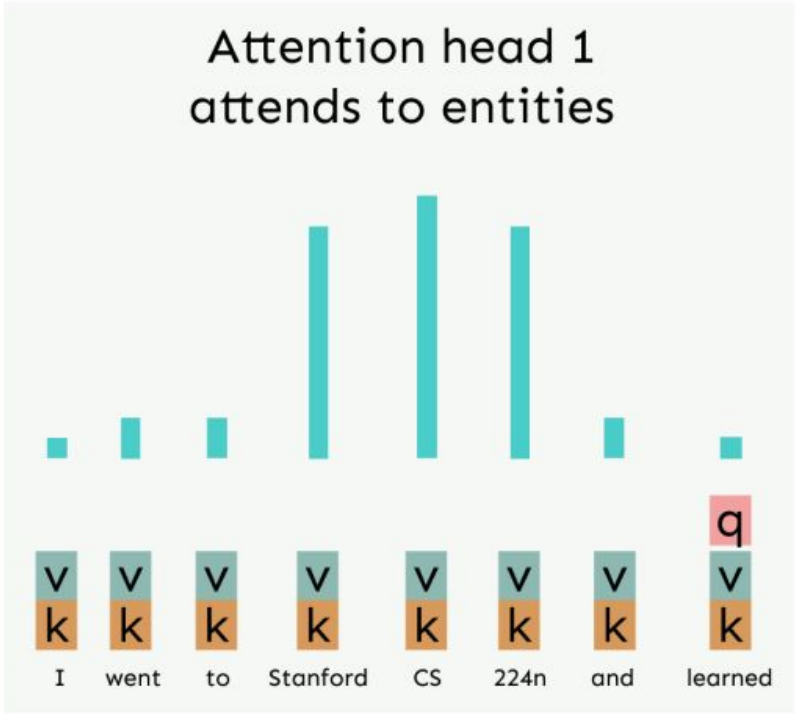
$$\boldsymbol{e}_{ij} = \boldsymbol{q}_i^\top \boldsymbol{k}_j \qquad \boldsymbol{\alpha}_{ij} = \frac{\exp(\boldsymbol{e}_{ij})}{\sum_{j'} \exp(\boldsymbol{e}_{ij'})}$$

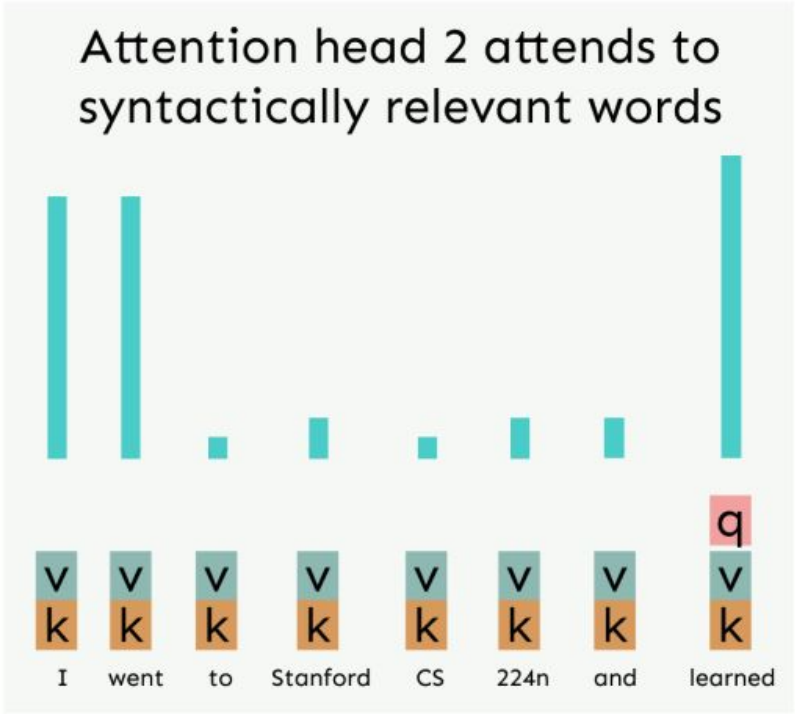
3. Compute output for each word as weighted sum of values

$$\boldsymbol{o}_i = \sum_j \boldsymbol{\alpha}_{ij} \boldsymbol{v}_i$$

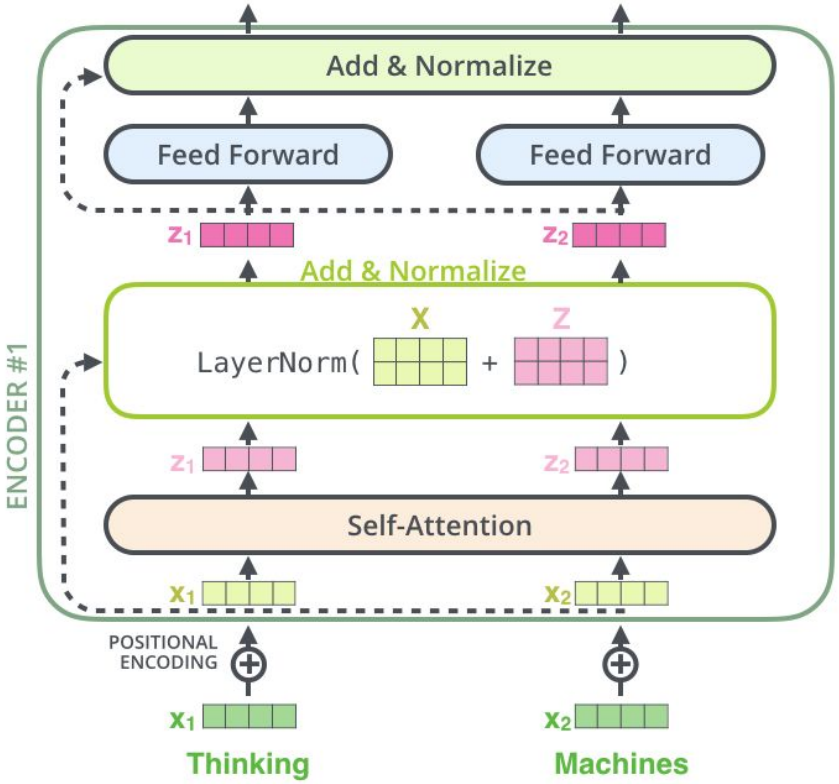


Recap – Transformers (Multi-head)





Recap – Transformers (Add & Norm)



Question 1

Consider a question-answering system where an attention-based seq2seq model is given a passage: "The Nile is the longest river in the world. It flows through northeastern Africa." The system needs to answer questions based on this passage.

1) Describe the step-by-step decoding process that the seq2seq model with attention uses to generate each token to answer the question "Which river is the longest?". What is the input for this task? (Assume the decoder hidden states are given as s_1, s_2,..., s_T. The encoder hidden states are h_1, h_2,..., h_N)

Solution 1(1)

The given passage is: "The Nile is the longest river in the world. It flows through northeastern Africa."

The question is: “Which river is the longest?”

The input should be the combination of both the passage (context) and the question: “The Nile is the longest river in the world. It flows through northeastern Africa. Which river is the longest?”

Solution 1 (1)

1) Given the input, at each decoding step:
   a) The decoder computes attention weights for each encoder hidden state h_n based on its current hidden state s_t. Here the encoder produces a hidden state for each token in "The Nile is the longest river in the world. It flows through northeastern Africa. Which river is the longest?"

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

$$\alpha^t = softmax(e^t) = [\frac{\exp(e_1)}{\sum_{i=1}^N \exp(e_i)}, \ldots, \frac{\exp(e_N)}{\sum_{i=1}^N \exp(e_i)}]$$

Solution 1 (1)

1) Given the input, at each decoding step:
   b) Context vector is computed as a weighted sum of the encoder hidden states:

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

   c) The output word probability distribution is then calculated as:

$$\hat{y}_t = softmax(W_y \cdot [a_t; s_t] + b_y)$$

   d) The word with the highest probability is chosen as the next word in the sequence.

Question 1

2) Assume when the decoder is generating the first token, the attention distribution is:

The	Nile	is	the	longest	river	in	the	world	.	It
0.4	0.2	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0
flows	through	north-eastern	Africa	.	Which	river	is	the	longest	?
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0

Question 1

2) And the attention distribution for generating the second token is:

The	**Nile**	is	the	longest	river	in	the	world	.	It
0.0	0.6	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0
flows	through	north-eastern	Africa	.	Which	river	is	the	longest	?
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0

Discuss what these attention distributions suggest? What could possibly be the two tokens being generated?

Solution 1 (2)

Attention distributions indicate the extent to which the input tokens could affect each decoding step. Given the two distributions, we can clearly observe that

- When generating the first token, the model only focuses on “The”, “Nile”, “longest”, “river”, with “The” having the largest attention. Hence, the first generated token is highly likely to be “The”.
- When generating the second token, the model only focuses on “Nile”, “longest”, “river”, with “Nile” having the largest attention. Hence, the second generated token is highly likely to be “Nile”.

Question 1

Consider a question-answering system where an attention-based seq2seq model is given a passage: "The Nile is the longest river in the world. It flows through northeastern Africa." The system needs to answer questions based on this passage.

3) Now suppose the question has been changed to “Where does the longest river flow?”. Will that affect the attention distributions when performing the decoding steps? Why? What’s the token(s) with the highest attention score(s) if the model is well-trained?

Solution 1 (3)

- In answering "Where does the longest river flow?", the model will produce different attention distributions compared with the previous question. This is because the input has been changed, leading to the change of each decoder hidden state (s_t), thus producing different attention scores.

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- In this case, the highly attended tokens should be “northeastern Africa”.

Question 1

Consider a question-answering system where an attention-based seq2seq model is given a passage: "The Nile is the longest river in the world. It flows through northeastern Africa." The system needs to answer questions based on this passage.

4) Imagine the passage is much longer and contains detailed descriptions of several rivers. How might this additional information impact the attention mechanism during decoding? What are some strategies for modifying the attention mechanism to maintain accuracy and focus in long passages?

Solution 1 (4)

In long passages, attention mechanisms can struggle to maintain focus and accuracy due to the sheer volume of information. The attention distribution can become really flat and lose focus on relevant tokens. Possible solutions include:

1. **Hierarchical attention**: Divide the passage into smaller segments and apply attention within each segment before combining them.
2. **Sparse attention**: Add regularizer (e.g., L1 loss) on attention scores to enforce many attention scores to be 0.

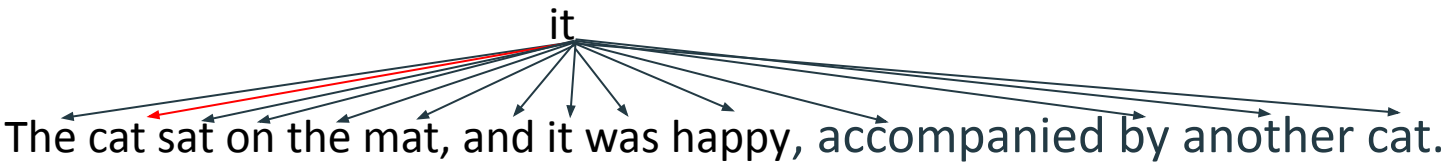
Question 2

Consider a self attention model that is processing the sentence "The cat sat on the mat, and it was happy, accompanied by another cat." This sentence is used as an input to a self-attention layer to obtain a hidden representation for each word.

1) Describe how self-attention captures the context of each word in the sentence. Specifically, detail how self-attention helps the model understand the relationship between "it" and the first "cat."

Solution 2 (1)

Self-attention allows each word in the input sentence to interact with every other word, helping the model to understand the context around each word. For "The cat sat on the mat, and it was happy, accompanied by another cat." the model uses self-attention to compute a set of weights that indicates the relevance of every word to "it." For example, the model may learn to assign a higher weight between "it" and "cat" because, in this context, "it" is an anaphoric reference to "cat." The self-attention mechanism can capture this relationship even though the words are not adjacent.



Question 2

Consider a self attention model that is processing the sentence "The cat sat on the mat, and it was happy, accompanied by another cat." This sentence is used as an input to a self-attention layer to obtain a hidden representation for each word.

2) Walk through the computation of self-attention weights for the word "it" in the given sentence. Explain how the self-attention scores between "it" and all other words in the sentence are calculated.

Solution 2 (2)

The self-attention scores are computed by taking the dot product of the query vector for "it" with the key vectors for all words in the sentence, including "it" itself. These scores are then normalized using a softmax function to yield the final attention weights. Mathematically:

$$q_{it} = Q \cdot x_{it} \in \mathbb{R}^d, Q \in \mathbb{R}^{d \times d}$$

$$k_n = K \cdot x_n \in \mathbb{R}^d, K \in \mathbb{R}^{d \times d}$$

$$v_n = V \cdot x_n \in \mathbb{R}^d, V \in \mathbb{R}^{d \times d}$$

$$e_{it,n} = q_{it}^T \cdot k_n \qquad \alpha_{it,n} = \frac{\exp(e_{it,n})}{\sum_{n'} \exp(e_{it,n'})}$$

Question 2

Consider a self attention model that is processing the sentence "The cat sat on the mat, and it was happy, accompanied by another cat." This sentence is used as an input to a self-attention layer to obtain a hidden representation for each word.

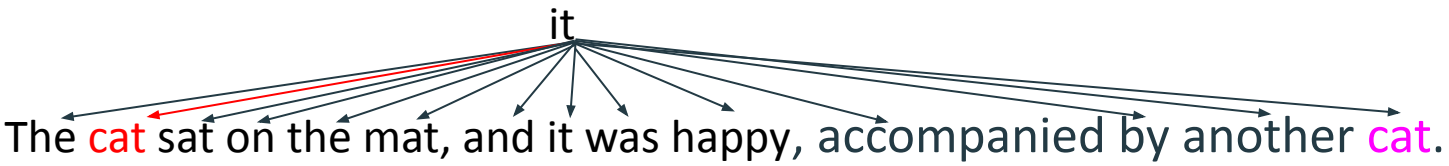
3) There are two occurrences of the word "cat" in the sentence. How might the self-attention model differentiate between these instances when processing the word "it"? What mechanisms or additional model adjustments could help disambiguate references in such cases?

Solution 2 (3)

Positional encodings are added to the input embeddings to provide the model with information about the order of words. Since self-attention does not inherently process words sequentially, these encodings are necessary for the model to utilize the sequential nature of language.

$$\boldsymbol{x}_n = \boldsymbol{p}_n + \boldsymbol{x}_n \in \mathbb{R}^d$$

For this sentence "The cat sat on the mat, and it was happy, accompanied by another cat." They allow the model to know that "it" refers to the first “cat” instead of the second “cat” because the two words have different position encodings.



Question 2

Consider a self attention model that is processing the sentence "The cat sat on the mat, and it was happy, accompanied by another cat." This sentence is used as an input to a self-attention layer to obtain a hidden representation for each word.

4) In self-attention, each word’s hidden representation is influenced by other words in the sentence. After self-attention is applied, how might the representation for the word "cat" differ between its first and second occurrence?

Solution 2 (4)

In self-attention, the representations for each occurrence of the same word "cat" are adjusted based on their unique attention distribution over all context tokens within the sentence. This is only possible if positional encodings are incorporated.

- The first "cat" might attend more strongly to words like "sat," "on the mat," and potentially "happy," capturing the setting and actions associated with it.
- The second "cat" might attend to "another" and "accompanied," emphasizing its role as a secondary, accompanying figure.

$$q_{cat_1} = Q \cdot (x_{cat} + p_{cat_1}) \qquad q_{cat_2} = Q \cdot (x_{cat} + p_{cat_2})$$

$$h_{cat_1} = \sum_n \alpha_{cat_1,n} v_n \qquad h_{cat_2} = \sum_n \alpha_{cat_2,n} v_n$$

Question 3

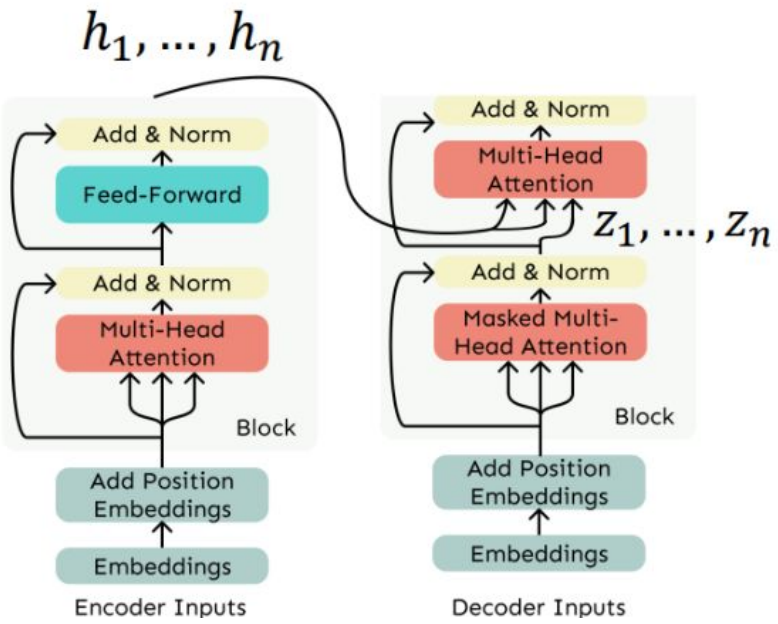
Imagine using a Transformer model with a single encoder block and decoder block to translate the English sentence "The quick brown fox jumps over the lazy dog" into French.

1) Discuss how information is transferred from the encoder to the decoder in a Transformer model. How does self-attention function in this scenario (i.e., crossing the encoder and decoder)?

Solution 3 (1)

Information from the encoder is passed to the decoder through the encoder's hidden vectors.

In the decoder, these hidden vectors are used in the encoder-decoder cross attention mechanism, which allows each word in the decoder to attend to all words in the encoder. This process is crucial for the decoder to focus on relevant parts of the input sentence at each step of the generation process.

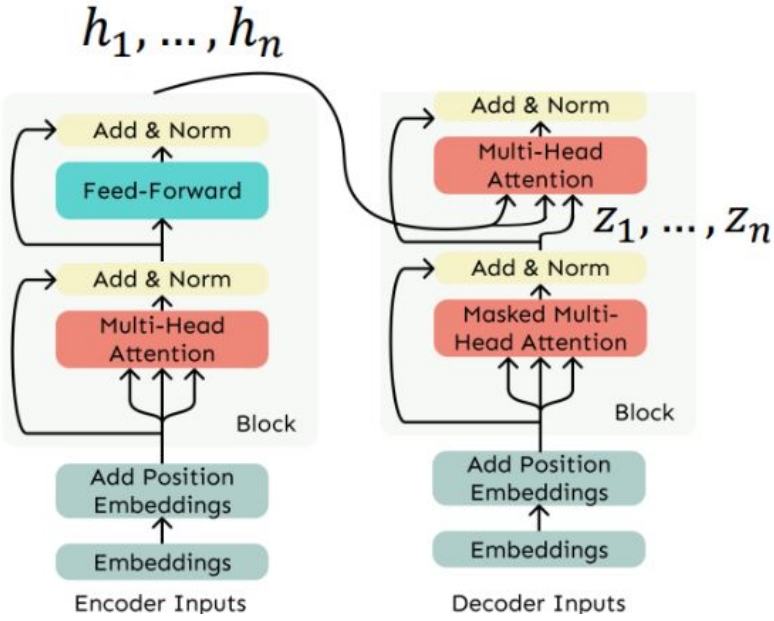


Solution 3 (1)

- The encoder layer contains: a multi-head self attention, Add & Norm layers and a feed-forward layer to produce h_i.
- These hidden vectors from the encoder are then used as keys and values in the decoder's attention to focus on different parts of the input sequence.
- Before that, the decoder needs to go through a self-attention layer to produce hidden vectors z_j at each decoding step j. z_j is then used to produce a query vector:

$$k_i = K_{encdec} h_i \qquad v_i = V_{encdec} h_i$$

$$q_j = Q_{encdec} z_j$$



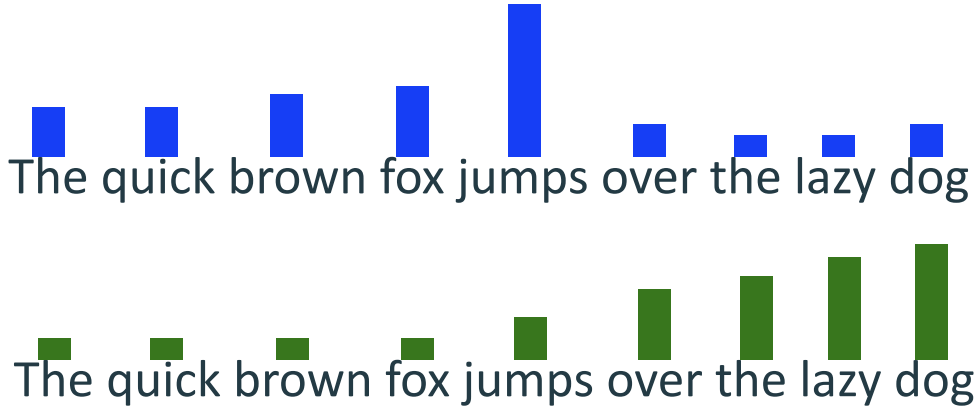
Question 3

Imagine using a Transformer model with a single encoder block and decoder block to translate the English sentence "The quick brown fox jumps over the lazy dog" into French.

2) Explain the role of multi-head attention in the context of translating the phrase "jumps over." How might different attention heads capture various aspects of this phrase's meaning and grammar?

Solution 3 (2)

Multi-head attention allows the model to focus on different parts of the input sentence simultaneously. For a phrase like "jumps over," one head might focus on "jumps" and its association with action, while another might relate "over" to the spatial relationship with "the lazy dog." This helps capture a multifaceted understanding of the phrase that contributes to a more nuanced translation.



Question 3

Imagine using a Transformer model with a single encoder block and decoder block to translate the English sentence "The quick brown fox jumps over the lazy dog" into French.

3) Explain the function of the "Add and Norm" layers found after each sub-layer in the encoder and decoder. Why are these layers critical to the Transformer's architecture?

Solution 3 (3)

- **Layer Norm**
  - Layer normalization is critical for stabilizing the training of the Transformer model. By normalizing the inputs across the features, it ensures that the scale of the inputs does not become too large or too small, which can lead to training difficulties.
- **Add (Residual connection)**
  - The "Add" part refers to the addition of the input of the sub-layer to the output of the sub-layer x + f(x). It allows the gradient to bypass the non-linear transformations f, which helps mitigate the vanishing gradient problem.

Question 3

Imagine using a Transformer model with a single encoder block and decoder block to translate the English sentence "The quick brown fox jumps over the lazy dog" into French.

4) In cases of translation ambiguity, such as the word "over" with multiple meanings, how might the Transformer model disambiguate the correct sense of the word during translation?

Solution 3 (4)

In the case of ambiguous words like "over", the Transformer model uses the context provided by the attention mechanism to disambiguate the meaning. It looks at surrounding words and the overall sentence structure to determine which meaning of "over" is being used and translates accordingly.

Coding

https://colab.research.google.com/drive/1nAUqAP4BQF2xFSq5V-50Bn_xEzx_m0Jm?usp=sharing