Attention Is All You Need:

Deriving the Seminal Transformer Architecture from First Principles

J. Setpal

September 3, 2024

MACHINE LEARNING @ PURDUE

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$

b. **Logit:** Pre-activation scores for the *final layer*.

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$

b. **Logit:** Pre-activation scores for the *final layer*.

c. **Dot / Inner Product:** $\langle a, b \rangle = \sum_{i=1}^{N} a_i b_i$

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$

b. **Logit:** Pre-activation scores for the *final layer*.

c. **Dot / Inner Product:** $\langle a, b \rangle = \sum_{i=1}^{N} a_i b_i$

d. **Matrix Multiplication:** For $X \in \mathbb{R}^{a \times b}$, $Y \in \mathbb{R}^{b \times c}$,
$Z = XY \in \mathbb{R}^{a \times c}$ $s.t.$ $z_{ij} = \sum_{k=1}^{b} a_{ik} b_{kj}$

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$
b. **Logit:** Pre-activation scores for the *final layer*.
c. **Dot / Inner Product:** $\langle a, b \rangle = \sum_{i=1}^{N} a_i b_i$
d. **Matrix Multiplication:** For $X \in \mathbb{R}^{a \times b},\ Y \in \mathbb{R}^{b \times c}$, $Z = XY \in \mathbb{R}^{a \times c}$ *s.t.* $z_{ij} = \sum_{k=1}^{b} a_{ik} b_{kj}$
e. **Activation:** Non-linear function over the output of matrix multiplication.

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$

b. **Logit:** Pre-activation scores for the *final layer*.

c. **Dot / Inner Product:** $\langle a, b \rangle = \sum_{i=1}^{N} a_i b_i$

d. **Matrix Multiplication:** For $X \in \mathbb{R}^{a \times b}$, $Y \in \mathbb{R}^{b \times c}$, $Z = XY \in \mathbb{R}^{a \times c}$ $s.t.$ $z_{ij} = \sum_{k=1}^{b} a_{ik} b_{kj}$

e. **Activation:** Non-linear function over the output of matrix multiplication.

f. **Gradient:** $\nabla_w \mathcal{L}(w)$, derivative of $\mathcal{L} : \mathbb{R}^a \to \mathbb{R}$

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$

b. **Logit:** Pre-activation scores for the *final layer*.

c. **Dot / Inner Product:** $\langle a, b \rangle = \sum_{i=1}^{N} a_i b_i$

d. **Matrix Multiplication:** For $X \in \mathbb{R}^{a \times b}$, $Y \in \mathbb{R}^{b \times c}$, $Z = XY \in \mathbb{R}^{a \times c}$ *s.t.* $z_{ij} = \sum_{k=1}^{b} a_{ik} b_{kj}$

e. **Activation:** Non-linear function over the output of matrix multiplication.

f. **Gradient:** $\nabla_w \mathcal{L}(w)$, derivative of $\mathcal{L} : \mathbb{R}^a \to \mathbb{R}$

g. **Latent Vector:** $h^{(\ell)}$, intermediary output from within a neural network.

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$
b. **Logit:** Pre-activation scores for the *final layer*.
c. **Dot / Inner Product:** $\langle a, b \rangle = \sum_{i=1}^{N} a_i b_i$
d. **Matrix Multiplication:** For $X \in \mathbb{R}^{a \times b}$, $Y \in \mathbb{R}^{b \times c}$, $Z = XY \in \mathbb{R}^{a \times c}$ $s.t.$ $z_{ij} = \sum_{k=1}^{b} a_{ik} b_{kj}$
e. **Activation:** Non-linear function over the output of matrix multiplication.
f. **Gradient:** $\nabla_w \mathcal{L}(w)$, derivative of $\mathcal{L} : \mathbb{R}^a \to \mathbb{R}$
g. **Latent Vector:** $h^{(\ell)}$, intermediary output from within a neural network.
h. **Embedding:** A look-up table that translates categorical values (words) to vectors.

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river <u>bank</u> yesterday.

b. I am NOT going to rob a <u>bank</u> tomorrow at 8:30am ET.

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river <u>bank</u> yesterday.

b. I am NOT going to rob a <u>bank</u> tomorrow at 8:30am ET.

Insight: Context is important.

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river bank yesterday.

b. I am NOT going to rob a bank tomorrow at 8:30am ET.

Insight: Context is important.

n-gram models use the following (Markov) assumption:

$$p\left(x_t \middle| \{x_i\}_{i=1}^{t-1}; \theta\right) \approx p\left(x_t | x_{t-1}; \theta\right) \tag{1}$$

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river <u>bank</u> yesterday.

b. I am NOT going to rob a <u>bank</u> tomorrow at 8:30am ET.

Insight: Context is important.

n-gram models use the following (Markov) assumption:

$$p\left(x_t | \{x_i\}_{i=1}^{t-1}; \theta\right) \approx p\left(x_t | x_{t-1}; \theta\right) \tag{1}$$

This is **incorrect.**

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river <u>bank</u> yesterday.

b. I am NOT going to rob a <u>bank</u> tomorrow at 8:30am ET.

Insight: Context is important.

n-gram models use the following (Markov) assumption:

$$p\left(x_t \middle| \{x_i\}_{i=1}^{t-1}; \theta\right) \approx p\left(x_t | x_{t-1}; \theta\right) \tag{1}$$

This is **incorrect.** Context is important!

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

How Expensive is Context?

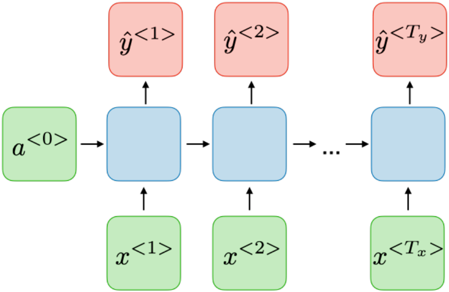
Q: So, why do n-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

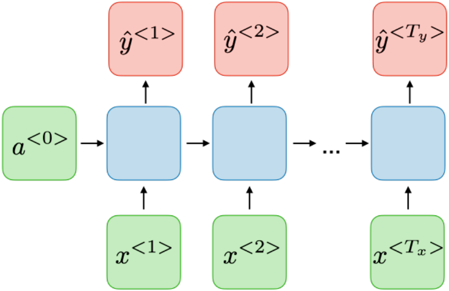
Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):



How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):

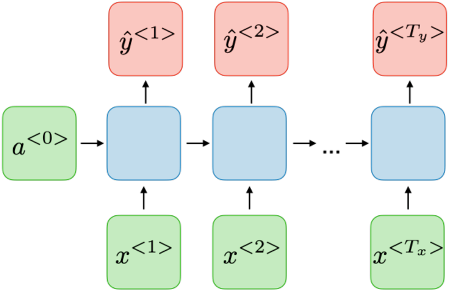


Here, a hidden representation $a^{\langle \ell \rangle}$ is propogated, encoding relevant context.

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):



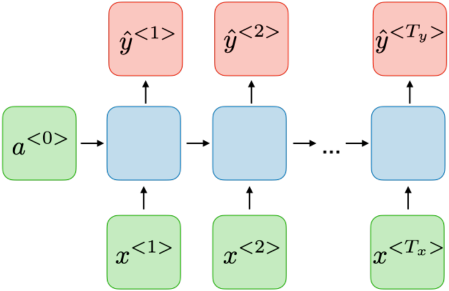
Here, a hidden representation $a^{\langle \ell \rangle}$ is propogated, encoding relevant context.

Caveat: $a^{\langle \ell \rangle} \in \mathbb{R}^d$, with *fixed* d.

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):



Here, a hidden representation $a^{\langle \ell \rangle}$ is propogated, encoding relevant context.

Caveat: $a^{\langle \ell \rangle} \in \mathbb{R}^d$, with *fixed* d. This mitigates capability for long-term memory & recollection.

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

Q: Well, what does it mean to "incorporate attention"?

[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

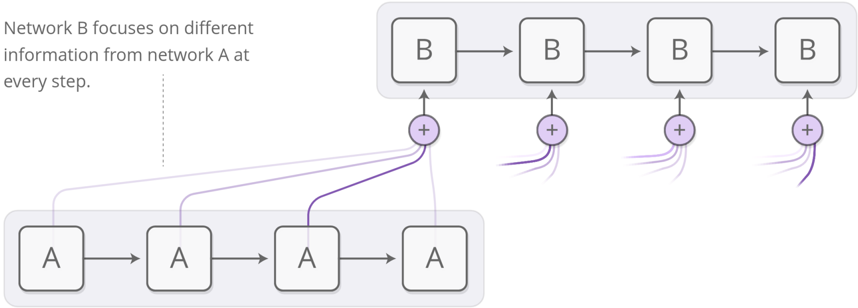
Q: Well, what does it mean to "incorporate attention"?
A: For latent vectors $\{\ell_i\}_{i=1}^N$, $r = \sum_{i=1}^N \ell_i \to \sum_{i=1}^N w_i \ell_i$,
where $\sum_{i=1}^N w_i = 1$, $w_i \geq \ \forall i$

[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

Q: Well, what does it mean to "incorporate attention"?
A: For latent vectors $\{\ell_i\}_{i=1}^{N}$, $r = \sum_{i=1}^{N} \ell_i \to \sum_{i=1}^{N} w_i \ell_i$,
where $\sum_{i=1}^{N} w_i = 1, \;\; w_i \geq \;\; \forall i$



[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{2}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{3}$$

(4)

(5)

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{2}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{3}$$

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{4}$$

(5)

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{2}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{3}$$

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{4}$$

$$h(x) = \sum_{i=1}^{t} \alpha_i v_i \tag{5}$$

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{2}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{3}$$

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{4}$$

$$h(x) = \sum_{i=1}^{t} \alpha_i v_i \tag{5}$$

Where q_i, k_i, v_i are each independently computed latent matrices.

$$\text{Self-Attention}(Q, K, V) = \left(\frac{QK^T}{\sqrt{d_{out}}}\right) V \tag{6}$$

Positional Encoding

A consequence of this setup, howeveer is that we are not considering the order of the tokens anymore. It is **permutation invariant**.

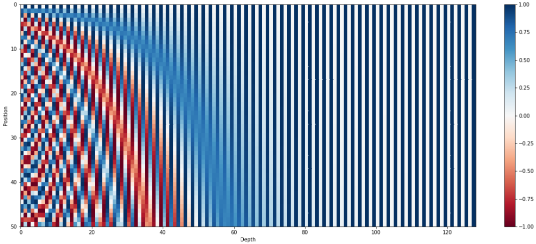
Positional Encoding

A consequence of this setup, howeveer is that we are not considering the order of the tokens anymore. It is **permutation invariant**.

To resolve this, we add positional encodings to each word embedding:

$$PE_{(pos,2i)} = \sin\left({}^{pos}\!/_{1E4^{2i/d_{model}}}\right) \tag{7}$$

$$PE_{(pos,2i+1)} = \sin\left({}^{pos}\!/_{1E4^{2i/d_{model}}}\right) \tag{8}$$



Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Intuitively, a different set of information may be desired from the same input.
e.g: NER, Sentence-Structure Decomposition, POS tagging – any valuable information that can inform the output.

Multi-Headed Setting

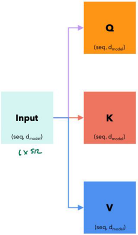
Each block of self-attention constitutes it's own head.

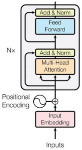
Intuitively, a different set of information may be desired from the same input.
e.g: NER, Sentence-Structure Decomposition, POS tagging – any valuable information that can inform the output.

Therefore, we instantiate *multiple* heads within each layer, and concatenate to construct a final output represetnation.

$$MHA = \text{Concat}(\{h_i\}_{i=1}^{H})W_O \tag{9}$$

Self-Attention (2/2)



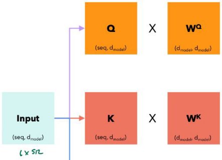


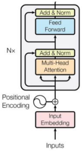
$$Attention(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)





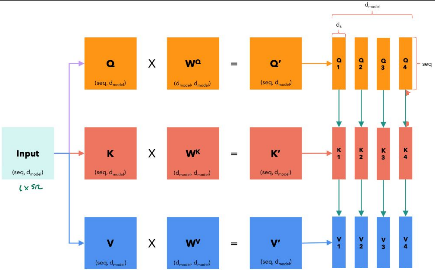
(copy)

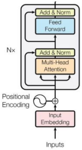
$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)





copy

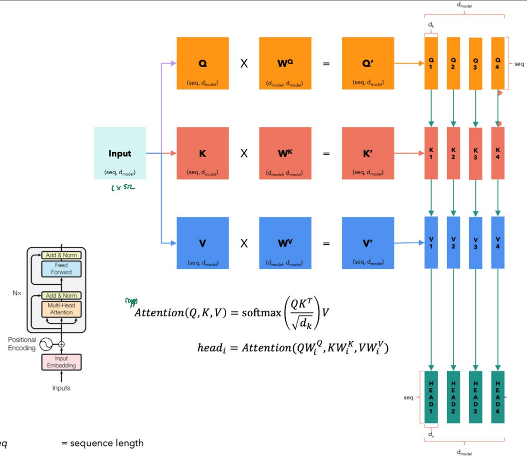
$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

s

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)

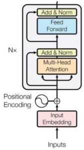


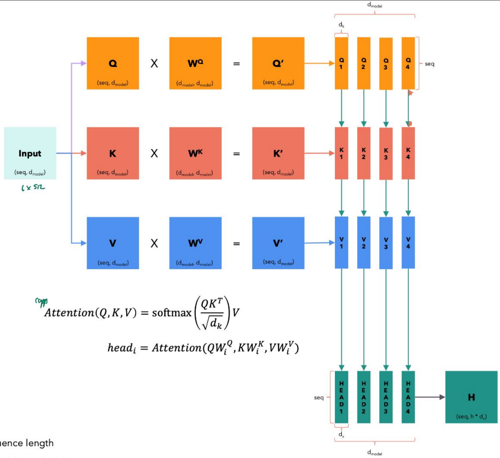
$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)





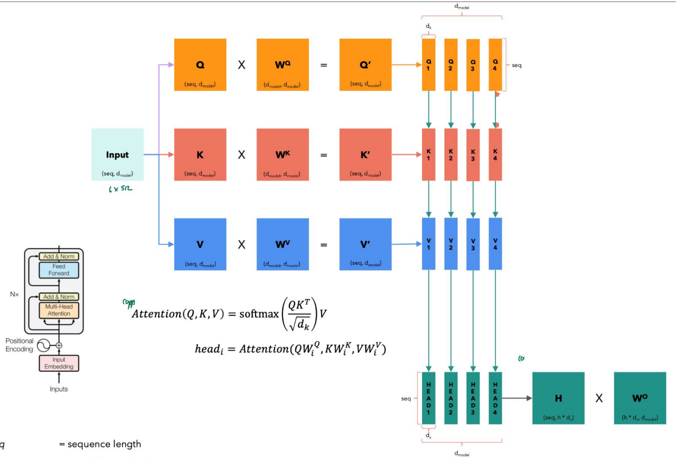
copy

$$\mathit{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathit{head}_i = \mathit{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)



$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

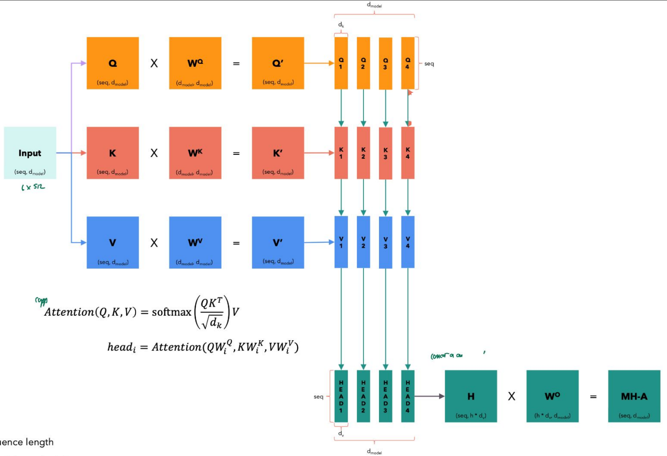
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

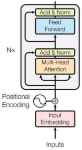
seq = sequence length

d_{model} = size of the embedding vector

h = number of heads

$d_k = d_v$ = d_{model} / h





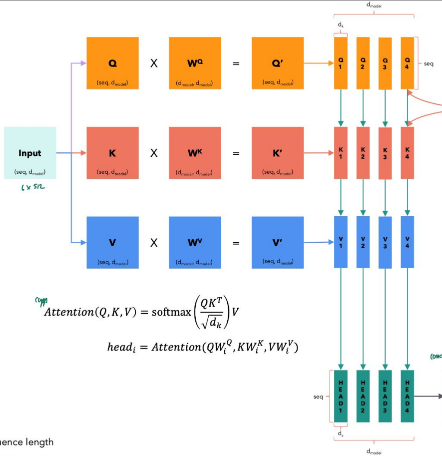
$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

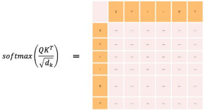
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q,K,V) = Concat(head_1 \; ... head_h)W^O$$

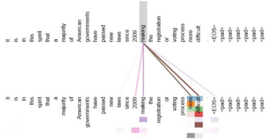
seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

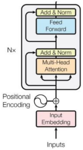
Self-Attention (2/2)





Attention Visualizations



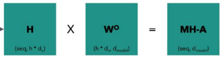


copy

$$Attention(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

concat on



$$MultiHead(Q,K,V) = Concat(head_1 \ldots head_h)W^O$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Feedforward MLP

Thanks to attention, we now have an updated representation of our input. We now need to perform operations on this contextualized input to make inferences about our actual objective.

Feedforward MLP

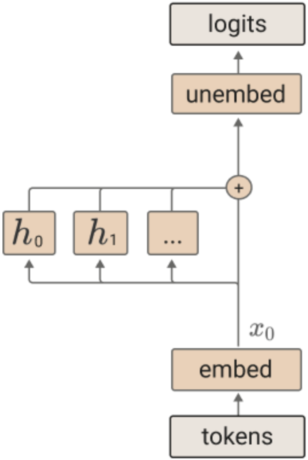
Thanks to attention, we now have an updated representation of our input. We now need to perform operations on this contextualized input to make inferences about our actual objective.

For this, a two-layer MLP is instantiated that expands and consequently contracts the input dimension.

$$FFN(x) = \sigma_{relu}(xW_1 + b_1)W_2 + b_2 \tag{10}$$

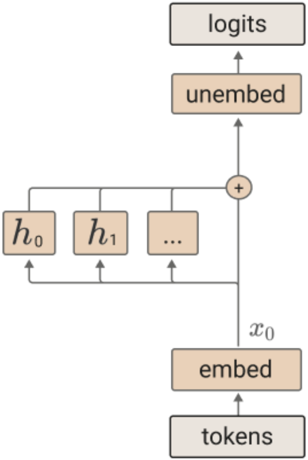
The original paper uses a factor of 4.

Scaling to Deeper Networks – A Better Interpretation



Deep neural architectures struggle with vanishing gradients, when trained at scale.

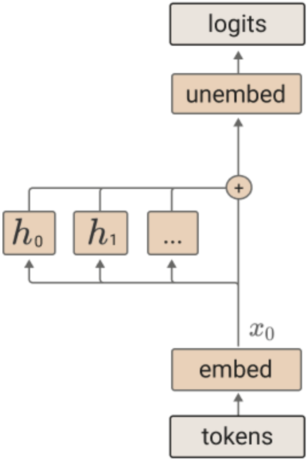
Scaling to Deeper Networks – A Better Interpretation



Deep neural architectures struggle with vanishing gradients, when trained at scale.

The approach used universally is to add “skip connections” to maintain a short gradient path, and mitigate this concern.

Scaling to Deeper Networks – A Better Interpretation

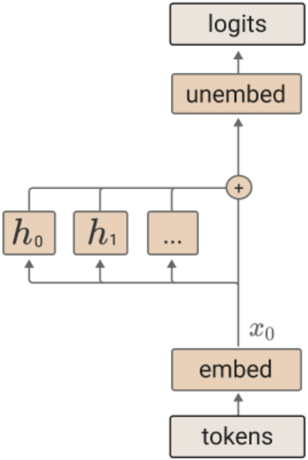


Deep neural architectures struggle with vanishing gradients, when trained at scale.

The approach used universally is to add "skip connections" to maintain a short gradient path, and mitigate this concern.

A result of this setup is that we can interpret each attention head and MLP as "reading from" and "writing to" a residual stream.

Scaling to Deeper Networks – A Better Interpretation



Deep neural architectures struggle with vanishing gradients, when trained at scale.

The approach used universally is to add "skip connections" to maintain a short gradient path, and mitigate this concern.

A result of this setup is that we can interpret each attention head and MLP as "reading from" and "writing to" a residual stream.

In addition, we also perform layer normalization over the latent vectors before MLP & self-attention.

Inference & Training

The output of the model during a single forward pass is a token.

Inference & Training

The output of the model during a single forward pass is a token. This token is then added to the context window, and the forward pass is run once again. This creates the final output, as a sentence.

Inference & Training

The output of the model during a single forward pass is a token. This token is then added to the context window, and the forward pass is run once again. This creates the final output, as a sentence.

For training, we exploit GPU parallelism, since each token sequence $\{x_i\}_{t=1}^{T}$ contains $T-1$ targets.

$$p\left(x_t | \{x_i\}_{i=1}^{t-1}\right) = x_t \;\forall t \in \{2, \ldots, T\} \tag{11}$$

Inference & Training

The output of the model during a single forward pass is a token. This token is then added to the context window, and the forward pass is run once again. This creates the final output, as a sentence.

For training, we exploit GPU parallelism, since each token sequence $\{x_i\}_{t=1}^{T}$ contains $T-1$ targets.

$$p\left(x_t | \{x_i\}_{i=1}^{t-1}\right) = x_t \ \forall t \in \{2, \dots, T\} \tag{11}$$

However, a consequence of this is that attention can *look into the future*.

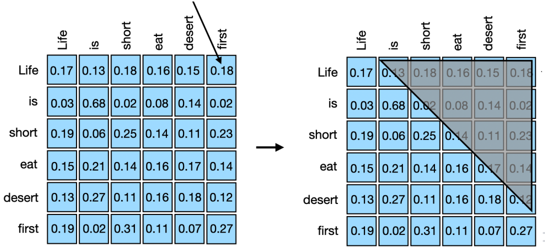
Inference & Training

The output of the model during a single forward pass is a token. This token is then added to the context window, and the forward pass is run once again. This creates the final output, as a sentence.

For training, we exploit GPU parallelism, since each token sequence $\{x_i\}_{t=1}^{T}$ contains $T-1$ targets.

$$p\left(x_t | \{x_i\}_{i=1}^{t-1}\right) = x_t \; \forall t \in \{2, \ldots, T\} \tag{11}$$

However, a consequence of this is that attention can *look into the future*. We prevent this by applying a causal mask:



Reviewing NanoGPT

If you can view this screen, I am making a mistake.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/transformer.pdf`