Introduction to Deep Learning

Lecture 19
Transformers and LLMs

Shikhar Agnihotri  Liangze Li

11-785, Fall 2023
Part 1

Transformers
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Residuals
- Query
- Key
- Value
- Add & Norm
- Encoder
- Decoder

- Attention
- Self Attention
- Multi Head Attention
- Masked Attention
- Encoder-Decoder Attention
- Output Probabilities / Logits
- Softmax
- Encoder-Decoder models
- Decoder only models
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Residuals
- Query
- Key
- Value
- Add & Norm
- Encoder
- Decoder

- Attention
- Self Attention
- Multi Head Attention
- Masked Attention
- Encoder-Decoder Attention
- Output Probabilities / Logits
- Softmax
- Encoder-Decoder models
- Encoder only models

Diagram showing the flow of data through the Transformer model, including input embedding, positional encoding, add & norm, encoder and decoder layers, and output probabilities/softmax.
Machine Translation

Targets
Ich have einen apfel gegessen

Inputs
I ate an apple
Inputs

I ate an apple
Inputs

I ate an apple

Generate Input Embeddings
I ate an apple

Generate Input Embeddings
WHERE IS THE CONTEXT?

Encoder
I ate an apple <eos>
I ate an apple.
I ate an apple.
I ate an apple
I ate an apple<br><br>CONTEXTUALLY RICH EMBEDDINGS<br><br>LEARN TO ADD CONTEXT<br><br>Encoder<br><br>$\alpha_{[ij]} \ ? \ \sum \ \prod \ ?$
Attention

\[ \alpha_{ij} ? \]

From lecture 18:

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

\[ \alpha_{ij} \ ? \]

From lecture 18:

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

- Query
- Key
- Value
Query, Key & Value

Database

{Key, Value store}

{"order_100": {"items": "a1", "delivery_date": "a2", ...}},
{"order_101": {"items": "b1", "delivery_date": "b2", ...}},
{"order_102": {"items": "c1", "delivery_date": "c2", ...}},
{"order_103": {"items": "d1", "delivery_date": "d2", ...}},
{"order_104": {"items": "e1", "delivery_date": "e2", ...}},
{"order_105": {"items": "f1", "delivery_date": "f2", ...}},
{"order_106": {"items": "g1", "delivery_date": "g2", ...}},
{"order_107": {"items": "h1", "delivery_date": "h2", ...}},
{"order_108": {"items": "i1", "delivery_date": "i2", ...}},
{"order_109": {"items": "j1", "delivery_date": "j2", ...}},
{"order_110": {"items": "k1", "delivery_date": "k2", ...}}
Query, Key & Value

Database

{Key, Value store}

{"order_100": {"items": "a1", "delivery_date": "a2", ...}},
{"order_101": {"items": "b1", "delivery_date": "b2", ...}},
{"order_102": {"items": "c1", "delivery_date": "c2", ...}},
{"order_103": {"items": "d1", "delivery_date": "d2", ...}},
{"order_104": {"items": "e1", "delivery_date": "e2", ...}},
{"order_105": {"items": "f1", "delivery_date": "f2", ...}},
{"order_106": {"items": "g1", "delivery_date": "g2", ...}},
{"order_107": {"items": "h1", "delivery_date": "h2", ...}},
{"order_108": {"items": "i1", "delivery_date": "i2", ...}},
{"order_109": {"items": "j1", "delivery_date": "j2", ...}},
{"order_110": {"items": "k1", "delivery_date": "k2", ...}}

{Query: “Order details of order_104”}

OR

{Query: “Order details of order_106”}
Query, Key & Value

{Query: “Order details of order_104”}
OR
{Query: “Order details of order_106”}

{Key, Value store}

{"order_100": {"items": "a1", "delivery_date": "a2", ...}},
{"order_101": {"items": "b1", "delivery_date": "b2", ...}},
{"order_102": {"items": "c1", "delivery_date": "c2", ...}},
{"order_103": {"items": "d1", "delivery_date": "d2", ...}},
{"order_104": {"items": "e1", "delivery_date": "e2", ...}},
{"order_105": {"items": "f1", "delivery_date": "f2", ...}},
{"order_106": {"items": "g1", "delivery_date": "g2", ...}},
{"order_107": {"items": "h1", "delivery_date": "h2", ...}},
{"order_108": {"items": "i1", "delivery_date": "i2", ...}},
{"order_109": {"items": "j1", "delivery_date": "j2", ...}},
{"order_110": {"items": "k1", "delivery_date": "k2", ...}}
Query, Key & Value

{Query: “Order details of order_104”}
OR
{Query: “Order details of order_106”}

{Key, Value store}

{"order_100": {"items": "a1", "delivery_date": "a2", ...}},
{"order_101": {"items": "b1", "delivery_date": "b2", ...}},
{"order_102": {"items": "c1", "delivery_date": "c2", ...}},
{"order_103": {"items": "d1", "delivery_date": "d2", ...}},
{"order_104": {"items": "e1", "delivery_date": "e2", ...}},
{"order_105": {"items": "f1", "delivery_date": "f2", ...}},
{"order_106": {"items": "g1", "delivery_date": "g2", ...}},
{"order_107": {"items": "h1", "delivery_date": "h2", ...}},
{"order_108": {"items": "i1", "delivery_date": "i2", ...}},
{"order_109": {"items": "j1", "delivery_date": "j2", ...}},
{"order_110": {"items": "k1", "delivery_date": "k2", ...}}
Query, Key & Value

{Query: “Order details of order_104”}

OR

{Query: “Order details of order_106”}

{Key, Value store}

```json
{"order_100": {"items": "a1", "delivery_date": "a2", "\ldots"},
"order_101": {"items": "b1", "delivery_date": "b2", "\ldots"},
"order_102": {"items": "c1", "delivery_date": "c2", "\ldots"},
"order_103": {"items": "d1", "delivery_date": "d2", "\ldots"},
"order_104": {"items": "e1", "delivery_date": "e2", "\ldots"},
"order_105": {"items": "f1", "delivery_date": "f2", "\ldots"},
"order_106": {"items": "g1", "delivery_date": "g2", "\ldots"},
"order_107": {"items": "h1", "delivery_date": "h2", "\ldots"},
"order_108": {"items": "i1", "delivery_date": "i2", "\ldots"},
"order_109": {"items": "j1", "delivery_date": "j2", "\ldots"},
"order_110": {"items": "k1", "delivery_date": "k2", "\ldots"}}
```
Query, Key & Value

Query: “Order details of order_104”
OR
Query: “Order details of order_106”

{Key, Value store}

```
{"order_100": {"items": "a1", "delivery_date": "a2", ...}},
{"order_101": {"items": "b1", "delivery_date": "b2", ...}},
{"order_102": {"items": "c1", "delivery_date": "c2", ...}},
{"order_103": {"items": "d1", "delivery_date": "d2", ...}},
{"order_104": {"items": "e1", "delivery_date": "e2", ...}},
{"order_105": {"items": "f1", "delivery_date": "f2", ...}},
{"order_106": {"items": "g1", "delivery_date": "g2", ...}},
{"order_107": {"items": "h1", "delivery_date": "h2", ...}},
{"order_108": {"items": "i1", "delivery_date": "i2", ...}},
{"order_109": {"items": "j1", "delivery_date": "j2", ...}},
{"order_110": {"items": "k1", "delivery_date": "k2", ...}}
```
Query, Key & Value

Query
1. Search for info

Key
1. Interacts directly with Queries
2. Distinguishes one object from another
3. Identify which object is the most relevant and by how much

Value
1. Actual details of the object
2. More fine grained

{Query: “Order details of order_104”}
OR
{Query: “Order details of order_106”}

{"order_100": {"items":"a1", "delivery_date":"a2", ...},
"order_101": {"items":"b1", "delivery_date":"b2", ...},
"order_102": {"items":"c1", "delivery_date":"c2", ...},
"order_103": {"items":"d1", "delivery_date":"d2", ...},
"order_104": {"items":"e1", "delivery_date":"e2", ...},
"order_105": {"items":"f1", "delivery_date":"f2", ...},
"order_106": {"items":"g1", "delivery_date":"g2", ...},
"order_107": {"items":"h1", "delivery_date":"h2", ...},
"order_108": {"items":"i1", "delivery_date":"i2", ...},
"order_109": {"items":"j1", "delivery_date":"j2", ...},
"order_110": {"items":"k1", "delivery_date":"k2", ...}}
Attention

Query

Key Value Store

Key

Value
Attention

Query

Key Value Store

Key

Value
Attention

Done at the same time!!
Attention

Parallelizable !!!

Query

Key Value Store

Key

Value

\( Q \)

\( QK^T \)

\( \text{softmax}(\frac{QK^T}{\sqrt{d}}) \)

\( \text{softmax}(\frac{QK^T}{\sqrt{d}})V \)
Attention

Parallelizable !!!

Query

Key Value Store

Key

Value

\[ Q \]

\[ QK^T \]

\[ \text{softmax}(\frac{QK^T}{\sqrt{d}}) \]

\[ \text{softmax}(\frac{QK^T}{\sqrt{d}})V \]
Attention

I ate an apple <eos>
Attention

Dimensions across QKV have been dropped for brevity.
I ate an apple <eos>

Dimensions across QKV have been dropped for brevity
Attention

Dimensions across QKV have been dropped for brevity
Attention

Dimensions across QKV have been dropped for brevity

\[ \alpha_{1,1} \]

\[ e_{1,1} \]

\[ \begin{align*}
Q_1 & \quad K_1 & \quad V_1 \\
W_Q & \quad W_K & \quad W_V \\
I_1 & \quad & \\
\end{align*} \]

\[ \begin{align*}
Q_2 & \quad K_2 & \quad V_2 \\
W_Q & \quad W_K & \quad W_V \\
I_2 & \quad & \\
\end{align*} \]

\[ \begin{align*}
Q_3 & \quad K_3 & \quad V_3 \\
W_Q & \quad W_K & \quad W_V \\
I_3 & \quad & \\
\end{align*} \]

\[ \begin{align*}
Q_4 & \quad K_4 & \quad V_4 \\
W_Q & \quad W_K & \quad W_V \\
I_4 & \quad & \\
\end{align*} \]

\[ \begin{align*}
Q_5 & \quad K_5 & \quad V_5 \\
W_Q & \quad W_K & \quad W_V \\
I_5 & \quad & \langle \text{eos} \rangle \\
\end{align*} \]
Attention

Dimensions across QKV have been dropped for brevity.

\[
\begin{align*}
\alpha_{1,1} & \times \quad e_{1,1} \\
\alpha_{1,2} & \times \quad e_{1,2}
\end{align*}
\]

\[
\begin{align*}
Q_1 & \times K_1 \times V_1 \\
Q_2 & \times K_2 \times V_2 \\
Q_3 & \times K_3 \times V_3 \\
Q_4 & \times K_4 \times V_4 \\
Q_5 & \times K_5 \times V_5
\end{align*}
\]

\[
\begin{align*}
W_Q & \times W_K \times W_V \\
W_Q & \times W_K \times W_V \\
W_Q & \times W_K \times W_V \\
W_Q & \times W_K \times W_V \\
W_Q & \times W_K \times W_V
\end{align*}
\]

\[
\begin{align*}
I_1 \\
I_2 \\
I_3 \\
I_4 \\
I_5
\end{align*}
\]

I ate an apple <eos>
Attention

Dimensions across QKV have been dropped for brevity

$\alpha_{1,1} \times e_{1,1}$

$\alpha_{1,2} \times e_{1,2}$

$\alpha_{1,3} \times e_{1,3}$

$\text{softmax}$
Attention

Dimensions across QKV have been dropped for brevity

I ate an apple <eos>
Attention

Dimensions across QKV have been dropped for brevity.
Dimensions across QKV have been dropped for brevity.

Contextually rich embedding

Attention

\[ \alpha_{1,1} \otimes e_{1,1} \rightarrow \text{softmax} \]

\[ \alpha_{1,2} \otimes e_{1,2} \rightarrow \text{softmax} \]

\[ \alpha_{1,3} \otimes e_{1,3} \rightarrow \text{softmax} \]

\[ \alpha_{1,4} \otimes e_{1,4} \rightarrow \text{softmax} \]

\[ \alpha_{1,5} \otimes e_{1,5} \rightarrow \text{softmax} \]

\[ Z_1 \]
Attention

Dimensions across QKV have been dropped for brevity

I ate an apple <eos>
Attention

Dimensions across QKV have been dropped for brevity

Contextually rich embedding

\[
\begin{align*}
\alpha_{1,1} \otimes W_Q & \rightarrow e_{1,1} \\
\alpha_{1,2} \otimes W_Q & \rightarrow e_{1,2} \\
\alpha_{1,3} \otimes W_Q & \rightarrow e_{1,3} \\
\alpha_{1,4} \otimes W_Q & \rightarrow e_{1,4} \\
\alpha_{1,5} \otimes W_Q & \rightarrow e_{1,5} \\
\sum & \rightarrow Z_1
\end{align*}
\]
I ate an apple <eos>
Poll 1 @1296

Which of the following are true about attention? (Select all that apply)

a. To calculate attention weights for input $I_2$, you would use key $k_2$, and all queries
b. To calculate attention weights for input $I_2$, you would use query $q_2$, and all keys
c. We scale the $QK^T$ product to bring attention weights in the range of $[0,1]$
d. We scale the $QK^T$ product to allow for numerical stability
Which of the following are true about attention? (Select all that apply)

a. To calculate attention weights for input $I_2$, you would use key $k_2$, and all queries

b. To calculate attention weights for input $I_2$, you would use query $q_2$, and all keys

c. We scale the $QK^T$ product to bring attention weights in the range of $[0,1]$

d. We scale the $QK^T$ product to allow for numerical stability
Positional Encoding

I ate an apple <eos>
I ate an apple <eos>

apple ate an I <eos>

Positional Encoding
Positional Encoding

Requirements for Positional Encodings

• Some representation of time? (like seq2seq?)
• Should be unique for each position – not cyclic
Positional Encoding

Requirements for Positional Encodings

• Some representation of time? (like seq2seq?)
• Should be unique for each position – not cyclic

Possible Candidates:

\[ P_{t+1} = P_t + \Delta c \]
\[ P_{t+1} = e^{P_t \Delta} c \]
\[ P_{t+1} = P_t \cdot t \Delta c \]
Requirements for Positional Encodings

- Some representation of time? (like seq2seq?)
- Should be unique for each position – not cyclic

Possible Candidates:

\[ P_{t+1} = P_t + \Delta c \]
\[ P_{t+1} = e^{P_{t\Delta}c} \]
\[ P_{t+1} = P_t \cdot t^{\Delta c} \]
Positional Encoding

Requirements for Positional Encodings

• Some representation of time? (like seq2seq?)
• Should be unique for each position – not cyclic
• Bounded

Possible Candidates:

- \( P_{t+1} = P_t + \Delta c \)
- \( P_{t+1} = e^{P_t \Delta c} \)
- \( P_{t+1} = P_t \cdot t \Delta c \)
Positional Encoding

Requirements for Positional Encodings

• Some representation of time? (like seq2seq?)
• Should be unique for each position – not cyclic
• Bounded

Possible Candidates:

\[ P(t + t') = M^t \times P(t) \]
Positional Encoding

Requirements for Positional Encodings

- Some representation of time? (like seq2seq?)
- Should be unique for each position – not cyclic
- Bounded

Possible Candidates:

\[ P(t + t') = M^{t'} \times P(t) \]

\( M \)?

1. Should be a unitary matrix
2. Magnitudes of eigen value should be 1 -> norm preserving

Positional Encoding
Requirements for Positional Encodings

- Some representation of time? (like seq2seq?)
- Should be unique for each position – not cyclic
- Bounded

Possible Candidates:

\[ P(t + t') = M' \times P(t) \]

\[ M \]

1. The matrix can be learnt
2. Produces unique rotated embeddings each time
Rotary Positional Embedding

**RoFormer: Enhanced Transformer with Rotary Position Embedding**

\[
f_{\{q,k\}}(x_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} x_m^{(1)} \\ x_m^{(2)} \end{pmatrix}
\]

Table 2: Comparing RoFormer and BERT by fine tuning on downstream GLEU tasks.

<table>
<thead>
<tr>
<th>Model</th>
<th>MRPC</th>
<th>SST-2</th>
<th>QNLI</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI(m/mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERTDevlin et al. [2019]</td>
<td>88.9</td>
<td>93.5</td>
<td>90.5</td>
<td>85.8</td>
<td>71.2</td>
<td>84.6/83.4</td>
</tr>
<tr>
<td>RoFormer</td>
<td><strong>89.5</strong></td>
<td>90.7</td>
<td>88.0</td>
<td><strong>87.0</strong></td>
<td><strong>86.4</strong></td>
<td><strong>80.2/79.8</strong></td>
</tr>
</tbody>
</table>

REF: Rotary Positional Embeddings
Positional Encoding

Requirements for Positional Encodings

- Some representation of time? (like seq2seq?)
- Should be unique for each position – **not cyclic**
- Bounded

Actual Candidates:

\[ \text{sine}(g(t)) \]
\[ \text{cosine}(g(t)) \]
Positional Encoding

Requirements for $g(t)$

- Must have same dimensions as input embeddings
- Must produce overall unique encodings

$\text{pos} \rightarrow \text{idx}$ of the token in input sentence

$i \rightarrow i^{\text{th}}$ dimension out of $d$

$$PE_{(\text{pos}, 2i)} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

$$PE_{(\text{pos}, 2i+1)} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$
Positional Encoding

Requirements for $g(t)$

- Must have same dimensions as input embeddings
- Must produce overall unique encodings

$\text{pos} \rightarrow \text{idx of the token in input sentence}$

$i \rightarrow i^{\text{th}} \text{ dimension out of } d$

Positional Encoding:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim 1</td>
<td>0.000</td>
<td>0.841</td>
<td>0.909</td>
<td>0.141</td>
<td>-0.757</td>
</tr>
<tr>
<td>Dim 2</td>
<td>1.000</td>
<td>0.540</td>
<td>-0.416</td>
<td>-0.990</td>
<td>-0.654</td>
</tr>
<tr>
<td>Dim 3</td>
<td>0.000</td>
<td>0.025</td>
<td>0.050</td>
<td>0.075</td>
<td>0.100</td>
</tr>
<tr>
<td>Dim 4</td>
<td>1.000</td>
<td>1.000</td>
<td>0.999</td>
<td>0.997</td>
<td>0.995</td>
</tr>
<tr>
<td>Dim 5</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
<td>0.002</td>
<td>0.003</td>
</tr>
</tbody>
</table>

$PE_{(\text{pos},2i)} = \sin(\text{pos}/10000^{2i/d_{\text{total}}})$

$PE_{(\text{pos},2i+1)} = \cos(\text{pos}/10000^{2i/d_{\text{total}}})$
Positional Encoding

I ate an apple

Embedding Layer

Tokenizer

I ate an apple

Input

Final Input Embeddings

Position Encodings

Input Embeddings

Tokens

<eos>

an

apple
I ate an apple.
Self Attention

From lecture 18:

\[ \text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \]
The animal didn’t cross the street because it was too wide.
The animal didn't cross the street because it was too wide.

Coreference resolution?
Self Attention

The animal didn't cross the street because it was too wide.
Self Attention
Self Attention
Self Attention

Query Inputs = Key Inputs = Value Inputs
**Self Attention**

\[ R^{d_{model} \times d_{model}} \]

\[
\begin{align*}
W_Q & \\
W_K & \\
W_v & \\
\end{align*}
\]

Input Embeddings
Self Attention

Input Embeddings

$R^{T \times d_{model}}$

$R^{d_{model} \times d_{model}}$

$R^{d_{model} \times d_{model}}$

$R^{T \times d_{model}}$

Input Embeddings

W_Q

Q Projections

W_K

K Projections

W_V

V Projections

$R^{T \times d_{model}}$

Add & Norm

Feed Forward

$R^{T \times d_{model}}$

Add & Norm

Multi-Head Attention

Positional Encoding

Input Embedding

Inputs
Self Attention

\[ \sqrt{d_{\text{model}}} \]

\[ R^{T \times d_{\text{model}}} \times R^{d_{\text{model}} \times T} \]

softmax

Q_{\text{Projection}} \times K_{\text{Projection}}
Softmax

$\Theta( T^2 \times d_{\text{model}})$

$\sqrt{d_{\text{model}}}$

$R^{T \times T}$

$R^{T \times d_{\text{model}}}$

$R^{d_{\text{model}} \times T}$

Self Attention

QProjection

KProjection

Feed Forward

Add & Norm

Multi-Head Attention

Input Embedding

Positional Encoding

Add & Norm

N\times
Self Attention

$\theta(T^2 \times d_{\text{model}})$

softmax

$\sqrt{d_{\text{model}}}$

$R^{T \times T}$

$R^{T \times d_{\text{model}}}$

$R^{d_{\text{model}} \times T}$

$R^{T \times d_{\text{model}}}$

$Q_{\text{Projection}}$  $K_{\text{Projection}}$  $V_{\text{Projection}}$

$\text{Add & Norm}$  

$\text{Feed Forward}$

$\text{Multi-Head Attention}$

$\text{Positional Encoding}$

$\text{Input Embedding}$

Inputs
Self Attention

Attention: Z
The animal didn't cross the street because it was too wide.
The animal didn't cross the street because it was too wide.

Sentence boundaries?

coreference resolution

Context?

Semantic relationships?

Part of Speech?

Comparisons?
Self Attention

Input Embeddings

$R^{d_{model} \times d_{model}}$

$W_Q$

$W_K$

$W_v$

$R^{T \times d_{model}}$
Multi-Head Attention

\[ W_{Q1}, W_{Q2}, \ldots, W_{QH}, \]

\[ W_{K1}, W_{K2}, \ldots, W_{KH}, \]

\[ W_{V1}, W_{V2}, \ldots, W_{VH}, \]

\[ d_h = \frac{d_{model}}{h} \]
Multi-Head Attention
Multi-Head Attention

softmax

$R^{T \times d_h}$  $R^{d_h \times T}$  $R^{T \times d_h}$

$Q_i$  $K_i$  $V_i$

$\sqrt{d_{model}}$

$R^{T \times T}$

for all $i \in [1, h]$
Multi-Head Attention

\[ R^{T \times d_h} \]

\[ R^{T \times d_h} \]

\[ R^{T \times d_h} \]

\[ R^{T \times d_h} \]

\[ Z_1 \]

\[ Z_2 \]

\[ \ldots \]

\[ Z_h \]

CONCAT

Multi Head Attention : Z

\[ d_h = \frac{d_{\text{model}}}{h} \]
The animal didn’t cross the street because it was too wide.
Normalization(Z)
- Mean 0, Std dev 1
- Stabilizes training
- Regularization effect

Add & Norm

Add -> Residuals
- Avoid vanishing gradients
- Train deeper networks
Feed Forward

- Non Linearity
- Complex Relationships
- Learn from each other

Feed Forward

Input

Residuals

Norm(Z)
**Add & Norm**

- **Add & Norm**
- **Feed Forward**
- **Residuals**
- **Input**
- **Norm(Z)**

![Graph of Nonlinearities](image_url)

- Add & Norm
- Feed Forward
- Residuals
- Input
- Norm(Z)
Encoders

Encoder

ENCODER
Encoders

Encoder

ENCODER

.  
.
.

ENCODER

ENCODER

Input to Encoder_{i+1}

Output from Encoder_{i}
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Residuals
- Query
- Key
- Value
- Add & Norm
- Encoder
  - Decoder
- Attention
- Self Attention
- Multi Head Attention
  - Masked Attention
  - Encoder Decoder Attention
  - Output Probabilities / Logits
  - Softmax
  - Encoder-Decoder models
  - Decoder only models
Machine Translation

Targets
Ich have einen apfel gegessen

Inputs
I ate an apple
Ich have einen apfel gegessen
Ich habe einen Apfel gegessen

Generate Target Embeddings
Ich habe einen Apfel gegessen <eos>
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Parallelized?
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Training

<sos> Ich have einen apfel gegessen <eos>
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Training

Outputs at time T should only pay attention to outputs until time T-1
### Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

<table>
<thead>
<tr>
<th></th>
<th>Ich</th>
<th>have</th>
<th>einen</th>
<th>apfel</th>
<th>gegessen</th>
<th>&lt;eos&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;sos&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>have</td>
<td>einen</td>
<td>apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>3</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>have</td>
<td>einen</td>
<td>apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>4</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>have</td>
<td>einen</td>
<td>apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>5</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>have</td>
<td>einen</td>
<td>apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>6</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>have</td>
<td>einen</td>
<td>apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>7</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>have</td>
<td>einen</td>
<td>apfel</td>
<td>gegessen</td>
</tr>
</tbody>
</table>
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)
2. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)
3. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)
4. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)
5. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)
6. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)
7. \( <\text{sos}> \) Ich have einen apfel gegessen \( <\text{eos}> \)

Mask the available attention values?
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1. <sos>
2. <sos> Ich
3. <sos> Ich have
4. <sos> Ich have einen
5. <sos> Ich have einen apfel
6. <sos> Ich have einen apfel gegessen
7. <sos> Ich have einen apfel gegessen <eos>

Softmax -> -∞ -> 0
Masked Multi Head Attention

\[ R^{T \times T} \]

\[ QK^T \]

Attention Mask: \( M \)

\[ = \]

Masked Attention

Masked Multi Head Attention: \( Z' \)
Masked Multi Head Attention

Masked Multi Head Attention : $Z'$
Encoder Decoder Attention

Encoder Decoder Attention?

Add & Norm

Input

Norm(Z')
Encoder Decoder Attention
Encoder Decoder Attention

**Encoder Self Attention**
1. Queries from Encoder Inputs
2. Keys from Encoder Inputs
3. Values from Encoder Inputs

**Decoder Masked Self Attention**
1. Queries from Decoder Inputs
2. Keys from Decoder Inputs
3. Values from Decoder Inputs
Attention

{Query: “Order details of order_104”}

{Query: “Order details of order_106”}

{Key, Value store}
Encoder Decoder Attention

Encoder

Keys from **Encoder Outputs**
Values from **Encoder Outputs**

Decoder

Queries from **Decoder Inputs**

NOTE: Every decoder block receives the same FINAL encoder output
Encoder Decoder Attention

\[ R^{T_d \times d_{\text{model}}} \]

\[ R^{T_d \times T_e} \]

\[ \text{softmax}(\frac{Q_d K_e^T}{\sqrt{d_{\text{model}}}}) \]

\[ V_e R^{T_e \times d_{\text{model}}} \]

\[ R^{T_d \times T_e} \]

\[ \text{softmax}(\frac{Q_d K_e^T}{\sqrt{d_{\text{model}}}}) \]

\[ Q_d \quad K_e \]

\[ R^{T_e \times d_{\text{model}}} \]
Encoder Decoder Attention

- Non Linearity
- Complex Relationships
- Learn from each other

Add n Norm Decoder Self Attn

Norm(Z’’)

Feed Forward

Residuals

Nonlinearities

-2 -1 0 1 2

0.0 0.5 1.0 1.5 2.0 2.5

Nonlinearities

ReLU

GELU

Addition

Output Probabilities

Softmax

Linear

Positional Encoding

Output Embedding

Inputs

Outputs

(shifted right)

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Input Embedding
Decoder
Decoder

Decoder output

$R_{T_d \times d_{model}}$
Linear weights are often tied with input embedding matrix.
Softmax

Output Probabilities

$R^{T_d \times V}$
Poll 2 (@1297)

Which of the following are true about transformers?

a. Transformers can always be run in parallel
b. Transformer decoders can only be parallelized during training
c. Positional encodings help parallelize the transformer encoder
d. Queries, keys, and values are obtained by splitting the input into 3 equal segments
e. Multiheaded attention helps transformers find different kinds of relations between the tokens
f. During decoding, decoder outputs function as queries and keys while the values come from the encoder
Which of the following are true about transformers?

a. Transformers can always be run in parallel
b. Transformer decoders can only be parallelized during training
c. Positional encodings help parallelize the transformer encoder
d. Queries, keys, and values are obtained by splitting the input into 3 equal segments
e. Multiheaded attention helps transformers find different kinds of relations between the tokens
f. During decoding, decoder outputs function as queries and keys while the values come from the encoder
Transformers

Targets
Ich have einen apfel gegessen

Inputs
I ate an apple

Machine Translation
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Residuals
- Query
- Key
- Value
- Add & Norm
- Encoder
- Decoder

- Attention
- Self Attention
- Multi Head Attention
- Masked Attention
- Encoder Decoder Attention
- Output Probabilities / Logits
- Softmax

- Encoder-Decoder models
- Decoder only models
Part 2

LLM
Transformers, mid-2017

Diagram of Transformer architecture with layers and connections.
Transformers, mid-2017

Representation
Transformers, mid-2017

Representation

Generation
Transformers, mid-2017

**Input** – input tokens

**Output** – hidden states

**Input** – output tokens and hidden states*

**Output** – output tokens

---

* Output tokens and hidden states are produced as output, indicating a causal or autoregressive model where hidden state from input tokens is used to generate output tokens.
Transformers, mid-2017

Input – input tokens
Output – hidden states

Model can see all timesteps

Input – output tokens and hidden states*
Output – output tokens

Model can only see previous timesteps

Representation

Generation
Transformers, mid-2017

Input – input tokens
Output – hidden states

Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

Representation

Input – output tokens and hidden states*
Output – output tokens

Model can only see previous timesteps

Model is auto-regressive with previous timesteps’ outputs

Generation
Transformers, mid-2017

**Input** – input tokens

**Output** – hidden states

Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

*Can also be adapted to generate tokens by appending a module that maps hidden state dimensionality to vocab size*

**Representation**

**Input** – output tokens and hidden states*

**Output** – output tokens

Model can only see previous timesteps

Model is auto-regressive with previous timesteps’ outputs

*Can also be adapted to generate hidden states by looking before token outputs*

**Generation**
2018 – The Inception of the LLM Era

BERT
Oct 2018

Representation

GPT
Jun 2018

Generation
• One of the biggest challenges in LM-building used to be the lack of task-specific training data.

• What if we learn an effective representation that can be applied to a variety of downstream tasks?
  • Word2vec (2013)
  • GloVe (2014)
BERT - Bidirectional Encoder Representations

BERT Pre-Training Corpus:
• English Wikipedia - 2,500 million words
• Book Corpus - 800 million words
BERT - Bidirectional Encoder Representations

BERT Pre-Training Corpus:
- English Wikipedia - 2,500 million words
- Book Corpus - 800 million words

BERT Pre-Training Tasks:
- MLM (Masked Language Modeling)
- NSP (Next Sentence Prediction)
BERT Pre-Training Corpus:
• English Wikipedia - 2,500 million words
• Book Corpus - 800 million words

BERT Pre-Training Tasks:
• MLM (Masked Language Modeling)
• NSP (Next Sentence Prediction)

BERT Pre-Training Results:
• BERT-Base – 110M Params
• BERT-Large – 340M Params
BERT - Bidirectional Encoder Representations

MLM (Masked Language Modeling)

How are <MASK> doing today <SEP>

BERT

<CLS> How are <MASK> doing today <SEP>
BERT - Bidirectional Encoder Representations

MLM (Masked Language Modeling)

Prediction head

is_next 95%
not_next 5%

<CLS> ... ... <SEP> ... ... <SEP>

<CLS> ... ... <SEP> ... ... <SEP>

<i>is_next</i> not_next 95% 5%

<CLS> ... ... <SEP> ... ... <SEP>

<i>Prediction head</i>

<CLS> ... ... <SEP> ... ... <SEP>

<i>Output Probabilities</i>

Softmax

Linear

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

N×

N×

Positional Encoding

Input Embedding

Inputs

Outputs (shifted right)
BERT Fine-Tuning:

- Simply add a task-specific module after the last encoder layer to map it to the desired dimension.
  
  - **Classification Tasks:**
    - Add a feed-forward layer on top of the encoder output for the [CLS] token
  
  - **Question Answering Tasks:**
    - Train two extra vectors to mark the beginning and end of answer from paragraph
  
- ...
BERT Evaluation:

• General Language Understanding Evaluation (GLUE)
  • Sentence pair tasks
  • Single sentence classification

• Stanford Question Answering Dataset (SQuAD)
## BERT - Bidirectional Encoder Representations

### BERT Evaluation:

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.8</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>87.4</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.1</td>
</tr>
<tr>
<td><strong>BERT_BASE</strong></td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.5</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td><strong>BERT_LARGE</strong></td>
<td><strong>86.7/85.9</strong></td>
<td><strong>72.1</strong></td>
<td><strong>92.7</strong></td>
<td><strong>94.9</strong></td>
<td><strong>60.5</strong></td>
<td><strong>86.5</strong></td>
<td><strong>89.3</strong></td>
<td><strong>70.1</strong></td>
<td><strong>82.1</strong></td>
</tr>
</tbody>
</table>

### Leaderboard (Oct 8th, 2018)

<table>
<thead>
<tr>
<th>System</th>
<th>Dev EM</th>
<th>F1</th>
<th>Test EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>82.3</td>
<td>91.2</td>
</tr>
<tr>
<td>#1 Ensemble - nlent</td>
<td>-</td>
<td>-</td>
<td>86.0</td>
<td>91.7</td>
</tr>
<tr>
<td>#2 Ensemble - QANet</td>
<td>-</td>
<td>-</td>
<td>84.5</td>
<td>90.5</td>
</tr>
<tr>
<td>#1 Single - nlent</td>
<td>-</td>
<td>-</td>
<td>83.5</td>
<td>90.1</td>
</tr>
<tr>
<td>#2 Single - QANet</td>
<td>-</td>
<td>-</td>
<td>82.5</td>
<td>89.3</td>
</tr>
</tbody>
</table>

**Published**

<table>
<thead>
<tr>
<th>System</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiDAF+ELMo (Single)</td>
<td>-</td>
<td>85.8</td>
</tr>
<tr>
<td>R.M. Reader (Single)</td>
<td>78.9</td>
<td>86.3</td>
</tr>
<tr>
<td>R.M. Reader (Ensemble)</td>
<td>81.2</td>
<td>87.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ours</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_BASE (Single)</td>
<td>80.8</td>
</tr>
<tr>
<td>BERT_LARGE (Single)</td>
<td>84.1</td>
</tr>
<tr>
<td>BERT_LARGE (Ensemble)</td>
<td>85.8</td>
</tr>
<tr>
<td>BERT_LARGE (Sgl.+TrivaQA)</td>
<td>84.2</td>
</tr>
<tr>
<td>BERT_LARGE (Ens.+TriviaQA)</td>
<td>86.2</td>
</tr>
</tbody>
</table>

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.
What is our takeaway from BERT?

• Pre-training tasks can be invented flexibly...
  • Effective representations can be derived from a flexible regime of pre-training tasks.
What is our takeaway from BERT?

• Pre-training tasks can be invented flexibly...
  • Effective representations can be derived from a flexible regime of pre-training tasks.

• Different NLP tasks seem to be highly transferable with each other...
  • As long as we have effective representations, that seems to form a general model which can serve as the backbone for many specialized models.
What is our takeaway from BERT?

• Pre-training tasks can be invented flexibly...
  • Effective representations can be derived from a flexible regime of pre-training tasks.

• Different NLP tasks seem to be highly transferable with each other...
  • As long as we have effective representations, that seems to form a general model which can serve as the backbone for many specialized models.

• And scaling works!!!
  • 340M is considered large in 2018
2018 – The Inception of the LLM Era

Representation

BERT
Oct 2018

Generation

GPT
Jun 2018
GPT – Generative Pretrained Transformer

• Similarly motivated as BERT, though differently designed

• Can we leverage large amounts of unlabeled data to pretrain an LM that understands general patterns?
GPT – Generative Pretrained Transformer

GPT Pre-Training Corpus:
• Similarly, BooksCorpus and English Wikipedia

GPT Pre-Training Tasks:
• Predict the next token, given the previous tokens
  • More learning signals than MLM

GPT Pre-Training Results:
• GPT – 117M Params
  • Similarly competitive on GLUE and SQuAD
GPT – Generative Pretrained Transformer

GPT Fine-Tuning:
• Prompt-format task-specific text as a continuous stream for the model to fit

Summarization

Summarize this article:

The summary is:

Answer the question based on the context.

Context:

Question:

Answer:

QA
What is our takeaway from GPT?

• The Effectiveness of Self-Supervised Learning
  • Specifically, the model seems to be able to learn from generating the language *itself*, rather than from any specific task we might cook up.
What is our takeaway from GPT?

• **The Effectiveness of Self-Supervised Learning**
  • Specifically, the model seems to be able to learn from generating the language *itself*, rather than from any specific task we might cook up.

• **Language Model as a Knowledge Base**
  • Specifically, a generatively pretrained model seems to have a decent zero-shot performance on a range of NLP tasks.
What is our takeaway from GPT?

• **The Effectiveness of Self-Supervised Learning**
  • Specifically, the model seems to be able to learn from generating the language *itself*, rather than from any specific task we might cook up.

• **Language Model as a Knowledge Base**
  • Specifically, a generatively pretrained model seems to have a decent zero-shot performance on a range of NLP tasks.

• **And scaling works!!!**
The original GPT’s parameter count is closest to...

A. 117
B. 117K
C. 117M
D. 117B
The original GPT’s parameter count is closest to...

A. 117
B. 117K
C. 117M
D. 117B
The LLM Era – Paradigm Shift in Machine Learning

BERT
Oct 2018

GPT
Jun 2018

Representation

Generation
The LLM Era – Paradigm Shift in Machine Learning

**Representation**

- BERT – 2018
- DistilBERT – 2019
- RoBERTa – 2019
- ALBERT – 2019
- ELECTRA – 2020
- DeBERTa – 2020
- ...

**Generation**

- GPT – 2018
- GPT-2 – 2019
- GPT-3 – 2020
- GPT-Neo – 2021
- GPT-3.5 (ChatGPT) – 2022
- LLaMA – 2023
- GPT-4 – 2023
- ...

T5 – 2019
BART – 2019
mT5 – 2021
...
The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that...

• Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

<table>
<thead>
<tr>
<th>Before LLMs</th>
<th>Since LLMs</th>
</tr>
</thead>
</table>

• Feature Engineering
  • How do we design or select the best features for a task?
From both BERT and GPT, we learn that...

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

<table>
<thead>
<tr>
<th>Before LLMs</th>
<th>Since LLMs</th>
</tr>
</thead>
</table>

- **Feature Engineering**
  - How do we design or select the best features for a task?

- **Model Selection**
  - Which model is best for which type of task?
The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that...
• Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

<table>
<thead>
<tr>
<th>Before LLMs</th>
<th>Since LLMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Engineering</td>
<td>Model Selection</td>
</tr>
<tr>
<td>• How do we design or select the best features for a task?</td>
<td>• Which model is best for which type of task?</td>
</tr>
<tr>
<td>• Transfer Learning</td>
<td></td>
</tr>
<tr>
<td>• Given scarce labeled data, how do we transfer knowledge from other domains?</td>
<td></td>
</tr>
</tbody>
</table>
The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that...

• Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

  Before LLMs                              Since LLMs

• Feature Engineering
  • How do we design or select the best features for a task?

• Model Selection
  • Which model is best for which type of task?

• Transfer Learning
  • Given scarce labeled data, how do we transfer knowledge from other domains?

• Overfitting vs Generalization
  • How do we balance complexity and capacity to prevent overfitting while maintaining good performance?
From both BERT and GPT, we learn that...

• Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

• Feature Engineering
  • How do we design or select the best features for a task?

• Model Selection
  • Which model is best for which type of task?

• Transfer Learning
  • Given scarce labeled data, how do we transfer knowledge from other domains?

• Overfitting vs Generalization
  • How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

• Pre-training and Fine-tuning
  • How do we leverage large scales of unlabeled data out there previously under-leveraged?
The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that...
- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

### Before LLMs
- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

### Since LLMs
- **Pre-training and Fine-tuning**
  - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
  - How can we make models perform on tasks they are not trained on?
The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that...

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
  - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
  - How can we make models perform on tasks they are not trained on?
- **Prompting**
  - How do we make models understand their task simply by describing it in natural language?
The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that...

• Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

• **Feature Engineering**
  • How do we design or select the best features for a task?

• **Model Selection**
  • Which model is best for which type of task?

• **Transfer Learning**
  • Given scarce labeled data, how do we transfer knowledge from other domains?

• **Overfitting vs Generalization**
  • How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

• **Pre-training and Fine-tuning**
  • How do we leverage large scales of unlabeled data out there previously under-leveraged?

• **Zero-shot and Few-shot learning**
  • How can we make models perform on tasks they are *not* trained on?

• **Prompting**
  • How do we make models understand their task simply by describing it in natural language?

• **Interpretability and Explainability**
  • How can we *understand* the inner workings of our own models?
The LLM Era – Paradigm Shift in Machine Learning

• What has caused this paradigm shift?
The LLM Era – Paradigm Shift in Machine Learning

• What has caused this paradigm shift?

  • Problem in recurrent networks
    • Information is effectively lost during encoding of long sequences
    • Sequential nature disables parallel training and favors late timestep inputs
The LLM Era – Paradigm Shift in Machine Learning

• What has caused this paradigm shift?
  • Problem in recurrent networks
    • Information is effectively lost during encoding of long sequences
    • Sequential nature disables parallel training and favors late timestep inputs
  • Solution: Attention mechanism
    • Handling long-range dependencies
    • Parallel training
    • Dynamic attention weights based on inputs
The LLM Era – Paradigm Shift in Machine Learning

• Attention and Transformer – is this the end?
The LLM Era – Paradigm Shift in Machine Learning

• Attention and Transformer – is this the end?
  • Problem in current Transformer-based LLMs??
Poll
Piazza @1292

What might be a flaw of our current Transformer-based LLMs?

Freeform response
The LLM Era – Paradigm Shift in Machine Learning

• Attention and Transformer – is this the end?

• Problem in current Transformer-based LLMs??
  • True understanding the material vs. memorization and pattern-matching
  • Cannot reliably follow rules – factual hallucination e.g. inability in arithmetic
The LLM Era – Paradigm Shift in Machine Learning

• Attention and Transformer – is this the end?

• Problem in current Transformer-based LLMs??
  • True understanding the material vs. memorization and pattern-matching
  • Cannot reliably follow rules – factual hallucination e.g. inability in arithmetic

• Solution: ???
Looking Back

It is true that language models are just programmed to predict the next token. But that is not as simple as you might think.

In fact, all animals, including us, are just programmed to survive and reproduce, and yet amazingly complex and beautiful stuff comes from it.

- Sam Altman*

*Paraphrased by IDL TAs