Introduction to Deep Learning

Lecture 19
Transformers

Liangze Li           Kateryna Shapovalenko

11-785, Spring 2024
Attendance poll @1585
Table of contents

1. The Transformer Architecture
2. Pre-training and Fine-tuning
3. Transformer Applications
4. Case study - Large Language Models
Part 1

Transformer Architecture
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders

- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders
- Masked Attention
- Encoder Decoder Attention
- Linear
- Encoder-Decoder Models
Machine Translation

**Inputs**
I ate an apple

**Targets**
Ich habe einen Apfel gegessen
I ate an apple
Tokenization

Tokenizer (split into individual words)

I ate an apple
I ate an apple

Tokenizer (split into individual words)

I ate an apple
Input Embeddings

I ate an apple

Tokenizer (split into individual words)

I ate an apple

Generate Input Embeddings
I ate an apple

Generate Input Embeddings
Position Encodings

I ate an apple <eos>
Position Encodings

I ate an apple <eos>

apple ate an I <eos>
Position Encodings

Requirements for Positional Encodings???
Position Encodings

Requirements for Positional Encodings

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

Requirements for Positional Encodings

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

Possible Candidates:

\[ P_{t+1} = P_t + \Delta c \]
\[ P_{t+1} = e^{P_t \Delta c} \]
\[ P_{t+1} = P_t^{t \Delta c} \]
Position Encodings

Requirements for Positional Encodings

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

Possible Candidates:

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

Requirements for Positional Encodings

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

Possible Candidates:

\[ P_{t+1} = P_t + \Delta c \]
\[ P_{t+1} = P_t \cdot c \]
\[ P_{t+1} = P_t \cdot t \Delta c \]
Position Encodings

Requirements for Positional Encodings

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

Possible Candidates

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

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
3. The matrix can be learnt
4. Produces unique rotated embeddings each time
Rotary Position Embedding

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

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

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><strong>90.7</strong></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 Position Embeddings
Requirements for Position Encodings

• Some representation of time? (like seq2seq?)
• Should be unique for each position
• Bounded

Actual Candidates

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

Requirements for \( g(t) \)

• Must have same dimensions as input embeddings
• Must produce overall unique encodings
Position Encoding

For each position, an embedded input is moved the same distance but at a different angle. **Inputs that are close to each other in the sequence have similar perturbations, but inputs that are far apart are perturbed in different directions.**

\[
P_E(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)
\]
\[
P_E(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)
\]

- **pos** -> idx of the token in input sentence
- **i** -> i\(^{th}\) dimension out of \(d\)
- **d model** -> embedding dimension of each token

Different calculations for odd and even embedding indices.
Positional Encoding:

\[ PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d_{model}}}) \]
\[ PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}}) \]
I ate an apple

**Position Encoding**

Input

I ate an apple

Tokens

I  ate  an  apple  <eos>

Input Embeddings

Final Input Embeddings

Embedding Layer

Tokenizer

Position Encodings

Multi-Head Attention

LayerNorm

Forward
Transformers

✔ Tokenization
✔ Input Embeddings
✔ Position Encodings
  • Query, Key, & Value
  • Attention
  • Self Attention
  • Multi-Head Attention
  • Feed Forward
  • Add & Norm
  • Encoders

• Masked Attention
• Encoder Decoder Attention
• Linear
• Softmax
• Decoders
• Encoder-Decoder Models
WHERE IS THE CONTEXT?
I ate an apple <eos>

BLACK BOX OF SORTS
Encoder

LEARN TO ADD CONTEXT

BLACK BOX OF SORTS

Encoder

I ate an apple <eos>

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention

Input Embedding
Inputs

Positional Encoding

30
I ate an apple.
I ate an apple
Attention

\( \alpha_{[i,j]} \) ?

From lecture 18:

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

\[ \alpha_{[i,j]} \]

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}
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

{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}

{"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

Done at the same time!!

{Query: “Order details of order_104”}

OR

{Query: “Order details of order_106”}

{Key, Value store}

```json
{"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

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

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

Query
1. {Query: “Order details of order_104”}
2. {Query: “Order details of order_106”}

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

Example JSON:

{"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
Attention

Done at the same time!!
Attention

Parallelizable !!!

Query

Key Value Store

$Q$

$QK^T$

Key

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

Value

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

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

Dimensions across QKV have been dropped for brevity

Attention
Attention

Dimensions across QKV have been dropped for brevity.

$$a_{1,1}$$

$$e_{1,1}$$

softmax

$$Q_1, K_1, V_1$$

$$Q_2, K_2, V_2$$

$$Q_3, K_3, V_3$$

$$Q_4, K_4, V_4$$

$$Q_5, K_5, V_5$$

$$W_Q, W_K, W_V$$

$$L_1, L_2, L_3, L_4, L_5$$

I ate an apple <eos>
Attention

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

Dimensions across QKV have been dropped for brevity

\[ e_{1,1} \quad e_{1,2} \quad e_{1,3} \]

\[ \alpha_{1,1} \quad \alpha_{1,2} \quad \alpha_{1,3} \]

softmax

\[ W_Q^Q \quad W_K^K \quad W_V^V \]

\[ Q_1 \quad K_1 \quad V_1 \]

\[ Q_2 \quad K_2 \quad V_2 \]

\[ Q_3 \quad K_3 \quad V_3 \]

\[ Q_4 \quad K_4 \quad V_4 \]

\[ Q_5 \quad K_5 \quad V_5 \]

\[ l_1 \quad l_2 \quad l_3 \quad l_4 \quad l_5 \]

ate
an
apple
<eos>
Attention

Dimensions across QKV have been dropped for brevity.

I ate an apple <eos>
Attention

Dimensions across QKV have been dropped for brevity

softmax

\[ e_{1,1} \]
\[ e_{1,2} \]
\[ e_{1,3} \]
\[ e_{1,4} \]
\[ e_{1,5} \]

\[ \alpha_{1,1} \]
\[ \alpha_{1,2} \]
\[ \alpha_{1,3} \]
\[ \alpha_{1,4} \]
\[ \alpha_{1,5} \]

Q_1 K_1 V_1
W_Q W_K W_V
I_1

Q_2 K_2 V_2
W_Q W_K W_V
I_2

Q_3 K_3 V_3
W_Q W_K W_V
I_3

Q_4 K_4 V_4
W_Q W_K W_V
I_4

Q_5 K_5 V_5
W_Q W_K W_V
I_5

W

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

Dimensions across QKV have been dropped for brevity

Contextually rich embedding
Attention

Parallelized

Dimensions across QKV have been dropped for brevity

Contextually rich embedding
Transformers

✔ Tokenization
✔ Input Embeddings
✔ Position Encodings
✔ Query, Key, & Value
✔ Attention
  • Self Attention
  • Multi-Head Attention
  • Feed Forward
  • Add & Norm
  • Encoders

• Masked Attention
• Encoder Decoder Attention
• Linear
• Softmax
• Decoders
• Encoder-Decoder Models
Which of the following are true about attention?

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?

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
Self Attention

From lecture 18:

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

Query Inputs = Key Inputs = Value Inputs

SELF
Self Attention

\[ W_Q \quad W_K \quad W_V \]

Input Embeddings

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

\[ R^{T \times d_{model}} \]
Self Attention

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

Input Embeddings

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

Q Projections

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

K Projections

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

V Projections

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

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

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

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

softmax

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

Add & Norm

Feed Forward

Multi-Head Attention

Positional Encoding

Input Embedding

Inputs
Self Attention

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

softmax

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

Feed Forward

Multi-Head Attention

Add & Norm

Add & Norm

Positional Encoding

Input Embedding

Inputs
Self Attention

\[ O(T^2 \times d_{\text{model}}) \]

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

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

softmax

\[ Q_{\text{Projection}} \times K_{\text{Projection}} \]

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

\[ V_{\text{Projection}} \]
Self Attention

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

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

Coreference resolution ✓
The animal didn’t cross the street because it was too wide.
Self Attention

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

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

\[ d_h = \frac{d_{model}}{h} \]

Input Embeddings

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

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

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

\[ W_{V1}, W_{V2}, \ldots, W_{VH}, \]
Multi-Head Attention

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

Inputs

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

\[ \mathbf{W}_{Qi} \]

\[ \mathbf{Q}_i \]

\[ \mathbf{H} \]

\[ \mathbf{H} \]

\[ \mathbf{R}^{T \times d_{h}} \]

\[ \mathbf{R}^{T \times d_{h}} \]

\[ \mathbf{R}^{T \times d_{h}} \]

\[ \mathbf{W}_{Ki} \]

\[ \mathbf{K}_i \]

\[ \mathbf{H} \]

\[ \mathbf{H} \]

\[ \mathbf{R}^{T \times d_{h}} \]

\[ \mathbf{W}_{Vi} \]

\[ \mathbf{V}_i \]

\[ \mathbf{H} \]

\[ \mathbf{H} \]
Multi-Head Attention

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

softmax

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

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

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

\[ \cdots \]

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

\[ Z_1 \]

\[ Z_2 \]

CONCAT

\[ Z_h \]

Multi Head Attention : Z

\[ d_h = \frac{d_{model}}{h} \]
The animal didn’t cross the street because it was too wide.
Feed Forward

- Non Linearity
- Complex Relationships
- Learn from each other
Add & Norm

Normalization
Mean 0, Std dev 1
Stabilizes training
Regularization effect

Add Residuals
Avoid vanishing gradients
Train deeper networks
Add & Norm

Feed Forward

Add & Norm

Input

Norm(Z)

Residuals
Enoders

Encoder

ENCODER
Encoders

Encoder

ENCODER

. . .

ENCODER

ENCODER

Input to Encoder_{i+1}

Output from Encoder_{i}

Diagram showing the flow of encoding from one encoder to another, with attention mechanisms and normalization steps involved.
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders

- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models
Machine Translation

**Inputs**

I ate an apple

**Targets**

Ich habe einen Apfel gegessen
Ich habe einen Apfel gegessen
ich habe einen apfel gegessen

tokenizer

ich habe einen apfel gegessen

generate target embeddings
Masked Multi Head Attention

Ich habe einen Apfel gegessen
Decoding step by step (using Teacher Forcing)

Masked Multi Head Attention

Inference

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

Decoding step by step (using Teacher Forcing)

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>habe</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>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>3</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>4</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>5</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>6</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>7</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
</tbody>
</table>
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

<table>
<thead>
<tr>
<th></th>
<th>Ich</th>
<th>habe</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>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>3</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>4</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>5</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>6</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
<tr>
<td>7</td>
<td>&lt;sos&gt;</td>
<td>Ich</td>
<td>habe</td>
<td>einen</td>
<td>Apfel</td>
<td>gegessen</td>
</tr>
</tbody>
</table>

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 habe - ∞ - ∞ - ∞ - ∞
4. <sos> Ich habe einen - ∞ - ∞ - ∞
5. <sos> Ich habe einen Apfel - ∞
6. <sos> Ich habe einen Apfel gegessen - ∞
7. <sos> Ich habe einen Apfel gegessen <eos>

Softmax - ∞ -> 0
Masked Multi Head Attention

Masked Multi Head Attention: $Z'$

$$QK^T + R^{T \times T} = \text{Masked Attention}$$

Attention Mask: $M$
Masked Multi Head Attention

\[ R^{T \times T} \times R^{T \times d_h} \]
Encoder Decoder Attention

Encoder Decoder Attention?

Add & Norm

Input  Norm(Z')
Encoder Decoder Attention

Diagram showing the flow of data through an encoder-decoder architecture with attention mechanisms.
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}

```
{"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", ...} }
```
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

- 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

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Multi-Head Attention
Add & Norm
Feed Forward

Output Probabilities
Softmax
Linear

Positional Encoding
Input Embedding
Inputs

Outputs (shifted right)
Decoder
Decoder

Decoder output

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

DECODER

DECODER

DECODER
Linear weights are often tied with input embedding matrix.
Softmax

Output Probabilities

\( R^{T_d \times V} \)
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. Queries, keys, and values are obtained by splitting the input into 3 equal segments
d. Multihead attention might help transformers find different kinds of relations between tokens
e. Decoder outputs provide attention queries and keys, while the values come from the encoder
Poll 2 - @1580

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. Queries, keys, and values are obtained by splitting the input into 3 equal segments
d. Multihead attention might help transformers find different kinds of relations between tokens
e. Decoder outputs provide attention queries and keys, while the values come from the encoder
Transformers

Targets
Ich habe einen Apfel gegessen

Inputs
I ate an apple

Machine Translation
Transformers

✔ Tokenization
✔ Input Embeddings
✔ Position Encodings
✔ Query, Key, & Value
✔ Attention
✔ Self Attention
✔ Multi-Head Attention
✔ Feed Forward
✔ Add & Norm
✔ Encoders

✔ Masked Attention
✔ Encoder Decoder Attention
✔ Linear
✔ Softmax
✔ Decoders
  • Encoder-Decoder Models
Transformers
Transformers

**Input** – input tokens

**Output** – hidden states

**Representation**

**Generation**

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

**Output** – output tokens
Transformers

Input – input tokens
Output – hidden states

Model can see all timesteps

Representation

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

Model can only see previous timesteps

Generation
Transformers

**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**

**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*

**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*
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders

- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models

[Diagram of Transformer architecture]
Part 2

Pre-training and Fine-tuning
How to train and fine-tune transformers

1. Training

2. Inference
How to train and fine-tune transformers

1. Pre-training
2. Fine-tuning
3. Inference
How to train and fine-tune transformers

1. Pre-training

- Transformer architecture
- Supervised training
- Larger general dataset

2. Fine-tuning

- Pre-trained transformer model

3. Inference

Lot's of data, learn general things. May serve as a parameter initialization.

Usually requires significant computational resources and time.
1. Pre-training

Lot's of data, learn general things. May serve as a parameter initialization.

Usually requires significant computational resources and time.

2. Fine-tuning

Adaptation to the specific task.

Potentially less computationally intensive.

3. Inference
Parameter-Efficient Fine-Tuning Techniques

LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199
Parameter-Efficient Fine-Tuning Techniques

LoRA (Lower-Rank Adaptation)

LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199
Parameter-Efficient Fine-Tuning Techniques

LoRA (Lower-Rank Adaptation)

LoRA: https://arxiv.org/abs/2106.09685

BitFit: https://arxiv.org/abs/2106.10199

\[
Q^{m,\ell}(x) = W^{m,\ell}_q x + b^{m,\ell}_q \\
K^{m,\ell}(x) = W^{m,\ell}_k x + b^{m,\ell}_k \\
V^{m,\ell}(x) = W^{m,\ell}_v x + b^{m,\ell}_v
\]
Part 3

Transformer Applications
Transformers

Representation / Encoder

Generation / Decoder
Data Modalities

- Language (see Part 4 of the lecture)
- Vision
- Audio
- … and many other modalities (e.g., biological/physiological signals, etc.)
- Multimodal (>2 data modalities)
In computer vision convolutional architectures remain largely dominant. Inspired by NLP successes, multiple works try introducing combining CNN-like architectures with self-attention or replacing the convolutions entirely. However, they faced challenges with performance and scaling. Key breakthrough - Vision Transformer (ViT) released in 2020
Computer Vision - Tokenization

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
1. Split an image into fixed-size patches (16x16 pixels).
2. Tokenize each path (linear projection of flattened patches).
3. Add position embedding.
4. Feed the resulting sequence of vectors to a standard Transformer encoder.
5. For classification, add an extra learnable “classification token” to the sequence.
- ViT learns the grid-like structure of the image patches via its position embeddings.
- The lower layers contain both global and local features, the higher layers contain only global features.

*Transformers for Image Recognition at Scale*
ViT model attains state-of-the-art performance on multiple popular benchmarks, including 88.55% top-1 accuracy on ImageNet and 99.50% on CIFAR-10.
Audio

- Similar to the computer vision but with spectrograms instead of images.

- Exists as encoder-decoder variants or as an encoder-only variant with CTC loss.

- Could be augmented with the CNN.

Conformer: Convolution-augmented Transformer for Speech Recognition

AST: Audio Spectrogram Transformer
1. UniT handles 7 tasks ranging from object detection to vision-and language reasoning and natural language understanding.

2. Components:
   ○ An image encoder to encode the visual inputs.
   ○ A text encoder to encode the language inputs.
   ○ A joint decoder with per-task query embedding.
   ○ Task-specific heads to make the final outputs for each task.
Multimodal Transformer - LLaVA

Language Model $f_\phi$

Projection $W$

Vision Encoder $Z_v$

Language Response $X_a$

Language Instruction $X_q$

Image $X_v$
The painting depicts a dog in a humorous situation, where it appears to be wearing a woman's dress and standing in front of a landscape. Although the artist of the painting is not explicitly identified, it could be a reference to the famous artist Da Vinci, known for his artistic works, including the Mona Lisa. The painting is likely a creative and comical take on the traditional portrait style, replacing a human figure with a dog dressed in a woman's clothing, making it an interesting and amusing visual piece.
Part 4

Large Language Models
Transformers, mid-2017
2018 – Inception of the LLM Era

Representation

BERT
Oct 2018

Generation

GPT
June 2018
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 - 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 Results:
• BERT-Base – 110M Params
• BERT-Large – 340M Params
BERT - Bidirectional Encoder Representations

MLM (Masked Language Modeling)

How are they doing today?

Prediction head

you 60%

they 20%

... ...

60%

20%

MLM head
BERT - Bidirectional Encoder Representations

NSP (Next Sentence Prediction)

is_next: 95%
not_next: 5%

Prediction head

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

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

BERT

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention

N x

Positional Encoding

Input Embedding

Inputs

Outputs

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></td>
<td>392k</td>
<td>363k</td>
<td>108k</td>
<td>67k</td>
<td>8.5k</td>
<td>5.7k</td>
<td>3.5k</td>
<td>2.5k</td>
<td></td>
</tr>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</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>86.7/85.9</td>
<td>72.1</td>
<td>92.7</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>82.1</td>
</tr>
</tbody>
</table>

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

**Leaderboard (Oct 8th, 2018)**

<table>
<thead>
<tr>
<th>Published</th>
<th>Dev</th>
<th>EM</th>
<th>F1</th>
<th>Test</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiDAF+ELMo(Single)</td>
<td>-</td>
<td>-</td>
<td>85.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R.M. Reader (Single)</td>
<td>78.9</td>
<td>86.3</td>
<td>79.5</td>
<td>86.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R.M. Reader (Ensemble)</td>
<td>81.2</td>
<td>87.9</td>
<td>82.3</td>
<td>88.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Ours**

<table>
<thead>
<tr>
<th>System</th>
<th>Dev</th>
<th>EM</th>
<th>F1</th>
<th>Test</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_BASE</td>
<td>80.8</td>
<td>88.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT_LARGE</td>
<td>84.1</td>
<td>90.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>BERT_LARGE</strong></td>
<td>85.8</td>
<td>91.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(Sgl+TriviaQA)</td>
<td>84.2</td>
<td>91.1</td>
<td>85.1</td>
<td>91.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(En+TriviaQA)</td>
<td>86.2</td>
<td>92.2</td>
<td>87.4</td>
<td>93.2</td>
<td>-</td>
<td>-</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 was considered large in 2018
2018 – Inception of the LLM Era

BERT
Oct 2018

GPT
June 2018

Representation

Generation
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:
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!!!**
Poll 3 - @1579

The original GPT’s parameter count is closest to...

A. 117
B. 117K
C. 117M
D. 117B
Poll 3 - @1579

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
The LLM Era – Paradigm Shift in Machine Learning

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

... Representation

- T5 – 2019
- BART – 2019
- mT5 – 2021

... Generation

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

...
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>Feature Engineering</td>
</tr>
<tr>
<td>• How do we design or select the best features for a task?</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?
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?

Since LLMs
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></td>
</tr>
<tr>
<td>• How do we design or select the best features for a task?</td>
<td></td>
</tr>
<tr>
<td>Model Selection</td>
<td></td>
</tr>
<tr>
<td>• Which model is best for which type of task?</td>
<td></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>
<tr>
<td>Overfitting vs Generalization</td>
<td></td>
</tr>
<tr>
<td>• How do we balance complexity and capacity to prevent overfitting while maintaining good performance?</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

• 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?

• Recall: 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?

  • **Recall: 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 is all you need!!!**
    • 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??
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
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…

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