Transformer Models

CSE545 - Spring 2019
Review:
Feed Forward Network (full-connected)
Review:
Convolutional NN

(Barter, 2018)
Review: Recurrent Neural Network

\[ y(t) = f(h(t)W) \]

Activation Function

\[ h(t) = g(h(t-1)U + x(t)V) \]

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)
Can model computation (e.g. matrix operations for a single input) be parallelized?
Can model computation (e.g. matrix operations for a single input) be parallelized?
Can model computation (e.g. matrix operations for a single input) be parallelized?
Can model computation (e.g. matrix operations for a single input) be parallelized?

Ultimately limits how complex the model can be (i.e. its total number of parameters/weights) as compared to a CNN.
The Transformer: “Attention-only” models

Can handle sequences and long-distance dependencies, but....

- Don’t want complexity of LSTM/GRU cells
- Constant num edges between input steps
- Enables “interactions” (i.e. adaptations) between words
- Easy to parallelize -- don’t need sequential processing.
The Transformer: “Attention-only” models

Challenge:

- Long distance dependency when translating:

Kayla kicked the ball.

The ball was kicked by Kayla.
The Transformer: “Attention-only” models

Challenge:

- Long distance dependency when translating:

The ball was kicked by kayla.

Kayla kicked the ball.
Attention

\[ c_{hi} \]

\[ \alpha_{hi->s} \]

\[ z_1 \quad z_2 \quad z_3 \quad z_4 \]

Values
Attention

Score function:
\[ \psi_{\text{mult}}(h_i, s) = s^T W h_i \]
\[ \alpha_{h_i \rightarrow s} = \text{softmax}(\psi(h_i, s)) \]
Attention

Score function:

$$\psi_{\text{mult}}(h_i, s) = s^T W h_i$$

$$\alpha_{h_i \rightarrow s} = \text{softmax}(\psi(h_i, s))$$

$$c_{hi} = \sum_{n=1}^{\lvert s \rvert} \alpha_{h_i \rightarrow s_n} z_n$$
The Transformer: “Attention-only” models

Challenge:

- Long distance dependency when translating:

  Attention came about for encoder decoder models.

Then self-attention was introduced:
Attention

Score function:

\[ \psi_{mult}(h_i, s) = s^T W h_i \]

\[ \alpha_{h_i \rightarrow s} = \text{softmax}(\psi(h_i, s)) \]

\[ c_{h_i} = \sum_{n=1}^{||s||} \alpha_{h_i \rightarrow s_n} z_n \]
Self-Attention

Score function:
\[ \psi_{\text{mult}}(h_i, s) = s^T W h_i \]
\[ \alpha_{h_i \rightarrow s} = \text{softmax}(\psi(h_i, s)) \]
\[ c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \rightarrow s_n} z_n \]
The Transformer: “Attention-only” models

Attention as weighting a value based on a query and key:

(Eisenstein, 2018)
The Transformer: “Attention-only” models

\[
\alpha \quad \psi \quad h_{i-1} \quad h_i \quad h_{i+1}
\]

(Eisenstein, 2018)
The Transformer: “Attention-only” models

(Eisenstein, 2018)
The Transformer: “Attention-only” models

Output

\[
\alpha \quad \psi \quad h
\]

\[
h_{i-1} \quad h_i \quad h_{i+1} \quad h_{i+2}
\]
The Transformer: “Attention-only” models

Output

\[ \alpha \]

\[ \psi \]

\[ b \]

\[ h_{i-1} \]

\[ h_i \]

\[ h_{i+1} \]

\[ h_{i+2} \]

\[ w_{i-1} \]

\[ w_i \]

\[ w_{i+1} \]

\[ w_{i+2} \]

\[ FFN \]
The Transformer: “Attention-only” models

\[
\begin{align*}
\alpha & \rightarrow \psi \\
\psi & \rightarrow b
\end{align*}
\]
The Transformer: “Attention-only” models

\[ y_{i-1} \quad y_i \quad y_{i+1} \quad y_{i+2} \]

Output

\[ \alpha \]

\[ \psi \]

\[ h \]

\[ k \]

\[ v \]

\[ q \]

\[ w_{i-1} \quad w_i \quad w_{i+1} \quad w_{i+2} \]

\[ \ldots \]
The Transformer: “Attention-only” models

Attend to all hidden states in your “neighborhood”.
The Transformer: “Attention-only” models

Output

\[ \psi_{dp}(h_i, s) = s^T h_i \]

\[ k^T q \]
The Transformer: “Attention-only” models

\[ \psi_{dp}(k, q) = (k^t q) \sigma \]
The Transformer: “Attention-only” models

\[ \psi_{dp}(k,q) = (k^t q) \sigma \]

Linear layer: \( W^T X \)

One set of weights for each of for K, Q, and V
The Transformer

Limitation (thus far): Can’t capture multiple types of dependencies between words.
The Transformer

Solution: Multi-head attention
Multi-head Attention

Scaled Dot-Product Attention

MatMul

SoftMax

Mask (opt.)

Scale

MatMul

Linear

Concat

Scaled Dot-Product Attention

Linear

Linear

Linear

Q

K

V

V

K

Q

h
Transformer for Encoder-Decoder
Transformer for Encoder-Decoder

Stage 1: Positional Encoding

Stage 2: Multi-Head Attention

Input Embedding

Inputs

POSITIONAL ENCODING

EMBEDDINGS

INPUT

je

suis

sequence index (t)
Transformer for Encoder-Decoder
Transformer for Encoder-Decoder

Stage 1: Positional Encoding
- Input Embedding

Stage 2: N× Residualized Connections
- Add & Norm
- Multi-Head Attention

Stage 3: Add & Norm
- Feed Forward

Embedding lookup

Inputs
Transformer for Encoder-Decoder

Stage 1: Positional Encoding
- Input Embedding
- Inputs

Stage 2: Residualized Connections
- Add & Norm
- Multi-Head Attention
- Feed Forward

Stage 3: Residualized Connections
- Add & Norm
- Feed Forward

Residuals enable positional information to be passed along.

With residuals

Without residuals
Transformer for Encoder-Decoder
Transformer for Encoder-Decoder

essentially, a language model
Transformer for Encoder-Decoder

essentially, a language model

Decoder blocks out future inputs
Transformer for Encoder-Decoder

essentially, a language model

Add conditioning of the LM based on the encoder
Transformer for Encoder-Decoder
Transformer (as of 2017)

“WMT-2014” Data Set. BLEU scores:

<table>
<thead>
<tr>
<th></th>
<th>EN-DE</th>
<th>EN-FR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT (orig)</td>
<td>24.6</td>
<td>39.9</td>
</tr>
<tr>
<td>ConvSeq2Seq</td>
<td>25.2</td>
<td>40.5</td>
</tr>
<tr>
<td>Transformer*</td>
<td>28.4</td>
<td>41.8</td>
</tr>
</tbody>
</table>
Transformer

- Utilize Self-Attention
- Simple att scoring function (dot product, scaled)
- Added linear layers for Q, K, and V
- Multi-head attention
- Added positional encoding
- Added residual connection
- Simulate decoding by masking

https://4.bp.blogspot.com/-OlrV-PAtEkQ/W3RkOJCBkal/AAAAAAAADOg/gNZXq_6dK3mDQmIbfzsuyvPzrRFtNh3qE0w/wiCCLcbGAs/s640/image1.gif
Transformer

Why?
● Don’t need complexity of LSTM/GRU cells
● Constant num edges between words (or input steps)
● Enables “interactions” (i.e. adaptations) between words
● Easy to parallelize -- don’t need sequential processing.

Drawbacks:
● Only unidirectional by default
● Only a “single-hop” relationship per layer (multiple layers to capture multiple)
Why?

- Don't need complexity of LSTM/GRU cells
- Constant num edges between words (or input steps)
- Enables "interactions" (i.e. adaptations) between words
- Easy to parallelize -- don't need sequential processing.

Drawbacks of Vanilla Transformers:

- Only unidirectional by default
- Only a "single-hop" relationship per layer (multiple layers to capture multiple)

**BERT**

Bidirectional Encoder Representations from Transformers

Produces contextualized embeddings (or pre-trained contextualized encoder)
Why?

- Don’t need complexity of LSTM/GRU cells
- Constant num edges between words (or input steps)
- Enables “interactions” (i.e. adaptations) between words
- Easy to parallelize -- don’t need sequential processing.

Drawbacks of Vanilla Transformers:
- Only unidirectional by default
- Only a “single-hop” relationship per layer
  (multiple layers to capture multiple)

BERT

**Bidirectional Encoder Representations from Transformers**

Produces contextualized embeddings
(or pre-trained contextualized encoder)

- Bidirectional context by “masking” in the middle
- A lot of layers, hidden states, attention heads.
BERT

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Tokenize into “word pieces”

(Devlin et al., 2019)
Bert: Attention by Layers

https://colab.research.google.com/drive/1vOJ1hdujVjfH857hyYKldKPTD9Kid8

(Vig, 2019)
BERT Performance: e.g. Question Answering

GLUE scores evolution over 2018-2019

- Single generic models
- 2018 Task-specific-SOTA
- Human performance

<table>
<thead>
<tr>
<th>Model</th>
<th>GLUE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BILSTM+ELMo</td>
<td>71</td>
</tr>
<tr>
<td>GPT</td>
<td>75.2</td>
</tr>
<tr>
<td>BERT</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT Big</td>
<td>81.2</td>
</tr>
<tr>
<td>BigBird</td>
<td>82.2</td>
</tr>
</tbody>
</table>

https://rajpurkar.github.io/SQuAD-explorer/
BERT: Pre-training; Fine-tuning

Transformer encoder
12 or 24 layers
BERT: Pre-training; Fine-tuning

Transformer encoder
12 or 24 layers
BERT: Pre-training; Fine-tuning

Novel classifier
(e.g. sentiment classifier; stance detector...etc..)

Transformer encoder
12 or 24 layers
The Transformer: “Attention-only” models

Can handle sequences and long-distance dependencies, but....

- Don’t want complexity of LSTM/GRU cells
- Constant num edges between input steps
- Enables “interactions” (i.e. adaptations) between words
- Easy to parallelize -- don’t need sequential processing.