Previously: Using LSTMs to solve sequence problems
- Process sequences one element at a time.
- Maintain a 'memory' (cell state) to capture information about previous steps.
- Mitigates the RNN vanishing gradient problem
- Suitable for time series, speech, text, and other sequential data.
Gated recurrent units (GRUs)

\[
\begin{align*}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
h_{t} &= \phi(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot \hat{h}_t + z_t \odot h_{t-1}
\end{align*}
\]

Bottleneck Problem

- All the information about the source sequence must be stored in a single vector
  - How to translate a long paragraph?
  - How to summarize long articles?

https://web.stanford.edu/~jurafsky/slp3/

RNN for Machine Translation

Would be nice if we could "look back" at previous hidden states

RNN with Attention

https://web.stanford.edu/~jurafsky/slp3/
RNN with Attention

Discuss: What are limitations of such sequence to sequence models? Hint: Think about runtime.

Language Modelling History

Discuss: What are limitations of such sequence to sequence models? Hint: Think about runtime.

A bat flew out of the dugout, startling the baseball player and making him drop his bat.
Transformer Architecture

Introduced for seq2seq tasks like Machine translation, summarization, question answering, etc.

Input Embeddings

- Replace tokens with continuous vectors
- Made up of two components:
  - token embeddings
  - positional encoding depends on position and dimension index as follows:

\[
P_{E_{pos, 2i}}(\text{pos}) = \sin\left(\frac{\text{pos} \times \text{d}}{10000^{\frac{2i}{d}}}ight)
\]

\[
P_{E_{pos, 2i+1}}(\text{pos}) = \cos\left(\frac{\text{pos} \times \text{d}}{10000^{\frac{2i+1}{d}}}ight)
\]

Example Positional Encoding
Transformer Architecture

Self-Attention

Self-Attention: General Formula

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

- Q, K, V are all \((n \times d)\) matrices. Consider have an input of shape \(b \times n \times d\).
- What is the shape of \(QK^T\)?
  - What does this matrix represent?
- What is the shape of the final output?
  - What does this matrix represent?

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

Multi-Head Attention

- The Scaled Dot-Product Attention attends to one or few entries in the input key-value pairs.
- Idea: apply Scaled Dot-Product Attention multiple times on the linearly transformed inputs.

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_k)W^O
\]

where \(\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\)
Cross-Attention

Self-Attention vs. Masked Self-Attention

Self-Attention
Masked Self-Attention

Self-Attention vs. Masked Self-Attention

Self-Attention
Masked Self-Attention
Self-Attention vs. Masked Self-Attention

Point-wise Feed-forward Networks
- Purpose
  - Applies non-linear transformations to the output of the attention layer
- Equation
  - $F\text{FN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$
    - where $W$ and $b$ are learned weights and biases
  - These FFN is applied separately to each position

Layernorm
Discuss:

- How does the transformer scale with sequence length?
  - Any problems with applying it to very long sequences?

BERT (Bidirectional Encoder Representations from Transformers)

- Bidirectional Context
- Pre-trained on the language, and then fine-tuned

Input:

- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
Training

- **Masked Language Modelling**
  - Mask out k% of the input words, and then predict the masked words
  - The man went to the store to [MASK] a [MASK] of milk
  - What can you use as a loss function?

- **Next sentence prediction**
  - To learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

Model Details

- **Data:** Wikipedia (2.5B words) + BookCorpus (800M words)
- **Batch Size:** 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- **Training Time:** 1M steps (~40 epochs)
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay
- **BERT-Base:** 12-layer, 768-hidden, 12-head, 110M params
- **BERT-Large:** 24-layer, 1024-hidden, 16-head, 340M params
- **Trained on 4x4 or 8x8 TPU slice for 4 days**

Demo

[https://huggingface.co/google-bert/bert-large-cased?text=Paris+is+the+capital+of+%5BMASK%5D.](https://huggingface.co/google-bert/bert-large-cased?text=Paris+is+the+capital+of+%5BMASK%5D.)
Self-supervised Learning

- Labels are generated automatically, no human labeling process
- Benefits
  - Scales well
  - Cost-Efficient
  - Flexible
- Challenges
  - Larger datasets are required
  - More compute is necessary

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-dev/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>66.1</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>BERTBASE</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>BERTLARGE</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>

Review

- LSTMs/GRUs are recurrent
- Self-attention can effectively replace recurrence in sequence-to-sequence models
- Transformers use self-attention and are parallelizable
- Pre-training using self-supervised learning help train large models that learn very good representations