Attention and Transformer Sequence Models
Most NLP Tasks. E.g.

- Sequence Tasks
  o Language Modeling
  o Machine Translation
  o Speech Recognition
  o Named Entity
- Document Classification

Transformer Networks
  o Transformers
  o BERT
Evolution of Sequence Modeling

RNNs    LSTMs    LSTMs with Attention    Attention without LSTMs
Evolution of Sequence Modeling

RNNs     LSTMs     **LSTM with Attention**     Attention without LSTMs
Multi-level bidirectional RNN (LSTM or GRU)

(Eisenstein, 2018)
Multi-level bidirectional RNN (LSTM or GRU)

**Bidirectional**: Each node has a forward -> and backward <- hidden state: Can represent as a concatenation of both.

\[ h_{m-1}^{(s,D)} \leftarrow h_m \rightarrow h_{m+1}^{(s,2)} \]

\[ h_{m-1}^{(s,2)} \leftarrow h_m \rightarrow h_{m+1}^{(s,1)} \]

\[ x_{m-1}^{(s)} \leftarrow x_m \rightarrow x_{m+1}^{(s)} \]

(Eisenstein, 2018)
Multi-level bidirectional RNN (LSTM or GRU)

Average of top layer is an embedding (average of concatenated vectors)

$\mathbf{h}_{m-1}^{(s,D)} \quad \mathbf{h}_m^{(s,D)} \quad \mathbf{h}_{m+1}^{(s,D)}$

$\mathbf{h}_{m-1}^{(s,2)} \quad \mathbf{h}_m^{(s,2)} \quad \mathbf{h}_{m+1}^{(s,2)}$

$\mathbf{h}_{m-1}^{(s,1)} \quad \mathbf{h}_m^{(s,1)} \quad \mathbf{h}_{m+1}^{(s,1)}$

$\mathbf{x}_{m-1}^{(s)} \quad \mathbf{x}_m^{(s)} \quad \mathbf{x}_{m+1}^{(s)}$

(Eisenstein, 2018)
Multi-level bidirectional RNN (LSTM or GRU)

Sometimes just use left-most and right-most hidden state instead

(Eisenstein, 2018)
Sentiment Analysis:
Example Application of Single Representation of document

I feel terrible about what happened to Stark but the movie was excellent!
Sentiment Analysis:
Example Application of Single Representation of document

I feel terrible about what happened to Stark but the movie was excellent!
Sentiment Analysis:
Example Application of Single Representation of document

I feel terrible about what happened to Stark but the movie was excellent!
Encoder

A representation of input.

Document classification only needs an **encoder**: Goes from input into a single representation.

(Eisenstein, 2018)
Encoder

A representation of input.

Document classification only needs an **encoder**: Goes from input into a single representation.

What about tasks where the output is another sequence?
- translation
- speech to text

(Eisenstein, 2018)
Encoder-Decoder (seq to seq models)

Representing input and converting to output

What about tasks where the output is another sequence?
- translation
- speech to text

(Eisenstein, 2018)
Encoder-Decoder

(Eisenstein, 2018)
Encoder-Decoder

\[ y^{(0)} \quad y^{(1)} \quad y^{(2)} \quad y^{(3)} \]

<go>

...
Encoder-Decoder (Simpler Representation)
Encoder-Decoder (Simpler Representation)

essentially a language model conditioned on the final state from the encoder.
Encoder-Decoder (Simpler Representation)

Softmax essentially a language model conditioned on the final state from the encoder.
Encoder-Decoder (Simpler Representation)

Language 1: (e.g. Chinese)

Embedding lookup

Language 2: (e.g. English)

<go>  y(0)  y(1)  y(2)  y(3)  y(4)

Softmax
**Encoder-Decoder**

**Challenge:**

- Long distance dependency when translating:

Kayla kicked the ball.

The ball was kicked by kayla.
Encoder-Decoder

Challenge:

- Long distance dependency when translating:

  The ball was kicked by kayla.

  Kayla kicked the ball.

  A lot of responsibility put fixed-size hidden state passed from encoder to decoder.
Long Distance / Out of order dependencies

A lot of responsibility put fixed-size hidden state passed from encoder to decoder
Long Distance / Out of order dependencies

<go>

\[
\begin{align*}
    y_0 & \quad h_{m-1}^{(s,D)} & \quad h_m^{(s)} & \quad h_{m+1}^{(s,D)} \\
    y_1 & \quad h_{m-1}^{(s,L)} & \quad h_m^{(s)} & \quad h_{m+1}^{(s,L)} \\
    y_2 & \quad \vdots & \quad \vdots & \quad \vdots \\
    y_3 & \quad \vdots & \quad \vdots & \quad \vdots
\end{align*}
\]

Softmax
Attention

Analogy: random access memory
Attention

softmax

$y^{(0)}$ $y^{(1)}$ $y^{(2)}$ $y^{(3)}$

...
Attention

... attention layer...

\[ y_0 \]
\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
...

\[ z_{n-1} \]
\[ z_n \]
\[ z_{n+1} \]

\[ h_{n-1} \]
\[ h_n \]
\[ h_{n+1} \]

\[ s_1 \]
\[ s_2 \]
\[ s_3 \]
\[ s_4 \]

\[ x_{m-1}^{(m-1)} \]
\[ x_m^{(m)} \]
\[ x_{m+1}^{(m+1)} \]

\[ y_{(0)} \]
\[ y_{(1)} \]
\[ y_{(2)} \]
\[ y_{(3)} \]

Softmax

\[ c_{bi} \]

\[ i: \text{current token of output} \]
\[ N: \text{tokens of input} \]

\[ c_{hi} = \sum_{n=1}^{s} \alpha_{h_i \rightarrow s_n} s_n \]
Attention

\[ c_{hi} = \sum_{n=1}^{\left|s\right|} \alpha_{h_i \rightarrow s_n} s_n \]
Attention

\[ c_{hi} = \sum_{n=1}^{\left| s \right|} \alpha_{h_i \rightarrow s_n} s_n \]
**Attention**

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

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

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

Score function:

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

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

$$c_{h_i} = \sum_{n=1}^{\mid s \mid} \alpha_{h_i \rightarrow s_n} s_n$$
Alternative Scoring Functions

\[ \psi_{\text{add}}(h_i, s) = v^T \tanh(W_h h_i + W_s s) \]

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

\[ \psi_{\text{mult}}(h_i, s) = s^T W h_i \]
If variables are standardized, matrix multiply produces a similarity score.

Alternative Scoring Functions

\[ \psi_{\text{add}}(h_i, s) = v^T \tanh(W_h h_i + W_s s) \]

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

\[ \psi_{\text{mult}}(h_i, s) = s^T W h_i \]
Score function:
\[ \psi_{mult}(h_i, s) = s^T W h_i \]
\[ \alpha_{h_i \rightarrow s} = \text{softmax}(\psi(h_i, s)) \]
\[ c_{h_i} = \sum_{n=1}^{\mid s \mid} \alpha_{h_i \rightarrow s_n} s_n \]
A useful abstraction is to make the vector attended to (the “value vector”, $Z$) separate than the “key vector” ($s$).
A useful abstraction is to make the vector attended to (the “value vector”, $Z$) separate than the “key vector” ($s$).

Score function:
$$\psi_{\text{mult}}(h_i, s) = s^T W h_i$$
$$\alpha_{h_i\rightarrow s} = \text{softmax}(\psi(h_i, s))$$
$$c_{hi} = \sum_{n=1}^{||s||} \alpha_{h_i\rightarrow s_n} z_n$$
Attention as weighting a value based on a query and key:

(Eisenstein, 2018)
Machine Translation

As an optimization problem (Eisenstein, 2018):

\[
\hat{w}^{(t)} = \arg\max_{w^{(t)}} \Psi(w^{(s)}, w^{(t)})
\]
Machine Translation

Why?

- $40 billion/year industry
- A center piece of many genres of science fiction
- A fairly “universal” problem:
  - Language understanding
  - Language generation
- Societal benefits of inter-cultural communication
Machine Translation

Why?

- $40\text{billion/year}$ industry
- A center piece of many genres of science fiction
- A fairly “universal” problem:
  - Language understanding
  - Language generation
- Societal benefits of inter-cultural communication

(Douglas Adams)
Machine Translation

Why Neural Network Approach works? (Manning, 2018)

- Joint end-to-end training: learning all parameters at once.
- Exploiting distributed representations (embeddings)
- Exploiting variable-length context
- High quality generation from deep decoders - stronger language models (even when wrong, make sense)
Attention as weighting a value based on a query and key:

(Eisenstein, 2018)
Attention as weighting a value based on a query and key:

\[ \text{Output} \rightarrow \alpha \rightarrow \psi_\alpha \rightarrow \text{Query} \rightarrow \text{Key} \rightarrow \text{Value} \]

(“synced”, 2017)
Attention

\[ c_{hi} = \sum_{n=1}^{\mid s \mid} \alpha_{hi \rightarrow s_n} s_n \]

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

(“synced”, 2017)
Attention

\[ c_{h_i} = \sum_{n=1}^{\vert s \vert} \alpha_{h_i \rightarrow s_n} s_n \]

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

(Bahdanau et al., 2015)
Attention

\[ c_{hi} = \sum_{n=1}^{||s||} \alpha_{hi \to s_n} s_n \]

\( \alpha_{hi \to s} = \text{softmax}(\psi(h_i, s)) \)

(Bahdanau et al., 2015)

(“synced”, 2017)
Attention

\[ y_0, y_1, y_2, y_3, \ldots \]

Softmax

\[ s_1, s_2, s_3, s_4, \ldots \]

\[ h_{m-1}^{(s,2)}, h_m^{(s,2)}, h_{m+1}^{(s,2)} \]

\[ h_{m-1}^{(s,1)}, h_m^{(s,1)}, h_{m+1}^{(s,1)} \]

\[ x_m^{(s)}, x_{m-1}^{(s)}, x_{m+1}^{(s)} \]

\[ <\text{go}> \]
Do we even need all these RNNs?
(Vaswani et al., 2017: Attention is all you need)
Evolution of Sequence Modeling

RNNs  LSTM  LSTM with Attention  Attention without LSTMs
Evolution of Sequence Modeling

RNNs  LSTMs  LSTMs with Attention  Attention without LSTMs
A useful abstraction is to make the vector attended to (the “value vector”, $Z$) separate than the “key vector” ($s$).
The Transformer: “Attention-only” models

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

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

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

Output

FFN

\[ \alpha \]

\[ \Psi \]

\[ b \]

\[ b_{i-1} \]

\[ b_i \]

\[ b_{i+1} \]

\[ w_{i-1} \]

\[ w_i \]

\[ w_{i+1} \]

\[ w_{i+2} \]
The Transformer: “Attention-only” models
The Transformer: “Attention-only” models

Output

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

\[ \alpha \]

\[ \psi \]

\[ b \]

\[ b_{i-1} \quad b_i \quad b_{i+1} \quad b_{i+2} \]

\[ w_{i-1} \quad w_i \quad w_{i+1} \quad w_{i+2} \quad \ldots \]
The Transformer: “Attention-only” models

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

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

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

\[ \alpha \]
\[ \psi \]
\[ b \]

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

Output

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

Linear layer weights:

\[ W_{(k)^TX} \]
\[ W_{(v)^TX} \]
\[ W_{(q)^TX} \]
The Transformer: “Attention-only” models

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.
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

Q K V

Linear

Concat

Scaled Dot-Product Attention

Linear Linear Linear

V K Q

h
Transformer for Encoder-Decoder

sequence index (t)

Stage 1
Positional Encoding

Stage 2
Add & Norm
Multi-Head Attention

Embedding lookup

Input Embedding

Input

je
suis
Transformer for Encoder-Decoder

Residualized Connections

Stage 1: Positional Encoding
- Input Embedding
- Inputs

Stage 2: Multi-Head Attention
- Add & Norm
- Feed Forward
- N x

Stage 3: Add & Norm
- Y (0)
- Y (1)
- Y (2)
- Y (3)

<go>
Transformer for Encoder-Decoder

Residualized Connections

Stage 1: Positional Encoding
- Input Embedding
- Input

Stage 2: \(N \times\)
- Add & Norm
- Multi-Head Attention

Stage 3:
- Add & Norm
- Feed Forward
- \(Y_{(0)}, Y_{(1)}, Y_{(2)}, Y_{(3)}\)

<go>

Embedding lookup

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

Add conditioning of the LM based on the encoder

essentially, a language model
Transformer for Encoder-Decoder

Stage 1: Positional Encoding
- Input Embedding
- Inputs

Stage 2: N×
- Add & Norm
- Multi-Head Attention
- Add & Norm
- Masked Multi-Head Attention
- Add & Norm
- Feed Forward
- Output Probabilities

Stage 3: N×
- Add & Norm
- Feed Forward
- Add & Norm
- Output Probabilities

Stage 4: Add & Norm
- Feed Forward
- Output Probabilities

Stage 5: Softmax
- Linear
- Output Probabilities

Outputs (shifted right)
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><strong>28.4</strong></td>
<td><strong>41.8</strong></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
**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 number of 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

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.

*She saw the man on the hill with the telescope.*

*She [mask] the man on the hill [mask] the telescope.*

Mask 1 in 7 words:
- Too few: expensive, less robust
- Too many: not enough context
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-Base, Cased:**
    - 12-layer, 768-hidden, 12-heads, 110M parameters
  - **BERT-Large, Cased:**
    - 24-layer, 1024-hidden, 16-heads, 340M parameters
  - **BERT-Base, Multilingual Cased:**
    - 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
Differences from previous state of the art:

- Bidirectional transformer (through masking)
- Directions jointly trained at once.

(Devlin et al., 2019)
Differences from previous state of the art:

- Bidirectional transformer (through masking)
- Directions jointly trained at once.
- Capture sentence-level relations

(Devlin et al., 2019)
BERT

Differences from previous state of the art:

- Bidirectional transformer (through masking)
- Directions jointly trained at once.
- Capture sentence-level relations

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

https://colab.research.google.com/drive/1vlOJ1lhdujVjfh857hvYKldKPTD9Kid8

(Vig, 2019)
BERT: Pre-training; Fine-tuning

Embedding to vocab + softmax

Classification Layer: Fully-connected layer + GELU + Norm

W'₁  W'₂  W'₃  W'₄  W'₅

O₁  O₂  O₃  O₄  O₅

Transformer encoder
12 or 24 layers

Embedding

W₁  W₂  W₃  [MASK]  W₅

W₁  W₂  W₃  W₄  W₅
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
BERT: Pre-training; Fine-tuning

[CLS] vector at start is supposed to capture meaning of whole sequence.

Novel classifier (e.g. sentiment classifier; stance detector...etc..)
BERT: Pre-training; Fine-tuning

[CLS] vector at start is supposed to capture meaning of whole sequence.

Average of top layer (or second to top) also often used.

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