Transformer Sequence Models and Sequence Applications
(Machine Translation, Speech Recognition)

CSE392 - Spring 2019
Special Topic in CS
Most NLP Tasks. E.g.
- Sequence Tasks
  - Language Modeling
  - Machine Translation
  - Speech Recognition

Transformers
- Transformers
- BERT
Multi-level bidirectional RNN (LSTM or GRU)

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

Each node has a forward -> and backward <- hidden state: Can represent as a concatenation of both.
Multi-level bidirectional RNN (LSTM or GRU)

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

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

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

(Eisenstein, 2018)
Encoder

A representation of input.

(Eisenstein, 2018)
Encoder-Decoder

Representing input and converting to output

(Eisenstein, 2018)
Encoder-Decoder

\[
\begin{align*}
\mathbf{y}(0) & \quad \mathbf{y}(1) & \quad \mathbf{y}(2) & \quad \mathbf{y}(3) \\
\vdots & & & & \\
\end{align*}
\]

Softmax

(Eisenstein, 2018)
Encoder-Decoder

\[ y(0) \rightarrow y(1) \rightarrow y(2) \rightarrow y(3) \rightarrow \ldots \]

\[ \langle \text{go} \rangle \rightarrow h_{m-1}^{(s,D)} \rightarrow h_m^{(s,D)} \rightarrow h_{m+1}^{(s,D)} \rightarrow \ldots \]

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

A representation of input.
Encoder-Decoder

A representation of input.

essentially a language model conditioned on the final state from the encoder.
Encoder-Decoder

When applied to new data...

essentially a language model conditioned on the final state from the encoder.
Encoder-Decoder

A representation of input.
Encoder-Decoder

“seq2seq” model

Language 1: (e.g. Chinese)

Language 2: (e.g. English)
Encoder-Decoder

Challenge:

- Long distance dependency when translating:

\[
\begin{align*}
\mathbf{x}_0 & \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \ldots \rightarrow \mathbf{y}_0 \rightarrow \mathbf{y}_1 \rightarrow \mathbf{y}_2 \rightarrow \mathbf{y}_3 \rightarrow \mathbf{y}_4 \rightarrow \ldots
\end{align*}
\]
Encoder-Decoder

Challenge:

- Long distance dependency when translating:
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:

Kayla kicked the ball. The ball was kicked by kayla.

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>

Softmax

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

\[
\begin{align*}
y^{(0)} & \quad \cdots \quad y^{(1)} \quad \cdots \quad y^{(2)} \quad \cdots \quad y^{(3)} \\
\text{Softmax} \\
\end{align*}
\]
Attention

Analogy: random access memory

\[ y(0) \quad y(1) \quad y(2) \quad y(3) \]

Softmax

\[ y(s, D) \quad h_{m-1}^{(s, D)} \quad h_{m}^{(s, D)} \quad h_{m+1}^{(s, D)} \]

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

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

\[ x_{m-1}^{(s)} \quad x_{m}^{(s)} \quad x_{m+1}^{(s)} \]
Attention

Softmax

\[ y(0) \]
\[ y(1) \]
\[ y(2) \]
\[ y(3) \]

attention layer
Attention

Attention layer

\( y(0) \)
\( y(1) \)
\( y(2) \)
\( y(3) \)

Softmax

\( h_{i-1} \)
\( h_i \)
\( h_{i+1} \)

\( z_{n-1} \)
\( z_n \)
\( z_{n+1} \)

\( h_{n-1} \)
\( h_n \)
\( h_{n+1} \)

\( c_{bi} \)

\( s_1 \)
\( s_2 \)
\( s_3 \)
\( s_4 \)

\( s \)

i: current token of output

N: tokens of input

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

\[ c_{hi} \]

\[ \alpha_{hi \rightarrow s_1} \quad \alpha_{hi \rightarrow s_2} \quad \alpha_{hi \rightarrow s_3} \quad \alpha_{hi \rightarrow s_4} \]

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

\[ c_{hi} = \sum_{n=1}^{\mid s \mid} \alpha_{h_i \rightarrow s_n} s_n \]
$Z$ is the vector to be attended to (the value in memory). It is typically hidden states of the input (i.e. $s_n$) but can be anything.

$$c_{hi} = \sum_{n=1}^{|s|} \alpha_{h_i \rightarrow s_n} z_n$$
Attention

\[ c_{hi} \]

\[ \alpha_{hi \rightarrow s_1} \]
\[ \alpha_{hi \rightarrow s_2} \]
\[ \alpha_{hi \rightarrow s_3} \]
\[ \alpha_{hi \rightarrow s_4} \]

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

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

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

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

Score function:

$$\psi(h_i, s) = v^T \tanh(W_h h_i + W_s s)$$

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

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

A useful abstraction is to make the vector attended to (the “value vector”, $Z$) separate than the “key vector” ($s$).

Score function:

$$\psi(h_i, s) = v^T \tanh(W_h h_i + W_s s)$$

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

$$c_{h_i} = \sum_{n=1}^{\mid s \mid} \alpha_{h_i \rightarrow s_n} z_n$$
Attention

A useful abstraction is to make the vector attended to (the “value vector”, $Z$) separate than the “key vector” ($s$).

Score function:

$$\psi(h_i, s) = v^T \tanh(W_h h_i + W_s s)$$

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

$$c_{h_i} = \sum_{n=1}^{\lvert s \rvert} \alpha_{h_i \rightarrow s_n} z_n$$
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**

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

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

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

I am a student <s> Je suis étudiant </s>

attention weights

context vector

attention vector

(“synced”, 2017)
Attention

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

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

(labeled image)
Attention

$$c_{hi} = \sum_{n=1}^{|s|} \alpha_{h_i \rightarrow s_n} s_n$$

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

(Bahdanau et al., 2015)
Attention

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

Attention weights

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

(Bahdanau et al., 2015)

"synced", 2017
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
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)
Machine Translation

As an optimization problem (Eisenstein, 2018):

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

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

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

(“synced”, 2017)
Attention

Analogy: random access memory

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

Do we even need all these RNNs?
(Vaswani et al., 2017: Attention is all you need)
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” ($s$). (Eisenstein, 2018)
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

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

Output

\[ \alpha \]
\[ \psi \]
\[ \beta \]

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

\[ \alpha, \psi, b \]

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

Output

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

Attend to all hidden states in your “neighborhood”.

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

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

\[ h_{i-1} \quad h_i \quad h_{i+1} \quad h_{i+2} \]

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

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

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

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

Output

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

Linear layer: \( W^TX \)

One set of weights for each of for K, Q, and V
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
Transformer for Encoder-Decoder
Transformer for Encoder-Decoder
Transformer for Encoder-Decoder

Residualized Connections

Stage 1: Positional Encoding
- Input Embedding
- Inputs

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

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

Embedding lookup
<go>
Transformer for Encoder-Decoder

Residualized Connections

Stage 1: Positional Encoding

Stage 2: N x

Stage 3: Add & Norm

Feed Forward

Multi-Head Attention

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
**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 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**

**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.*
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

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
BERT

(Devlin et al., 2019)
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

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

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

(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

(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

Tokenizer 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

- BILSTM+ELMo: 71
- GPT: 75.2
- BERT: 79.6
- BERT Big: 81.2
- BigBird: 82.2

https://rajpurkar.github.io/SQuAD-explorer/
Bert: Attention by Layers

https://colab.research.google.com/drive/1vI0J1lhdujVifH857hyYKldKPTD9Kid8

(Vig, 2019)
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
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..)
BERT for Machine Translation:

(Lample & Conneau, Facebook, 2019)
BERT for Machine Translation:

(Lample & Conneau, Facebook, 2019)
BERT for Machine Translation:

Use as a pre-trained model for feeding into a machine translation system.

(Lample & Conneau, Facebook, 2019)
BERT for Machine Translation:

Use as a pre-trained model for feeding into a machine translation system.

(Lample & Conneau, Facebook, 2019)

<table>
<thead>
<tr>
<th>Pretraining</th>
<th>-</th>
<th>CLM</th>
<th>MLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sennrich et al. (2016)</td>
<td>33.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ro $\rightarrow$ en</td>
<td>28.4</td>
<td>31.5</td>
<td>35.3</td>
</tr>
<tr>
<td>ro $\leftrightarrow$ en</td>
<td>28.5</td>
<td>31.5</td>
<td>35.6</td>
</tr>
<tr>
<td>ro $\leftrightarrow$ en + BT</td>
<td>34.4</td>
<td>37.0</td>
<td>38.5</td>
</tr>
</tbody>
</table>

Table 3: Results on supervised MT. BLEU scores on WMT’16 Romanian-English. The previous state-of-the-art of Sennrich et al. (2016) uses both back-translation and an ensemble model. ro $\leftrightarrow$ en corresponds to models trained on both directions.
Neural Machine Translation

Where does neural approach fall short? (Manning, 2018)

- Translation process is mostly a black box -- can’t answer “why” for reordering, word choice decisions
- No direct use of semantic or syntactic structures
- Not modeling discourse structure -- only rough sense of how sentences relate to each other. Doesn’t model long distance anaphora.