Recurrance, Self-Attention and Transformers

Nov 25 2021

Lecturer: Jan van Gemert
Topic: Self-Attention

- Sequences
- Recurrance
- Self-attention / transformers

Sequence data

Let's deep learn a language translation system.

Consider this sentence:
The student, after reading the book, was well prepared.

Q: Would you use a feed-forward network?
A: No.
Q: Why not?
A: Sequences can have variable length.
Q: So what? Can't I just pad with zeros?
A: No; Feed forward will learn exact word locations.
Q: Why is that a problem?
A: Need to see all possible word/sentence options during training (impossible).
Sequence data

Let's deep learn a language translation system.

Consider this sentence:

*The student, after reading the book, was well prepared.*
Sequence data

Let's deep learn a language translation system.

Consider this sentence:

*The student, after reading the book, was well prepared.*

Q: Would you use feed forward network?
Sequence data

Let’s deep learn a language translation system.

Consider this sentence:

*The student, after reading the book, was well prepared.*

Q: Would you use feed forward network?
A: No.

Q: Why not?
A: Sequences can have variable length.
Q: So what? Can’t I just pad with zeros?
A: No; Feed forward will learn exact word locations;
Q: Why is that a problem?
A: Need to see all possible word/sentence options during training (impossible).
Sequence data

Let's deep learn a language translation system.

Consider this sentence:

The student, after reading the book, was well prepared.

Q: Would you use feed forward network?
A: No. Q: Why not?

Q: Why not?
A: Sequences can have variable length.

Q: So what? Can't I just pad with zeros?
A: No; Feed forward will learn exact word locations;

Q: Why is that a problem?
A: Need to see all possible word/sentence options during training (impossible).
Sequence data

Let's deep learn a language translation system.

Consider this sentence:

*The student, after reading the book, was well prepared.*

Q: Would you use feed forward network?
A: No. Q: Why not?
A: Sequences can have variable length.
Sequence data

Lets deep learn a language translation system.

Consider this sentence:

*The student, after reading the book, was well prepared.*

Q: Would you use feed forward network?
A: No. Q: Why not?
A: Sequences can have variable length.
Q: So what? Can’t I just pad with zeros?
Sequence data

Let's deep learn a language translation system.

Consider this sentence:

*The student, after reading the book, was well prepared.*

Q: Would you use feed forward network?
A: No. Q: Why not?
A: Sequences can have variable length.
Q: So what? Can’t I just pad with zeros?
A: No; Feed forward will learn exact word locations;
Sequence data

Let's deep learn a language translation system.

Consider this sentence:

_The student, after reading the book, was well prepared._

Q: Would you use feed forward network?
A: No. Q: Why not?
A: Sequences can have variable length.
Q: So what? Can't I just pad with zeros?
A: No; Feed forward will learn exact word locations; Q: Why is that a problem?
Sequence data

Lets deep learn a language translation system.

Consider this sentence:

\textit{The student, after reading the book, was well prepared.}

Q: Would you use feed forward network?
A: No. Q: Why not?
A: Sequences can have variable length.
Q: So what? Can’t I just pad with zeros?
A: No; Feed forward will learn exact word locations; Q: Why is that a problem?
A: Need to see all possible word/sentence options during training (impossible).
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Q: Explain the layout?
A: A 2-layer network for letter prediction.

Q: How to represent input letters?
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Q: Explain the layout?

Input characters

Output characters

Hidden layer

Input layer

Q: How to represent input letters?
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/


Q: How to represent input letters?
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Example: Predict the next character
DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Q: How to represent output?
### Example: Predict the next character

DL book, Chapter 10; and: [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Input layer</th>
<th>Hidden layer</th>
<th>Output layer</th>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>-3</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>4.1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>0</td>
<td></td>
<td>4.1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Input layer</th>
<th>Hidden layer</th>
<th>Output layer</th>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>-1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>1.2</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>1.9</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>0</td>
<td>-1.1</td>
<td>2.2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Input layer</th>
<th>Hidden layer</th>
<th>Output layer</th>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0.5</td>
<td>-1.5</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>1.9</td>
<td>-0.1</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>-1.1</td>
<td>2.2</td>
<td></td>
</tr>
</tbody>
</table>
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Q: What do the bold numbers mean?

Q: What would a fully connected network look like?
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Output characters</th>
<th>E</th>
<th>L</th>
<th>L</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td>1.0</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>0.3</td>
<td>0.5</td>
<td>-1.5</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>-1</td>
<td>1.9</td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>4.1</td>
<td>1.2</td>
<td>-1.1</td>
<td>2.2</td>
</tr>
</tbody>
</table>

Hidden layer

<table>
<thead>
<tr>
<th>Input layer</th>
<th>H</th>
<th>E</th>
<th>L</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input characters</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Input layer

Q: What do the bold numbers mean? A: Target (make high)
Example: Predict the next character

DL book, Chapter 10; and: [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

<table>
<thead>
<tr>
<th>Output characters</th>
<th>E</th>
<th>L</th>
<th>L</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>0.3</td>
<td>0.5</td>
<td>-1.5</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>-1</td>
<td>1.9</td>
<td>-0.1</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>1.2</td>
<td>-1.1</td>
<td>2.2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hidden layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
</tr>
</tbody>
</table>

Q: What do the bold numbers mean? A: Target (make high)
Q: What would a fully connected network look like?
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hidden layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>L</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>L</td>
</tr>
</tbody>
</table>
### Example: Predict the next character

DL book, Chapter 10; and: [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

---

<table>
<thead>
<tr>
<th>Output characters</th>
<th>E</th>
<th>L</th>
<th>L</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>Hidden layer</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>Input layer</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Q: Easiest solution to remove position-specific weights?

---

*Recurrence, Self-Attention and Transformers*  
*ASCI - EDL Winterschool*  
*Nov 25 2021 7 / 34*
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Output layer</th>
<th>Hidden layer</th>
<th>Input layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1.0</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>2.2</td>
<td>-0.1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-3</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4.1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0.5</td>
<td>-0.5</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0.3</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>O</td>
<td>-1.1</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>O</td>
<td>1.2</td>
<td>-0.3</td>
<td>1</td>
</tr>
<tr>
<td>O</td>
<td>4.1</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Output characters:

- E
- L
- O

Q: Easiest solution to remove position-specific weights? A: Share all weights.
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input layer</th>
<th>Hidden layer</th>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{H} )</td>
<td>( \text{E} )</td>
<td>( \text{L} )</td>
</tr>
<tr>
<td>( \text{E} )</td>
<td>( \text{L} )</td>
<td>( \text{O} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{W}_{xh} )</th>
<th>( \text{W}_{hq} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 0, 0, 0]</td>
<td>[0.3, -0.1, 0.9, 0.1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>[1.0, 0.3, 0.0, 0.1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>[0.1, -0.5, -0.3, 0.1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>[0.2, -1.5, -0.1, 0.7]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{W}_{xh} )</th>
<th>( \text{W}_{hq} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 0, 0, 0]</td>
<td>[0.3, -0.1, 0.9, 0.1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>[1.0, 0.3, 0.0, 0.1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>[0.1, -0.5, -0.3, 0.1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>[0.2, -1.5, -0.1, 0.7]</td>
</tr>
</tbody>
</table>

Q: What is the problem for 'L'?
A: Individual predictions. No past/context.

Q: How can the past be included?
Recurrance, Self-Attention and Transformers
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Q: What is the problem for 'L'?

A: Individual predictions. No past/context.
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Output characters</th>
<th>E</th>
<th>L</th>
<th>L</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td>1.0</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>0.3</td>
<td>0.5</td>
<td>-1.5</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>-1</td>
<td>1.9</td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>4.1</td>
<td>1.2</td>
<td>-1.1</td>
<td>2.2</td>
</tr>
</tbody>
</table>

Hidden layer:

<table>
<thead>
<tr>
<th></th>
<th>Whq</th>
<th>Whq</th>
<th>Whq</th>
<th>Whq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>1.0</td>
<td>0.1</td>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>-0.1</td>
<td>0.3</td>
<td>-0.5</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>-0.3</td>
<td>0.7</td>
<td></td>
</tr>
</tbody>
</table>

Input layer:

<table>
<thead>
<tr>
<th></th>
<th>Wxh</th>
<th>Wxh</th>
<th>Wxh</th>
<th>Wxh</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Q: What is the problem for 'L'? A: Individual predictions. No past/context.
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input layer</th>
<th>Hidden layer</th>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0</td>
<td>0.3 -0.1 0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>1.0 0.3 0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0.1 -0.5 -0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td></td>
<td>0.2</td>
</tr>
</tbody>
</table>

Q: What is the problem for 'L'? A: Individual predictions. No past/context.
Q: How can the past be included?

Recurrance, Self-Attention and Transformers
ASCI - EDL Winterschool
Nov 25 2021
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Output characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input layer</th>
<th>Hidden layer</th>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>E</td>
<td>2.2</td>
<td>-3</td>
</tr>
<tr>
<td>L</td>
<td>-3</td>
<td>4.1</td>
</tr>
<tr>
<td>L</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>O</td>
<td>0.5</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input characters</th>
<th>Hidden layer</th>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>E</td>
<td>-0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>L</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>L</td>
<td>0.3</td>
<td>-0.1</td>
</tr>
<tr>
<td>O</td>
<td>0.5</td>
<td>-1.1</td>
</tr>
</tbody>
</table>

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

Recurrance, Self-Attention and Transformers

ASCI - EDL Winterschool
Nov 25 2021
9 / 34
Example: Predict the next character

DL book, Chapter 10; and: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.
RNN
D2L-book chapter 8.4.2.

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

\[ H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \]
\[ O_t = H_t W_{hq} + b_q \]

Q: Explain the symbols?
- \( H_t \): Activation at current time step
- \( \phi \): non-linearity, (often a \( \tanh \), range in \((-1,1)\) )
- \( X_t \): input at time \( t \)
- \( W_{xh} \): Learned weights for input at time \( t \)
- \( H_{t-1} \): Activation at previous time step \( t-1 \)
- \( W_{hh} \): Learned weights: how to use the previous information at \( t-1 \)
- \( W_{hq} \): Learned weights for output at time \( t \)
- \( b_h, b_q \): Learned bias terms
Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $X_t$ for $t = 1, 2, 3, \ldots, T$
Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $X_t$ for $t = 1, 2, 3, \ldots, T$

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h),$$
Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $X_t$ for $t = 1, 2, 3, \ldots, T$

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h), \quad \text{output } O_t = H_t W_{hq} + b_q$$
RNN
D2L-book chapter 8.4.2.

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence \( X_t \) for \( t = 1, 2, 3, \ldots, T \)

\[
H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h), \quad \text{output } O_t = H_t W_{hq} + b_q
\]

Q: Explain the symbols?

- \( H_t \): Activation at current time step
- \( \phi \): non-linearity, (often a \( \tanh \), range in \((-1,1)\) )
- \( X_t \): input at time \( t \)
- \( W_{xh} \): Learned weights for input at time \( t \)
- \( H_{t-1} \): Activation at previous time step \( t-1 \)
- \( W_{hh} \): Learned weights: how to use the previous information at time \( t-1 \)
- \( W_{hq} \): Learned weights for output at time \( t \)
- \( b_h, b_q \): Learned bias terms
Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $X_t$ for $t = 1, 2, 3, \ldots, T$

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h), \quad \text{output } O_t = H_t W_{hq} + b_q$$

Q: Explain the symbols?

- $H_t$: Activation at current time step $t$
- $\phi$: non-linearity, (often a $\tanh$, range in (-1,1) )
- $X_t$: input at time $t$
- $W_{xh}$: Learned weights for input at time $t$
- $H_{t-1}$: Activation at previous time step $t - 1$
- $W_{hh}$: Learned weights: how to use the previous information at $t - 1$
- $W_{hq}$: Learned weights for output at time $t$
- $b_h, b_q$: Learned bias terms
RNN: Nr parameters doesn’t grow with sequence length
D2L-book chapter 8.4.2.

Recurrent Neural Network:  
Process a sequence in order and share the same parameters at each time step.

For a sequence $X_t$ for $t = 1, 2, 3, \ldots, T$

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h), \quad \text{output } O_t = H_t W_{hq} + b_q$$
RNN: Nr parameters doesn’t grow with sequence length

D2L-book chapter 8.4.2.

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $X_t$ for $t = 1, 2, 3, \ldots, T$

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h), \quad \text{output } O_t = H_t W_{hq} + b_q$$

More advanced variants of RNN: GRU/LSTM
Questions?
Sequence models

Let's learn a model to predict the next word.

Consider these sentences:

*The student, after reading the [old, thick, red, …] book, was prepared.*
*The students, after reading the [old, thick, red, …] book, were prepared.*

Q: What is the difficulty here?
Sequence models

Lets learn a model to predict the next word.

Consider these sentences:

*The student, after reading the [old, thick, red, ...] book, was prepared.*

*The students, after reading the [old, thick, red, ...] book, were prepared.*

Q: What is the difficulty here?
A: Variable sized long distance relation between “student[s]” and “was[were]”.
Recap: Predict the next word with an RNN

<table>
<thead>
<tr>
<th>Output words</th>
<th>student</th>
<th>...</th>
<th>...</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td>1.0</td>
<td>2.2</td>
<td>-3</td>
<td>4.1</td>
</tr>
<tr>
<td>Hidden layer</td>
<td>0.3</td>
<td>-0.1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>Input layer</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Input words</td>
<td>the</td>
<td>student</td>
<td>...</td>
<td>book</td>
</tr>
</tbody>
</table>

Q: Explain the figure?
A: Recurrent net, with shared weights at each time step.

Q: Why are recurrent nets (RNN/GRU/LSTM) difficult to train in parallel?
A: Inherently sequential: Each step depends on the previous.
Recap: Predict the next word with an RNN

Q: Explain the figure?
Recap: Predict the next word with an RNN

Q: Explain the figure? A: Recurrent net, with shared weights at each time step
Recap: Predict the next word with an RNN

Q: Explain the figure? A: Recurrent net, with shared weights at each time step
Q: Why are recurrent nets (RNN/GRU/LSTM) difficult to train in parallel?
Recap: Predict the next word with an RNN

Q: Explain the figure? A: Recurrent net, with shared weights at each time step

Q: Why are recurrent nets (RNN/GRU/LSTM) difficult to train in parallel?
A: Inherently sequential: Each step depends on the previous.
Self-attention: Removing the recurrence

<table>
<thead>
<tr>
<th>Output words</th>
<th>student</th>
<th>...</th>
<th>...</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td>1.0</td>
<td>2.2</td>
<td>-3</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-attention layer</td>
<td>W_{hq}</td>
<td></td>
<td></td>
<td>W_{hq}</td>
</tr>
<tr>
<td>Input layer</td>
<td>0.3</td>
<td>1.0</td>
<td>...</td>
<td>0.1</td>
</tr>
<tr>
<td>Input words</td>
<td>the</td>
<td>student</td>
<td>...</td>
<td>book</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
<td>0.3</td>
<td></td>
<td>-0.5</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.1</td>
<td></td>
<td>-0.3</td>
</tr>
</tbody>
</table>

Assume a (learned) word representation (Word2vec: D2L-book chapter 14.1)
Assume a (learned) word representation (Word2vec: D2L-book chapter 14.1)
Q: How to remove recurrence, yet introduce word relationships?
Assume a (learned) word representation (Word2vec: D2L-book chapter 14.1)

Q: How to remove recurrence, yet introduce word relationships?

A: Link each word to each other word.
Q: What happened to the word vectors in the self-attention layer?
Self-attention: Removing the recurrence

<table>
<thead>
<tr>
<th>Input words</th>
<th>the</th>
<th>student</th>
<th>...</th>
<th>...</th>
<th>book</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.9</td>
<td>1.0</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>-0.1</td>
<td></td>
<td>0.1</td>
<td>-0.1</td>
<td></td>
<td>-0.3</td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Self-attention layer</th>
<th>0.1</th>
<th>1.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.1</td>
<td>0.1</td>
<td>-0.1</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output layer</th>
<th>student</th>
<th>...</th>
<th>...</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2.2</td>
<td>-3</td>
<td>4.1</td>
<td>0.1</td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td>1.9</td>
</tr>
<tr>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
<td>-1.1</td>
</tr>
</tbody>
</table>

\[ W_{hq} \]

Q: What happened to the word vectors in the self-attention layer?
A: Influenced and changed by all other words.
Self attention

D2L-book chapter 10

See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

- Input vectors: $x_1, x_2, \ldots, x_t$, of size $d$
- Output vectors: $y_1, y_2, \ldots, y_t$, of size $d$

Start with words: the, student, reads, a, book

Learned embedding vectors: $x_{the}, x_{student}, x_{reads}, x_{a}, x_{book}$

Self-attention converts this to: $y_{the}, y_{student}, y_{reads}, y_{a}, y_{book}$

Self-attention: Learning how to re-weight words with all other words.

Q: Why is self-attention parallelizable?
A: No more recurrence; each vector can be computed in parallel.
Self attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets
Self attention
D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.
Self attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

- Input vectors: $x_1, x_2, \ldots, x_t$, of size $d$
- Output vectors: $y_1, y_2, \ldots, y_t$, of size $d$
Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

• Input vectors: $x_1, x_2, \ldots, x_t$, of size $d$
• Output vectors: $y_1, y_2, \ldots, y_t$, of size $d$

Start with words: the, student, reads, a, book
Self attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

- Input vectors: \(x_1, x_2, \ldots, x_t\), of size \(d\)
- Output vectors: \(y_1, y_2, \ldots, y_t\), of size \(d\)

Start with words: \(\text{the, student, reads, a, book}\)
Learned embedding vectors: \(x_{\text{the}}, x_{\text{student}}, x_{\text{reads}}, x_{\text{a}}, x_{\text{book}}\)
Self attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

- Input vectors: $x_1, x_2, \ldots, x_t$, of size $d$
- Output vectors: $y_1, y_2, \ldots, y_t$, of size $d$

Start with words: the, student, reads, a, book

Learned embedding vectors: $x_{\text{the}}, x_{\text{student}}, x_{\text{reads}}, x_{a}, x_{\text{book}}$

Self attention converts this to: $y_{\text{the}}, y_{\text{student}}, y_{\text{reads}}, y_{a}, y_{\text{book}}$
Self attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

• Input vectors: $x_1, x_2, \ldots, x_t$, of size $d$
• Output vectors: $y_1, y_2, \ldots, y_t$, of size $d$

Start with words: the, student, reads, a, book
Learned embedding vectors: $x_{\text{the}}, x_{\text{student}}, x_{\text{reads}}, x_a, x_{\text{book}}$
Self attention converts this to: $y_{\text{the}}, y_{\text{student}}, y_{\text{reads}}, y_a, y_{\text{book}}$

Self-attention: Learning how to re-weight words with all other words.
Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

- Input vectors: $x_1, x_2, \ldots, x_t$, of size $d$
- Output vectors: $y_1, y_2, \ldots, y_t$, of size $d$

Start with words:  
the, student, reads, a, book

Learned embedding vectors:  
$x_{\text{the}}, x_{\text{student}}, x_{\text{reads}}, x_{a}, x_{\text{book}}$

Self attention converts this to:  
$y_{\text{the}}, y_{\text{student}}, y_{\text{reads}}, y_{a}, y_{\text{book}}$

Self-attention: Learning how to re-weight words with all other words.

Q: Why is self-attention parallelizable?
Self attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Self attention: Parallelizable alternative to training recurrent nets

Self attention is a sequence-to-sequence operation.

- Input vectors: \( x_1, x_2, \ldots, x_t \), of size \( d \)
- Output vectors: \( y_1, y_2, \ldots, y_t \), of size \( d \)

Start with words: the, student, reads, a, book
Learned embedding vectors: \( x_{\text{the}}, x_{\text{student}}, x_{\text{reads}}, x_{a}, x_{\text{book}} \)
Self attention converts this to: \( y_{\text{the}}, y_{\text{student}}, y_{\text{reads}}, y_{a}, y_{\text{book}} \)

Self-attention: Learning how to re-weight words with all other words.

Q: Why is self-attention parallelizable?
A: No more recurrence; each vector can be computed in parallel.
Self-attention: Removing the recurrence

<table>
<thead>
<tr>
<th>Input words</th>
<th>the</th>
<th>student</th>
<th>...</th>
<th>...</th>
<th>book</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.1</td>
<td>1.0</td>
<td>1.0</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>-0.1</td>
<td>-0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>-0.5</td>
<td>-0.5</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>-0.3</td>
<td>-0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output words</th>
<th>student</th>
<th>...</th>
<th>...</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2.2</td>
<td>-3</td>
<td>4.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1.1</td>
</tr>
</tbody>
</table>

Self-attention: Learning how to re-weight words with all other words.
Self-attention: Learning how to re-weight words with all other words.

Q: How to determine how much word A can re-weight a word B?
Self-attention: Learning how to re-weight words with all other words.

Q: How to determine how much word A can re-weight a word B?
A: Based on some similarity of A and B.
Self-attention: Word re-weighting

Q: How many similarities?
A: All words to all words; for $n$ words, there are $n^2$ similarities.
Self-attention: Word re-weighting

Q: How many similarities?
Self-attention: Word re-weighting

Q: How many similarities?
A: All words to all words; for $n$ words, there are $n^2$ similarities.
Self-attention: Word re-weighting

The student book was

Q: Let's use the dot product. Can you compute $\langle x, x \rangle$?
Self-attention: Word re-weighting

Q: Let's use the dot product. Can you compute $<x_{\text{the}}, x_{\text{the}}>$?
## Self-attention: Word re-weighting

<table>
<thead>
<tr>
<th></th>
<th>was</th>
<th>book</th>
<th>student</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3</td>
<td>0.1</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>0.9</td>
<td>0.5</td>
<td>1.1</td>
<td>0.4</td>
<td>0.9</td>
</tr>
<tr>
<td>0.7</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>0.7</td>
<td>0.5</td>
<td>0.7</td>
<td>0.0</td>
<td>0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>student</th>
<th>book</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>-0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>1.0</td>
<td>-0.1</td>
<td>0.3</td>
<td>0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>-0.5</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Q:** Dot product $x^\top_i x_j$ is unbounded in $(-\infty, \infty)$, how to normalize?

**A:** Use the soft-max $w_{ij} = \exp x_i^\top x_j / \sum_t j \exp x_i^\top x_j$, so $\sum_j w_{ij} = 1$.
### Self-attention: Word re-weighting

<table>
<thead>
<tr>
<th></th>
<th>was</th>
<th>book</th>
<th>student</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3</td>
<td>0.1</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.3</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>-0.2</td>
<td>0.4</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>-0.1</td>
<td>-0.1</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>-0.7</td>
<td>0.4</td>
<td>0.1</td>
<td>-0.2</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>-0.7</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Q: Dot product $x_i^T x_j$ is unbounded in $(-\infty, \infty)$, how to normalize?
Q: Dot product $x_i^T x_j$ is unbounded in $(-\infty, \infty)$, how to normalize?

A: Use the soft-max $w_{ij} = \frac{\exp x_i^T x_j}{\sum_j \exp x_i^T x_j}$, so $\sum_j w_{ij} = 1$
Self-attention: Word re-weighting

```
<table>
<thead>
<tr>
<th></th>
<th>was</th>
<th>book</th>
<th>student</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3</td>
<td>0.9</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.1</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>0.25</th>
<th>0.2</th>
<th>0.1</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.15</td>
<td>0.1</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>the</th>
<th>student</th>
<th>book</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>1.0</td>
<td>0.1</td>
<td>-0.3</td>
</tr>
<tr>
<td>-0.1</td>
<td>0.3</td>
<td>-0.5</td>
<td>0.9</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>-0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>
```

Q: Wait, but the rows do not sum to 1?
A: Softmax here was over the columns.

Q: How to compute new x?
A: Weighted sum of word vectors.
Self-attention: Word re-weighting

<table>
<thead>
<tr>
<th></th>
<th>was</th>
<th>book</th>
<th>student</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>was</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.5</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
</tbody>
</table>

|     | 0.25 | 0.15 | 0.2  | 0.4  |
|     | 0.2  | 0.1  | 0.5  | 0.1  |
|     | 0.1  | 0.4  | 0.1  | 0.1  |
|     | 0.6  | 0.1  | 0.1  | 0.2  |

Q: Wait, but the rows do not sum to 1?

A: Softmax here was over the columns.

Q: How to compute new \( x \) book?

A: Weighted sum of word vectors.
Q: Wait, but the rows do not sum to 1? A: Softmax here was over the columns.
Self-attention: Word re-weighting

Q: Wait, but the rows do not sum to 1? A: Softmax here was over the columns.

Q: How to compute new $x_{\text{book}}$?
Q: Wait, but the rows do not sum to 1? A: Softmax here was over the columns.
Q: How to compute new $x_{book}$? A: Weighted sum of word vectors.
Self-attention: Word re-weighting

The diagram illustrates the process of self-attention in a neural network, focusing on re-weighting words within a sentence. The weights assigned to each word are shown in the matrix, with each word's weight being multiplied by other weights to produce a final weighted value. For example, the word "student" has weights of 0.3, 0.5, and 0.1 for the words "was", "book", and "the", respectively. These weights are multiplied together (0.3 * 0.5 * 0.1 = 0.015) and then summed up with the weights from other words to give a final weighted value. This process helps in understanding the importance of each word in the context of the sentence.
Self-attention: Word re-weighting

Questions ?
Questions?
Self-attention: Learnable word roles and modifiers

Self-attention: Learning how to re-weight words with all other words.

<table>
<thead>
<tr>
<th>Input words</th>
<th>the</th>
<th>student</th>
<th>...</th>
<th>book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-attention layer</td>
<td>1.0</td>
<td>2.2</td>
<td>-3</td>
<td>4.1</td>
</tr>
<tr>
<td>Output words</td>
<td></td>
<td></td>
<td>...</td>
<td>was</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>-0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Summing/matching expects multiple roles for a single vector. For example for "book":
- To predict 'was/were', only "plurality" needs adding to 'book'
- 'book' is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles
Self-attention: Learnable word roles and modifiers

Self-attention: Learning how to re-weight words with all other words.

![Diagram of self-attention layers]

Summing/matching expects multiple roles for a single vector.
For example for $x_{book}$:
- To predict 'was/were', only "plurality" needs adding to $x_{book}$
Self-attention: Learnable word roles and modifiers

Self-attention: Learning how to re-weight words with all other words.

Summing/matching expects multiple roles for a single vector.
For example for $x_{\text{book}}$:

- To predict 'was/were', only "plurality" needs adding to $x_{\text{book}}$
- $x_{\text{book}}$ is a noun, and should be matched only to relevant other words
Self-attention: Learnable word roles and modifiers

Self-attention: Learning how to re-weight words with all other words.

Summing/matching expects multiple roles for a single vector.

For example for $x_{book}$:

- To predict 'was/were', only "plurality" needs adding to $x_{book}$
- $x_{book}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{\text{book}}$:

- To predict 'was/were', only ”plurality” needs adding to $x_{\text{book}}$
- $x_{\text{book}}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{\text{book}}$:

- To predict 'was/were', only "plurality" needs adding to $x_{\text{book}}$
- $x_{\text{book}}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{\text{book}}$:
- To predict 'was/were', only "plurality" needs adding to $x_{\text{book}}$
- $x_{\text{book}}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:
- As a part of the weighted sum for the output (value)
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{\text{book}}$:
- To predict ’was/were’, only ”plurality” needs adding to $x_{\text{book}}$
- $x_{\text{book}}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:
- As a part of the weighted sum for the output (value)
- For its own weights: compared to all others (query)
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{book}$:

- To predict 'was/were', only "plurality" needs adding to $x_{book}$
- $x_{book}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:

- As a part of the weighted sum for the output (value)
- For its own weights: compared to all others (query)
- For other vectors weights (key)
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{book}$:

- To predict 'was/were', only "plurality" needs adding to $x_{book}$
- $x_{book}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:

- As a part of the weighted sum for the output (value)
- For its own weights: compared to all others (query)
- For other vectors weights (key)

Each role has learnable parameters: $W_q$, $W_k$, $W_v$. 
Self attention: Queries, keys, values

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{\text{book}}$:

- To predict 'was/were', only "plurality" needs adding to $x_{\text{book}}$
- $x_{\text{book}}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:

- As a part of the weighted sum for the output (value)
- For its own weights: compared to all others (query)
- For other vectors weights (key)

Each role has learnable parameters: $W_q, W_k, W_v$.

Q: Where would you add $W_q, W_k, W_v$ ?
Self attention: Queries, keys, values
D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

For example for $x_{book}$:
- To predict 'was/were', only "plurality" needs adding to $x_{book}$
- $x_{book}$ is a noun, and should be matched only to relevant other words
- Other words may be matched on different roles

Each vector $x_i$ is used in 3 roles:
- As a part of the weighted sum for the output (value)
- For its own weights: compared to all others (query)
- For other vectors weights (key)

Each role has learnable parameters: $W_q, W_k, W_v$,

Q: Where would you add $W_q, W_k, W_v$ ?
A: $W_q$ and $W_k$ in the dot-product, and $W_v$ in the summing.
### Self-attention: Word re-weighting

<table>
<thead>
<tr>
<th></th>
<th>was</th>
<th>book</th>
<th>student</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>was</strong></td>
<td>0.3</td>
<td>0.9</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td><strong>book</strong></td>
<td>0.1</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>student</strong></td>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>the</strong></td>
<td>0.3</td>
<td>0.1</td>
<td>0.9</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>student</th>
<th>book</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>the</strong></td>
<td>0.3</td>
<td>1.0</td>
<td>0.1</td>
<td>-0.3</td>
</tr>
<tr>
<td><strong>student</strong></td>
<td>-0.1</td>
<td>0.3</td>
<td>-0.5</td>
<td>0.9</td>
</tr>
<tr>
<td><strong>book</strong></td>
<td>0.9</td>
<td>0.1</td>
<td>-0.3</td>
<td>0.7</td>
</tr>
<tr>
<td><strong>was</strong></td>
<td>0.7</td>
<td>0.1</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Q: Where would you add $W_q$, $W_k$, $W_v$?
Q: Where would you add \( \mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \)?
Self-attention: Word re-weighting

<table>
<thead>
<tr>
<th></th>
<th>( W_k )</th>
<th>( W_q )</th>
<th>( W_q )</th>
<th>( W_q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>was</td>
<td>0.3</td>
<td>0.9</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>book</td>
<td>0.1</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>student</td>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>the</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>the</td>
<td>0.3</td>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>student</td>
<td>-0.1</td>
<td>0.3</td>
<td>-0.5</td>
<td>-0.3</td>
</tr>
<tr>
<td>book</td>
<td>0.9</td>
<td>0.1</td>
<td>-0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>was</td>
<td>-0.3</td>
<td>0.9</td>
<td>0.7</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Self-attention: Word re-weighting

\[
\begin{align*}
\text{was} & : 0.3 \quad 0.9 \quad 0.7 \\
\text{book} & : 0.1 \quad 0.5 \quad 0.3 \\
\text{student} & : 1.0 \quad 0.3 \quad 0.1 \\
\text{the} & : 0.3 \quad 0.7 \quad 0.9
\end{align*}
\]

\[
\begin{align*}
0.3 & \quad 1.0 & \quad 0.4 & \quad -0.3 \\
-0.1 & \quad 0.3 & \quad 0.0 & \quad 0.9 \\
0.9 & \quad 0.1 & \quad 0.2 & \quad 0.7
\end{align*}
\]

Q: Where would you add \( W_v \)?
Self-attention: Word re-weighting

Q: Where would you add $W_v$?
Self-attention: Word re-weighting

\[
\begin{array}{cccccc}
\text{was} & \mathbf{W}_v & \times & \mathbf{0.1} \\
\text{book} & \mathbf{W}_v & \times & \mathbf{0.1} \\
\text{student} & \mathbf{W}_v & \times & \mathbf{0.6} \\
\text{the} & \mathbf{W}_v & \times & \mathbf{0.2} \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{the} & \text{student} & \text{book} & \text{was} \\
0.3 & 1.0 & 0.5 & 0.3 \\
-0.1 & 0.3 & 0.1 & -0.1 \\
0.9 & 0.1 & 0.2 & 0.9 \\
\end{array}
\]
Questions?
Questions?

Self-attention: Learning how to re-weight words with all other words.
Self attention: Multi-head attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Consider the sentence:

*Mary, gave, roses, to, Susan*

Q: How many relations does 'gave' have?
A: 3 relations which get summed together (no difference if Susan gave them).

Use multiple 'heads':

\[
W_r q, W_r k, W_r v, \text{ indexed by } r.
\]

Option: Narrow multi-head attention
- Divide vector in \( k \) chunks; eg: turn single 256d vector in eight 32d heads

Option: Wide multi-head attention
- Apply each head independently, concatenate and project back; eg: Run 8x a 256d vector, project 2048d back to 256d
Self attention: Multi-head attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Consider the sentence:

Mary, gave, roses, to, Susan

Q: How many relations does 'gave' have?
Self attention: Multi-head attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Consider the sentence:

Mary, gave, roses, to, Susan

Q: How many relations does 'gave' have?
A: 3 relations which get summed together (no difference if Susan gave them).
Self attention: Multi-head attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Consider the sentence:

Mary, gave, roses, to, Susan

Q: How many relations does 'gave' have?
A: 3 relations which get summed together (no difference if Susan gave them).

Use multiple 'heads': \( W^r_q, W^r_k, W^r_v \), indexed by \( r \).

Recurrance, Self-Attention and Transformers

ASCI - EDL Winterschool

Nov 25 2021 32 / 34
Consider the sentence:

\[
\text{Mary, gave, roses, to, Susan}
\]

Q: How many relations does 'gave' have?  
A: 3 relations which get summed together (no difference if Susan gave them).

Use multiple 'heads': \(W^r_q, W^r_k, W^r_v\), indexed by \(r\).

Option: Narrow multi-head attention
Consider the sentence:

Mary, gave, roses, to, Susan

Q: How many relations does 'gave' have?
A: 3 relations which get summed together (no difference if Susan gave them).

Use multiple 'heads': $W^r_q, W^r_k, W^r_v$, indexed by $r$.

Option: Narrow multi-head attention
- Divide vector in $k$ chunks; eg: turn single 256d vector in eight 32d heads
Self attention: Multi-head attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Consider the sentence:

\[ \text{Mary, gave, roses, to, Susan} \]

Q: How many relations does 'gave' have?
A: 3 relations which get summed together (no difference if Susan gave them).

Use multiple 'heads': \( W_q^r, W_k^r, W_v^r \), indexed by \( r \).

Option: Narrow multi-head attention
- Divide vector in \( k \) chunks; eg: turn single 256d vector in eight 32d heads

Option: Wide multi-head attention
Self attention: Multi-head attention

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Consider the sentence:

\[ \text{Mary, gave, roses, to, Susan} \]

Q: How many relations does 'gave' have?
A: 3 relations which get summed together (no difference if Susan gave them).

Use multiple 'heads': \( W_q^r, W_k^r, W_v^r \), indexed by \( r \).

Option: Narrow multi-head attention
- Divide vector in \( k \) chunks; eg: turn single 256d vector in eight 32d heads

Option: Wide multi-head attention
- Apply each head independently, concatenate and project back; eg: Run 8x a 256d vector, project 2048d back to 256d
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?

A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?

A: Add position information to the input vectors.

Option: Position embedding
• Embed the positions; like
  x cat, x Susan learn x 12, x 25 and sum with word vectors

Option: Position encoding
• Construct a function $f: \mathbb{N} \to \mathbb{R}^d$ and let the network learn with it.
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?

A: Add position information to the input vectors.
   Option: Position embedding
   • Embed the positions; like $x_{cat}$, $x_{Susan}$ learn $x_{12}$, $x_{25}$ and sum with word vectors
   Option: Position encoding
   • Construct a function $f: \mathbb{N} \rightarrow \mathbb{R}^d$ and let the network learn with it.
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?
A: Add position information to the input vectors.
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?
A: Add position information to the input vectors.

Option: Position embedding
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?
A: Add position information to the input vectors.

Option: Position embedding
- Embed the positions; like $x_{\text{cat}}, x_{\text{Susan}} \text{ learn } x_{12}, x_{25}$ and sum with word vectors
Self attention: Position embedding

D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?
A: Add position information to the input vectors.

Option: Position embedding
- Embed the positions; like $x_{\text{cat}}, x_{\text{Susan}}$ learn $x_{12}, x_{25}$ and sum with word vectors

Option: Position encoding
Self attention: Position embedding
D2L-book chapter 10
See: http://www.peterbloem.nl/blog/transformers

Q: If I change the word order, do I get different results?
A: No; dot-product and sum is independent of position (permutation invariant)

Q: Can you think of a solution?
A: Add position information to the input vectors.

Option: Position embedding
- Embed the positions; like $x_{\text{cat}}, x_{\text{Susan}}$ learn $x_{12}, x_{25}$ and sum with word vectors

Option: Position encoding
- Construct a function $f : \mathbb{N} \rightarrow \mathbb{R}^d$ and let the network learn with it.
Questions?

There are (too) many resources online. I like these:

• From scratch: http://www.peterbloem.nl/blog/transformers
• Also from scratch: https://e2eml.school/transformers.html
• Illustrated self-attention: https://colab.research.google.com/github/mrm8488/shared_colab_notebooks/blob/master/basic_self_attention_.ipynb
Questions?

There are (too) many resources online.

I like these:

- From scratch:
  http://www.peterbloem.nl/blog/transformers
- Also from scratch:
  https://e2eml.school/transformers.html
- Illustrated self-attention
  https://colab.research.google.com/github/mrm8488/shared_colab_notebooks/blob/master/basic_self_attention_.ipynb