Attention mechanisms

CS 585, Fall 2019
Introduction to Natural Language Processing

Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst

some slides from Richard Socher
Sequence-to-sequence: the bottleneck problem
Sequence-to-sequence: the bottleneck problem

Encoding of the source sentence. This needs to capture all information about the source sentence. Information bottleneck!

Source sentence (input)

Target sentence (output)

<START> the poor don’t have any money <END>
“you can’t cram the meaning of a whole %&@#&ing sentence into a single $*(@ing vector!”

— Ray Mooney (NLP prof at UT Austin)
idea: what if we use multiple vectors?

Instead of:
les pauvres sont démunis =

Let’s try:
les pauvres sont démunis = (all 4 hidden states!)
The solution: **attention**

- **Attention mechanisms** (Bahdanau et al., 2015) allow the decoder to focus on a particular part of the source sequence at each time step
- Conceptually similar to *word alignments*
How does it work?

• In general, we have a single query vector and multiple key vectors. We want to score each query-key pair.
Sequence-to-sequence with attention

Encoder RNN

Source sentence (input)

les pauvres sont démunis

Decoder RNN

Attention scores

dot product with keys (encoder hidden states)

Query 1: decoder, first time step
Sequence-to-sequence with attention

On this decoder timestep, we’re mostly focusing on the first encoder hidden state ("les")

Take softmax to turn the scores into a probability distribution
Sequence-to-sequence with attention

Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information the hidden states that received high attention.
Sequence-to-sequence with attention

Encoder RNN

Attention distribution

Attention scores

Source sentence (input)

les pauvres sont démunis

Decoder RNN

Attention output

Attention distribution

Attention scores

The concatenation of attention output with decoder hidden state, then use to compute \( \hat{y}_1 \) as before.

\[ \text{concatenate attention output with decoder hidden state, then use to compute } \hat{y}_1 \text{ as before} \]
Sequence-to-sequence with attention

Encoder RNN

Source sentence (input)

les pauvres sont démunis

Attention distribution

Attention scores

Attention output

Decoder RNN

Attention distribution

Attention scores

Attention output

poor

\( \hat{y}_2 \)

decoder, second time step

<START> the
Attention is great

- Attention significantly **improves NMT performance**
  - It’s very useful to allow decoder to focus on certain parts of the source
- Attention **solves the bottleneck problem**
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention **helps with vanishing gradient problem**
  - Provides shortcut to faraway states
- Attention **provides some interpretability**
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get **alignment for free**!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself
Many variants of attention

• Original formulation: \( a(q, k) = w_2^T \tanh(W_1[q; k]) \)

• Bilinear product: \( a(q, k) = q^T W k \)  
  Luong et al., 2015

• Dot product: \( a(q, k) = q^T k \)  
  Luong et al., 2015

• Scaled dot product: \( a(q, k) = \frac{q^T k}{\sqrt{|k|}} \)  
  Vaswani et al., 2017
Attention is not just for MT!
Here we have a standard seq2seq model for summarization.
Here we have a seq2seq model with a **copy mechanism** for summarization.
Target-side attention (in LMs or more complex MT models)

\[
p(Yellen) = g p_{\text{vocab}}(Yellen) + (1 - g) p_{\text{ptr}}(Yellen)
\]
Image Captioning with Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al., 2015
visual attention

- Use the question representation $q$ to determine where in the image to look

How many benches are shown?
How many benches are shown?

attention over final convolutional layer in network: 196 boxes, captures color and positional information
Hierarchical attention

Yang et al., 2016
Self-attention as an encoder!
(core component of Transformer)

Vaswani et al., 2017
Attention variants
How many benches are shown?

Attention over final convolutional layer in network: 196 boxes, captures color and positional information.

We can use reinforcement learning to focus on just one box.

Xu et al., 2015
Multi-headed attention

- Preview of next class!

- Intuition: $k$ different attentions, each of which is computed independently and focuses on different parts of the sentence

- Transformers = stacked layers of multi-headed self-attention