CS839: Probabilistic Graphical Models

Lecture 22: The Attention Mechanism

Theo Rekatsinas
Why Attention?

• Consider machine translation:
  • We need to pay attention to the word we are currently translating. Is the entire sequence needed as context?
  
  • The cat is black -> Le chat est noir
Why Attention?

• Consider machine translation:
  • We need to pay attention to the word we are currently translating. Is the entire sequence needed as context?
  • The cat is black -> Le chat est noir

• RNNs are the de-facto standard for machine translation
• Problem: translation relies on reading a complete sentence and compresses all information into a fixed-length vector a sentence with hundreds of words represented by several words will surely lead to information loss, inadequate translation, etc.
• Long-range dependencies are tricky.
Basic encoder - decoder
Soft Attention for Translation

“I love coffee” -> “Me gusta el café”
Soft Attention for Translation

Distribution over input words

“I love coffee” -> “Me gusta el café”

Soft Attention for Translation

Distribution over input words

"I love coffee" -> "Me gusta el café"

Soft Attention for Translation

Distribution over input words

“I love coffee” -> “Me gusta el café”

Soft Attention for Translation

Distribution over input words

“I love coffee” -> “Me gusta el café”

Soft Attention

\[ f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .}) \]

From Y. Bengio CVPR 2015 Tutorial
Soft Attention

Context vector (input to decoder):

\[ c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \]

Mixture weights:

\[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \]

Alignment score (how well do input words near j match output words at position i):

\[ e_{ij} = a(s_{i-1}, h_j) \]
Soft Attention

Luong, Pham and Manning’s Translation System (2015):

Thermometer Graph 1: BLEU (CASED)

Thermometer Graph 2: HTER (HE SET)

-26%
Hard Attention

- Instead of a soft interpolation, make a zero-one decision about where to attend (Xu et al. 2015)

- Harder to train, requires methods such as reinforcement learning (see later classes)

- Perhaps this helps interpretability? (Lei et al. 2016)
Monotonic Attention

- In some cases, we might know the output will be the same order as the input

- Speech recognition, incremental translation, morphological inflection (?), summarization (?)

• **Basic idea:** hard decisions about whether to read more
Global Attention

- Blue = encoder
- Red = decoder
- Attend to a context vector.
- Decoder captures global information not only the information from one hidden state.
- Context vector takes all cell’s outputs as input and computes a probability distribution for each token the decoder wants to generate.
Local Attention

• Compute a best aligned position first
• Then compute a context vector centered at that position
RNN for Captioning

Image: $H \times W \times 3$

Features: $D$

Hidden state: $H$

RNN only looks at whole image, once

What if the RNN looks at different parts of the image at each timestep?
Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Soft Attention for Captioning

Soft Attention for Captioning

Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Weighted combination of features

Distribution over L locations

a1

h0

Weighted features: D

z1
Soft Attention for Captioning

Image: $\text{H} \times \text{W} \times 3$

CNN

Features: $L \times D$

Weighted combination of features

Distribution over $L$ locations

Weighted features: $D$

First word

$h_0 \rightarrow a_1 \rightarrow h_1$

$h_0 \rightarrow z_1 \rightarrow y_1$
Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Weighted combination of features

Weighted features: D

Distribution over L locations

Distribution over vocab

First word
Soft Attention for Captioning

Image: $H \times W \times 3$

Features:
- $L \times D$

Weighted combination of features

CNN

Distribution over L locations

$a_1$ $a_2$ $d_1$

Distribution over vocab

$h_0$ $h_1$

Weighted features:
- $D$

First word

$z_1$ $y_1$ $z_2$
Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Weighted combination of features

Weighted features: D

Distribution over L locations

Distribution over vocab

First word

h0 → a1 → h1

h0 → a2 → h1

h0 → d1 → h1

h1 → h2

h1 → z1 → y1

h1 → z2 → y2
Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Weighted combination of features

Weighted features: D

First word

Distribution over locations

Distribution over vocab
Soft vs Hard Attention

Image: $H \times W \times 3$

Grid of features (Each D-dimensional)

From RNN:

Distribution over grid locations

$p_a + p_b + p_c + p_d = 1$

Soft vs Hard Attention

Image:
H x W x 3

Grid of features
(Each D-dimensional)

From RNN:

Context vector z
(D-dimensional)

Distribution over grid locations
\[ p_a + p_b + p_c + p_d = 1 \]

Soft vs Hard Attention

From RNN:

Image: H x W x 3

Grid of features (Each D-dimensional)

From RNN:

Distribution over grid locations
\[ p_a + p_b + p_c + p_d = 1 \]

Soft attention:
Summarize ALL locations
\[ z = p_a a + p_b b + p_c c + p_d d \]

Derivative \( dz/dp \) is nice!
Train with gradient descent

Context vector \( z \) (D-dimensional)
Soft vs Hard Attention

**Soft attention:**
Summarize ALL locations
\[ z = p_a a + p_b b + p_c c + p_d d \]
Derivative \( dz/dp \) is nice!
Train with gradient descent

**Hard attention:**
Sample ONE location according to \( p \), \( z = \text{that vector} \)

With argmax, \( dz/dp \) is zero almost everywhere ...
Can’t use gradient descent; need reinforcement learning

From RNN:

\[ p_a \quad p_b \]
\[ p_c \quad p_d \]

Distribution over grid locations
\[ p_a + p_b + p_c + p_d = 1 \]

Image:

\[ H \times W \times 3 \]

Grid of features (Each D-dimensional)

CNN

Multi-headed Attention

- **Idea:** multiple attention “heads” focus on different parts of the sentence

- e.g. Different heads for “copy” vs regular (Allamanis et al. 2016)

- Or multiple independently learned heads (Vaswani et al. 2017)

<table>
<thead>
<tr>
<th>Target</th>
<th>Attention Vectors</th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$ set</td>
<td>$\alpha =$ <code>&lt;$&gt;$ { this, use Browser Cache = use Browser Cache; } &lt;$/&gt;</code></td>
<td>0.012</td>
</tr>
<tr>
<td>$m_2$ use</td>
<td>$\alpha =$ <code>&lt;$&gt;$ { this, use Browser Cache = use Browser Cache; } &lt;$/&gt;</code></td>
<td>0.974</td>
</tr>
<tr>
<td>$m_3$ browser</td>
<td>$\alpha =$ <code>&lt;$&gt;$ { this, use Browser Cache = use Browser Cache; } &lt;$/&gt;</code></td>
<td>0.069</td>
</tr>
<tr>
<td>$m_4$ cache</td>
<td>$\alpha =$ <code>&lt;$&gt;$ { this, use Browser Cache = use Browser Cache; } &lt;$/&gt;</code></td>
<td>0.583</td>
</tr>
<tr>
<td>$m_5$ END</td>
<td>$\alpha =$ <code>&lt;$&gt;$ { this, use Browser Cache = use Browser Cache; } &lt;$/&gt;</code></td>
<td>0.066</td>
</tr>
</tbody>
</table>
Attention is all you need

Summary of the "Transformer"
(Vaswani et al. 2017)

- A sequence-to-sequence model based entirely on attention
- Strong results on standard WMT datasets
- Fast: only matrix multiplications
Attention tricks

- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Normalized Dot-product Attention:** Remove bias in dot product when using large networks
- **Positional Encodings:** Make sure that even if we don’t have RNN, can still distinguish positions
Attention Takeaways

**Performance:**
- Attention models can *improve accuracy* and *reduce computation* at the same time.

**Complexity:**
- There are many design choices.
- Those choices have a big effect on performance.
- Ensembling has unusually large benefits.
- Simplify where possible!
Attention Takeaways

**Explainability:**
- Attention models encode explanations.
- Both locus and trajectory help understand what’s going on.

**Hard vs. Soft:**
- Soft models are easier to train, hard models require reinforcement learning.
- They can be combined, as in Luong et al.