Attentive Sequence-to-Sequence Learning

Jindřich Helcl, Jindřich Libovický

March 6, 2018
Neural Network Language Models
• Train RNN as classifier for next words (unlimited history)

• Can be used to estimate sentence probability / perplexity → defines a distribution over sentences

• We can sample from the distribution
Two views on RNN LM

- RNN is a for loop (functional map) over sequential data
- All outputs are conditional distributions \( \rightarrow \) probabilistic distribution over sequences of words

\[
P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1}, \ldots, w_1)
\]
Vanilla Sequence-to-Sequence Model
• Exploits the conditional LM scheme
• Two networks
  1. A network processing the input sentence into a single vector representation (*encoder*)
  2. A neural language model initialized with the output of the encoder (*decoder*)

Encoder-Decoder – Image

Source language input + target language LM
state = np.zeros(emb_size)
for w in input_words:
    input_embedding = source_embeddings[w]
    state, _ = enc_cell(encoder_state,
                        input_embedding)

last_w = "<s>"
while last_w != "</s>":
    last_w_embedding = target_embeddings[last_w]
    state, dec_output = dec_cell(state,
                                  last_w_embedding)
    logits = output_projection(dec_output)
    last_w = np.argmax(logits)
yield last_w
Encoder-Decoder Model – Formal Notation

Data
input embeddings (source language) \( \mathbf{x} = (x_1, \ldots, x_{T_x}) \)
output embeddings (target language) \( \mathbf{y} = (y_1, \ldots, y_{T_y}) \)

Encoder
initial state \( h_0 \equiv \mathbf{0} \)
j-th state \( h_j = \text{RNN}_{\text{enc}}(h_{j-1}, x_j) \)
final state \( h_{T_x} \)

Decoder
initial state \( s_0 = h_{T_x} \)
\( i \)-th decoder state \( s_i = \text{RNN}_{\text{dec}}(s_{i-1}, \hat{y}_i) \)
\( i \)-th word score \( t_{i+1} = U_o + V_o E y_i + b_o, \)
\hspace{1cm} \text{or multi-layer projection}
output \( \hat{y}_{i+1} = \text{arg max} t_{i+1} \)
For output word $y_i$ we have:

- Estimated conditional distribution $\hat{p}_i = \frac{\exp t_i}{\sum \exp t_i}$ (softmax function)
- Unknown true distribution $p_i$, we lay $p_i \equiv 1[y_i]$

Cross entropy $\approx$ distance of $\hat{p}$ and $p$:

$$\mathcal{L} = H(\hat{p}, p) = E_p(-\log \hat{p}) = -\log \hat{p}(y_i)$$

...computing $\frac{\partial \mathcal{L}}{\partial t_i}$ is super simple
Implementation: Runtime vs. training

\[ \hat{y}_j \quad (\text{decoded}) \times \quad y_j \quad (\text{ground truth}) \]
Sutskever et al.

- Reverse input sequence
- Impressive empirical results – made researchers believe NMT is way to go

Evaluation on WMT14 EN → FR test set:

<table>
<thead>
<tr>
<th>method</th>
<th>BLEU score</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla SMT</td>
<td>33.0</td>
</tr>
<tr>
<td>tuned SMT</td>
<td>37.0</td>
</tr>
<tr>
<td>Sutskever et al.: reversed</td>
<td>30.6</td>
</tr>
<tr>
<td>−”−: ensemble + beam search</td>
<td>34.8</td>
</tr>
<tr>
<td>−”−: vanilla SMT rescoring</td>
<td>36.5</td>
</tr>
<tr>
<td>Bahdanau’s attention</td>
<td>28.5</td>
</tr>
</tbody>
</table>

*Why is better Bahdanau’s model worse?*
Sutskever et al. × Bahdanau et al.

### Sutskever et al. vs. Bahdanau et al.

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>Sutskever et al.</th>
<th>Bahdanau et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>160k enc, 80k dec</td>
<td>30k both</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Sutskever et al.</th>
<th>Bahdanau et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$4 \times$ LSTM, 1,000 units</td>
<td>bidi GRU, 2,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decoder</th>
<th>Sutskever et al.</th>
<th>Bahdanau et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$4 \times$ LSTM, 1,000 units</td>
<td>GRU, 1,000 units</td>
</tr>
<tr>
<td>Word Embeddings</td>
<td>1,000 dimensions</td>
<td>620 dimensions</td>
</tr>
<tr>
<td>Training Time</td>
<td>7.5 epochs</td>
<td>5 epochs</td>
</tr>
</tbody>
</table>

With Bahdanau’s model size:

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder-decoder</td>
<td>13.9</td>
</tr>
<tr>
<td>Attention model</td>
<td>28.5</td>
</tr>
</tbody>
</table>
Attentive Sequence-to-Sequence Learning
Main Idea

• Same as reversing input: do not force the network to catch long-distance dependencies
• Use decoder state only for target sentence dependencies and a as query for the source word sentence
• RNN can serve as LM — it can store the language context in their hidden states
Inspiration: Neural Turing Machine

- General architecture for learning algorithmic tasks, finite imitation of a Turing Machine
- Needs to address memory somehow – either by position or by content

- In fact does not work well – it hardly manages simple algorithmic tasks
- Content-based addressing $\rightarrow$ attention
Attention Model

Attentive Sequence-to-Sequence Learning
Attention Model in Equations (1)

Inputs:
- decoder state $s_i$
- encoder states $h_j = [\vec{h}_j; \vec{h}_j] \quad \forall i = 1 \ldots T_x$

Attention energies:

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j + b_a)$$

Attention distribution:

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

Context vector:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$
Output projection:

\[ t_i = \text{MLP}(U_os_{i-1} + V_o E_y_{i-1} + C_o c_i + b_o) \]

...context vector is mixed with the hidden state

Output distribution:

\[ p(y_i = w|s_i, y_{i-1}, c_i) \propto \exp(W_o t_i)_w + b_w \]
Attention Visualization

Attentive Sequence-to-Sequence Learning
## Attention vs. Alignment

Differences between attention model and word alignment used for phrase table generation:

<table>
<thead>
<tr>
<th>attention (NMT)</th>
<th>alignment (SMT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>probabilistic</td>
<td>discrete</td>
</tr>
<tr>
<td>declarative</td>
<td>imperative</td>
</tr>
<tr>
<td>LM generates</td>
<td>LM discriminates</td>
</tr>
</tbody>
</table>
Image Captioning

Attention over CNN for image classification:

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.


Question:
The model uses the scaled dot-product attention which is a non-parametric variant of the attention mechanism. Why do you think it is sufficient in this setup? Do you think it would work in the recurrent model as well?

The way the model processes the sequence is principally different from RNNs or CNNs. Does it agree with your intuition of how language should be processed?