Sequence-to-Sequence Learning
March 15, 2017

Jindřich Libovický, Jindřich Helcl

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
RNN Language Model

- 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 → probabilistic distribution over sequences of words

\[ P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1}, \ldots, w_1) \]
source language LM + 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) $x = (x_1, \ldots, x_{T_x})$
- output embeddings (target language) $y = (y_1, \ldots, y_{T_y})$

**Encoder**
- initial state $h_0 \equiv 0$
- $j$-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 $t_{i+1} = U_o + V_o E y_{i} + b_o,$
  or multi-layer projection
- output $\hat{y}_{i+1} = \arg \max t_{i+1}$
Encoder-Decoder: Training Objective

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 $p$:

$$L = H(\hat{p}, p) = \mathbb{E}_p (- \log \hat{p}) = - \log \hat{p}(y_i)$$

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

**runtime:** $\hat{Y}_j$ (decoded) \[\times\] **training:** $Y_j$ (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.

<table>
<thead>
<tr>
<th>Sutskever et al.</th>
<th>Bahdanau et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td>vocabulary</td>
<td>160k enc, 80k dec</td>
</tr>
<tr>
<td>encoder</td>
<td>$4 \times$ LSTM, 1,000 units</td>
</tr>
<tr>
<td>decoder</td>
<td>$4 \times$ LSTM, 1,000 units</td>
</tr>
<tr>
<td>word embeddings</td>
<td>1,000 dimensions</td>
</tr>
<tr>
<td>training time</td>
<td>7.5 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>
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 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 → attention
Small Trick before We Start

Bidirectional network

- read the input sentence from both sides
- every $h_i$ contains in fact information from the whole sentence
Attention Model

\[ <s> \]

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ x_4 \]

\[ \alpha_0 \times \alpha_1 \times \alpha_2 \times \alpha_3 \times \alpha_4 \]

\[ s_{i-1} \]
\[ s_i \]
\[ s_{i+1} \]

\[ \sim y_i \]
\[ \sim y_{i+1} \]
**Attention Model in Equations (1)**

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

**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:**
\[
c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j
\]
**Attention Model in Equations (2)**

**Output projection:**

\[ t_i = \text{MLP} (U_o s_{i-1} + V_o E y_{i-1} + C_o c_i + b_o) \]

...attention is mixed with the hidden state

**Output distribution:**

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

The agreement on the European Economic Area was signed in August 1992.

Il convient de noter que l'environnement marin est le moins connu de l'environnement.
Attention vs. Alignment

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

**attention (NMT)**
- probabilistic
- declarative
- LM generates

**alignment (SMT)**
- discrete
- imperative
- LM discriminates
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:
What are the reasons authors do not use character-level encoder? How would you improve the architecture such that it would allow character level encoding?